Trabalho de Conclusão de Curso

Segurança na transmissão de dados: Uma abordagem utilizando criptografia e esteganografia de imagem

Ayrton Oliveira Ouriques
aoo@ic.ufal.br

Orientador:

Prof. Dr. Almir Pereira Guimarães

Maceió, março de 2024

Ayrton Oliveira Ouriques

Segurança na transmissão de dados: Uma abordagem utilizando criptografia e esteganografia de imagem

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação pelo Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Almir Pereira Guimarães

Maceió, março de 2024

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Helena Cristina Pimentel do Vale – CRB4 - 661

O93s Ouriques, Ayrton Oliveira.
Segurança na transmissão de dados : uma abordagem utilizando criptografia e esteganografia de imagem / Ayrton Oliveira Ouriques. – 2024.
49 f.: il.

Orientador: Almir Pereira Guimarães.
Monografia (Trabalho de Conclusão de Curso em Ciências da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Graduação em Ciência da Computação. Maceió, 2024.

Bibliografia: f. 45-49.

1. Criptografia. 2. Esteganografia de imagem. 3. Processamento de imagem. 4. Segurança da informação. 5. Transmissão de dados. I. Título.

CDU: 004.383.5

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação pelo Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Prof. Dr. Almir Pereira Guimarães - Orientador
Universidade Federal de Alagoas

Prof. Dr. nome - Examinador
local
universidade

Prof. Dr. nome - Examinador
local
universidade

Maceió, março de 2024

Agradecimentos

Primeiramente, agradeço a Deus, que me concedeu a oportunidade e a capacidade para concluir esta etapa importante da minha vida. Aos meus pais, expresso minha mais profunda gratidão. Vocês foram a minha fortaleza e minha motivação durante toda essa jornada. Obrigado por acreditarem em mim. Ao meu orientador, Almir Pereira, agradeço pela paciência e disponibilidade, pelo conhecimento compartilhado e pela orientação precisa. Sua sabedoria e experiência foram fundamentais para o desenvolvimento deste trabalho. Aos meus amigos, agradeço pelo apoio, pelas risadas nos momentos de estresse e pela descontração. Vocês tornaram essa jornada muito mais agradável. Por fim, agradeço à minha faculdade, que me proporcionou um ambiente de aprendizado rico e estimulante. Agradeço a todos os professores e funcionários que contribuíram para a minha formação. Este trabalho é dedicado a todos vocês.

Resumo

Este trabalho aborda a questão da proteção de dados no contexto moderno, onde a transmissão segura de dados é cada vez mais necessária devido ao rápido aumento na geração e compartilhamento de dados. A necessidade de proteger esses dados de ameaças externas e internas é fundamental para o funcionamento de ambientes de transmissão de dados. Neste contexto, o estudo se concentra na transmissão segura de dados por meio da combinação de técnicas de criptografia e esteganografia. Ele é iniciado com uma revisão dos fundamentos teóricos dessas áreas, destacando conceitos importantes como protocolos de comunicação, algoritmos criptográficos e métodos de ocultação esteganográficos. Em seguida, apresento o desenvolvimento de uma ferramenta prática de esteganografia de imagem, detalhando sua arquitetura, funcionamento e algoritmos utilizados. Foram realizados testes rigorosos para avaliar o desempenho da ferramenta, considerando métricas como MSE, PSNR, SSIM e capacidade de carga em diferentes tamanhos de imagem. Os resultados obtidos forneceram dados valiosos sobre a eficácia da esteganografia da ferramenta na proteção da confidencialidade dos dados e na integridade de sua comunicação digital. Concluí destacando o que foi descoberto nos testes e propondo direções para futuras pesquisas e desenvolvimentos da ferramenta.

Palavras-chave: Criptografia; Esteganografia de Imagem; Processamento de Imagem; Segurança da Informação; Transmissão de Dados.

Abstract

This paper addresses the issue of data protection in the modern context, where secure data transmission is increasingly necessary due to the rapid increase in data generation and sharing. The need to protect this data from external and internal threats is fundamental to the functioning of data transmission environments. In this context, the study focuses on secure data transmission through the combination of encryption and steganography techniques. It begins with a review of the theoretical foundations of these areas, highlighting important concepts such as communication protocols, cryptographic algorithms, and steganographic hiding methods. Next, I present the development of a practical image steganography tool, detailing its architecture, operation, and algorithms used. Rigorous tests were conducted to evaluate the performance of the tool, considering metrics such as MSE, PSNR, SSIM, and payload capacity across different image sizes. The results obtained provided valuable data on the effectiveness of the tool's steganography in protecting data confidentiality and the integrity of its digital communication. I conclude by highlighting what was discovered in the tests and proposing directions for future research and development of my tool.

Key-words: Cryptography; Image Steganography; Image Processing; Information Security; Data Transmission.

Lista de Figuras

Conteúdo

1

Introdução

1.1 Visão Geral

Na era contemporânea da comunicação digital, a transmissão segura de dados representa um desafio crucial diante das crescentes ameaças à privacidade e à integridade das informações. É essencial a segurança de dados no contexto atual, onde a quantidade de dados gerados e compartilhados está aumentando exponencialmente. Com o advento da Internet das Coisas (IoT), Big Data e computação em nuvem, estima-se que 2,5 quintilhões de bytes de dados sejam criados a cada dia, e esse número só tende a crescer com o avanço da tecnologia digital[4]. A necessidade de proteger esses dados de ameaças externas e internas é mais importante do que nunca.

Esses dados, que variam de informações pessoais a segredos comerciais, estão constantemente sob ameaça de serem interceptados ou manipulados por atores mal-intencionados. As ameaças podem vir de fontes externas, como hackers e cibercriminosos, ou de fontes internas, como funcionários descontentes ou negligentes[5]. A violação desses dados pode resultar em perdas financeiras significativas, danos à reputação e perda de confiança do cliente. Além disso, a violação de dados pessoais pode levar a problemas de privacidade e até mesmo a crimes como roubo de identidade.

Este trabalho exploratório visa abordar essa problemática da transmissão de dados, destacando a relevância da criptografia e esteganografia como ferramentas na salvaguarda da confidencialidade e na ocultação de informações sensíveis. Ao se aprofundar em segurança criptográfica e esteganográfica, esta pesquisa, junto de uma ferramenta desenvolvida, busca fornecer uma compreensão abrangente de estratégias para proteger a comunicação em ambientes vulneráveis.

1.2 Motivação

A crescente dependência da comunicação digital expõe informações sensíveis a riscos significativos de interceptação e manipulação. Diante desse contexto, a motivação para este estudo reside na necessidade de estudar abordagens que fortaleçam a segurança da transmissão de dados. A cada dia, milhões de transações digitais são realizadas, desde a troca de mensagens pessoais até a transferência de grandes volumes de dados corporativos. Cada uma dessas transações carrega consigo a necessidade de segurança, a garantia de que as informações transmitidas cheguem ao destinatário pretendido sem serem lidas, alteradas ou forjadas por terceiros não autorizados.

Ao explorar a interseção entre criptografia e esteganografia, pretendemos oferecer soluções robustas para mitigar essas ameaças, promovendo a confiança em trocas de informações privadas.

Além disso, a motivação para este estudo também vem do desejo de contribuir para a comunidade científica e tecnológica. Acreditamos que, ao aprofundar nossa compreensão da criptografia e esteganografia, podemos ajudar a melhorar o futuro da segurança de dados. Com a crescente comunicação digital e, subsequentemente, o crescimento de ataques cibernéticos e violações de dados, nunca foi tão importante investir em pesquisas e desenvolvimentos nesta área. Portanto, este estudo representa nosso compromisso em fazer nossa parte para proteger a integridade das comunicações digitais.

1.3 Objetivos

Os objetivos deste trabalho incluem a análise aprofundada dos conceitos fundamentais de criptografia e esteganografia, destacando suas aplicações práticas na proteção da informação. Além disso, pretendemos usar uma ferramenta para demonstrar os conceitos duma maneira prática e avaliar a eficácia da técnica de estaganografia de imagem RPE (Random Pixel Embedding), explorando casos de uso e identificando possíveis limitações. Ao final, busca-se fornecer insights valiosos para profissionais de segurança da informação e pesquisadores interessados na proteção efetiva da comunicação digital utilizando criptografia e esteganografia.

Durante a produção deste TCC, os objetivos específicos para o desenvolvimento do trabalho incluíram a pesquisa e criação da ferramenta de ocultação de dados, depois foi iniciado o estudo para a realização da fundamentação teórica. Após a conclusão da fundamentação, foi realizada a pesquisa e bateria de testes sobre o programa.

1.4 Estrutura do TCC

Este trabalho está estruturado em diversos capítulos, começando pela fundamentação teórica,

serão abordados conceitos e algoritmos específicos, seguidos por análises comparativas. Em seguida, é mostrado o uso da ferramenta desenvolvida para demonstrar exemplos de cenários de transmissão usando criptografia e esteganografia. Depois, serão reunidos os resultados do uso da ferramenta, com estudos de caso e análises utilizando métricas de processamento de imagem para interpretar e avaliar os resultados da ferramenta. A conclusão destacará as descobertas mais relevantes, oferecendo recomendações práticas e delineando possíveis direções para pesquisas futuras. Com essa estrutura, esperamos proporcionar uma visão abrangente e aprofundada sobre o tópico, contribuindo para a evolução contínua das práticas de segurança digital.

2

Fundamentação Teórica

2.1 Transmissão de Dados

A transmissão de dados é um componente fundamental no âmbito da comunicação digital no mundo atual, representando o processo pelo qual informações são transferidas de uma fonte para um destino por meio de um canal de comunicação, o mais proeminente sendo a internet. Este processo é essencial em diversos contextos, desde comunicações cotidianas até transações comerciais e trocas de informações sensíveis em ambientes corporativos ou governamentais. Sendo isso, devido a crescente digitalização de informações e a ubiquidade da conectividade cria-se uma demanda de métodos robustos e seguros para garantir a integridade, confidencialidade e disponibilidade dos dados transmitidos.

2.1.1 Desafios na Transmissão de Dados

A transmissão de dados enfrenta desafios significativos relacionados à segurança, principalmente quando ocorre em ambientes inseguros, tais como redes públicas e ambientes online suscetíveis a ataques cibernéticos. Entre os desafios comuns estão a garantia da confidencialidade, da autenticidade e da integridade dos dados sendo trocadas[6].

Segundo a referência [7], esses 3 conceitos são princípios da segurança da informação, são eles:

1. **Confidencialidade:**

   Este princípio concentra-se em garantir que informações sensíveis sejam protegidas contra acesso não autorizado. As medidas de confidencialidade visam prevenir a divulgação não autorizada de dados para manter a privacidade e o sigilo. Técnicas como criptografia, controles de acesso e autenticação de usuários são comumente usadas para impor a confidencialidade.

2. **Integridade:**

   A integridade garante que os dados permaneçam precisos, completos e não adulterados. Envolve proteger os dados contra modificação, exclusão ou corrupção não autorizada. As medidas de integridade verificam a confiabilidade dos dados ao longo de seu ciclo de vida, muitas vezes por meio de mecanismos como checksums (*hashing*), assinaturas digitais e controles de acesso.

3. **Autenticidade:**

   A autenticidade está relacionada à verificação da identidade de usuários, dispositivos ou recursos para garantir que sejam genuínos e confiáveis. Alguns métodos comuns de garantir autenticidade incluem o uso de credenciais de usuário (como senhas ou chaves de autenticação), certificados digitais, biometria e técnicas de verificação de identidade.

Em resumo, ter esses pilares da segurança da informação em sistemas de transmissões de dados é essencial para proteger os dados e os ativos da organização, garantindo sua integridade, confidencialidade e autenticidade em todos os níveis.

Para ajudar na implementação desses pilares, diversos protocolos foram desenvolvidos para regular a transmissão de dados. Protocolos como o HTTPS (Hypertext Transfer Protocol Secure)[8], desempenham papéis cruciais na garantia da integridade, confidencialidade e autencidade durante a transmissão de dados pela internet. Por sua vez, a criptografia desempenha um papel fundamental na proteção dos dados durante transmissões, sendo utilizada, por exemplo, no HTTPS e no protocolo SSL/TLS (Secure Sockets Layer/Transport Layer Security) que incorporam técnicas criptográficas para garantir os requisitos de segurança da informação[8].

2.2 Criptografia

A criptografia é uma disciplina essencial no domínio da segurança da informação, desempenhando um papel fundamental na proteção e garantia da confidencialidade e integridade dos dados. Por meio da aplicação de algoritmos matemáticos, a criptografia transforma dados legíveis em uma forma ilegível, conhecida como texto cifrado, tornando-os inúteis para qualquer pessoa sem o método de decifração correto. Num ambiente computacional, além de ser essencial na transmissão de informações segura, a criptografia é fundamental em contextos diversos, como armazenamento seguro de dados, autenticação de usuários e assinaturas digitais[9].

2.2.1 Criptografia Simétrica

De acordo com a referência [10], a criptografia é dividida em dois tipos, simétrica e assimétrica, na simétrica uma única chave é utilizada tanto para criptografar quanto para descriptografar os dados. Este método é eficiente, rápido e amplamente empregado em sistemas nos

quais uma confiança mútua já foi estabelecida entre as partes envolvidas. Por outro lado, há uma necessidade da transferência da chave entre as partes para que a troca de dados possa começar, isso traz mais insegurança e mais dificuldade dependendo de quantas partes irão receber a chave. A segurança da criptografia simétrica reside no algoritmo escolhido e na manutenção cuidadosa da chave compartilhada, uma vez que a perda ou comprometimento dela comprometeria o sistema[10].

Na criptografia simétrica, de acordo com [1], encontramos dois conceitos centrais: cifragem e decifragem. A cifragem envolve a transformação dos dados originais em um formato criptografado, enquanto a decifragem é o processo reverso, no qual os dados cifrados são convertidos de volta para seu estado original. Tanto a cifragem quanto a decifragem são realizadas executando um passo-a-passo bem definido que pode ser seguido para cifrar ou decifrar algo. Um exemplo simples de uma cifra é a cifra monoalfabética, na qual cada letra do alfabeto é substituída por qualquer outra letra fixa, por exemplo, se "A"for substituído por "C", todas as ocorrências de "A"no texto original serão substituídas por "C"no texto cifrado[1]. Seguindo a cifra abaixo, a frase "Estou escondido"viraria "Dfual dfmarnkna".

- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z - Alfabeto
- C Q M N D Y O J K B S W V R A Z X H F U L T G P E I - Cifra

Tendo isso, apesar de ter 26! (da ordem de 10^{26}) totais possivéis combinações, algo que seria custoso de se atacar utilizando força-bruta, a cifra monoalfabética é vulnerável a ataques de frequência de letras, nos quais a frequência de ocorrência de letras no texto cifrado é analisada para deduzir as correspondências mais prováveis com o alfabeto original, ou também ataques em que já se sabe certos conteúdos da mensagem, como nome de envolvidos ou lugares, o que ajuda na análise para replicar a cifra[1].

Criptografia em Blocos

No meio computacional, o método mais utilizado para criptografia é o de blocos, no qual os dados são divididos em blocos fixos de tamanho k e cada bloco é criptografado separadamente. Por exemplo, se $k = 64$, então a mensagem é dividida em blocos de 64 bits, e cada bloco é criptografado independentemente. Para codificar um bloco, a cifra utiliza uma correspondência de *one-to-one* (um em um) para mapear o bloco de 64 bits do texto puro para um bloco de 64 bits do texto cifrado[10].

Nos algoritmos modernos, de acordo com a referência [1], cifras de bloco geralmente usam funções que simulam tabelas permutadas aleatoriamente, onde os blocos são divididos em subblocos que, de acordo com a chave utilizada, tem seus bits embaralhados e depois juntados de volta em blocos, este processo se repete n vezes até resultar num bloco criptografado[1].

A segurança de criptografia em blocos depende do tamanho da chave utilizada, por exemplo, no algoritmo DES (Data Encryption Standard)[11] que usa chaves de tamanho 56 bits temos

2^{56} possibilidades de chave, algo que atualmente pode ser quebrado por força bruta em torno de 2 dias[12], já no algoritmo AES(Advanced Encryption System)[13] que a menor chave tem tamanho de 128 bits com 2^{128} possibilidades de chave, se a compararmos com a chave do DES de 56 bits, temos uma estimativa de 1 segundo para 1 trilhão de anos, ou seja, uma máquina que quebraria uma chave DES de 56 bits em 1 segundo demoraria 1 trilhão de anos para quebrar uma chave AES de 128 bits, tendo isso, com a tecnologia atual, demoraria em torno de 170 quadrilhões de anos para quebrar uma chave de 128 bits AES por força bruta[14].

Sendo assim, na imagem abaixo, temos um exemplo dum diagrama para criptografia simétrica num âmbito computacional, onde ambos Alice e Bob tem a mesma chave.

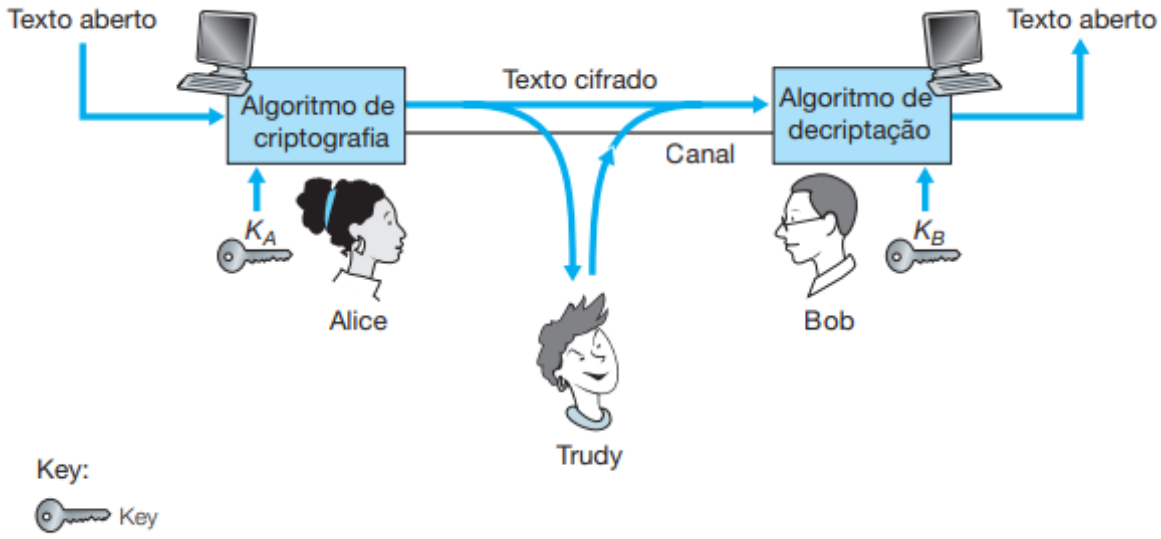

Figura 2.1: Diagrama de criptografia simétrica, fonte: [1]

AES (Advanced Encryption System)

Um exemplo de algoritmo de criptografia simétrica é o AES[13], vencedor da competição da NIST (National Institute of Standards and Technology) dos Estados Unidos de 1997, para substituir o DES[11]. Ele é usado mundialmente na maioria tecnologias de armazamento e troca de dados, desde sistemas bancários e governamentais até aplicativos de mensagem e redes Wi-fi, ele é escolhido devido a sua impenetrabilidade, simplicidade de uso e rapidez[15] .

De acordo com a referência [2], o AES funciona com base na criptografia por blocos de tamanho de 128 bits e aceita chaves de 128, 192 e 256 bits.

Além de ser altamente seguro e simples de implementar, o AES é rápido na sua execução, pois é altamente paralelizável, o que significa que múltiplos blocos de dados podem ser criptografados ou descriptografados simultaneamente. E também, a maioria dos processadores atuais já vêm com instruções para implementar eficientemente o AES, oque o torna ainda mais eficaz[2].

Um questionamento do uso do AES é o fato de como criptografar dados que não caibam perfeitamente em blocos de 128 bits, ou seja, quando sobra espaço no bloco final. Para resolver isso, segundo a referência [2], é usado uma técnica de preenchimento (padding) e também modos de operação. O preenchimento é uma ideia simples, caso reste espaço vazio no fim de um bloco a ser cifrado ele é preenchido com certos bytes até completar o bloco, esses bytes dependem do tipo de preenchimento escolhido, o mais comum é o PKCS7 que popula os bytes finais com o tamanho total do preenchimento, então se falta 5 bytes para terminar um bloco o PKCS7 preenche esses 5 bytes com o valor 5, isso ajuda na hora de remover o preenchimento na decifragem. O mais importante no preenchimento é garantir que ele possa ser removido[2].

A outra ferramente utilizada no manuseio dos blocos do AES são os modos de operação, segundo [2], eles atuam como um jeito de alterar como os blocos criptografados são concatenados e transportados, no modo de operação mais simples, o ECB (Electronic Codebook), os blocos são cifrados independentemente e depois juntos em sequência, o problema desse método é que ele deixa vazar informações dos dados originais devido a análise de padrões, como por exemplo no pinguim do ECB[2]:

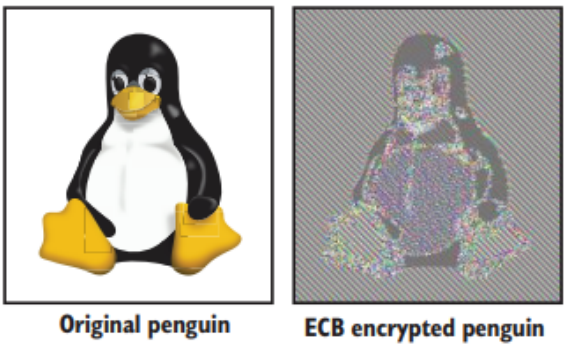

Figura 2.2: Pinguim do ECB, fonte: [2]

Ainda de acordo com [2], outros modos de operação, como o CBC (Cipher Block Chaining) utilizam operações com XOR e um vetor de inicialização (IV) para randomizar a posição dos blocos cifrados no produto final, o que o torna mais seguro que o ECB. Sendo assim, um obstáculo que surgiu ao usar o CBC foi a potencial falta de autenticidade e integridade do dados criptografados, pois um atacante pode modificar bits dos dados criptografados ou do IV o que alteraria a mensagem decifrada. A solução disso foi utilizar autenticadores de mensagem com hash (HMACs) junto ao modo de operação, os HMACs servem tanto para verificar a autenticidade dos dados quanto para garantir sua integridade, eles funcionam como uma assinatura digital e uma hash, que serve como um meio de resumir os dados[2].

2.2.2 Criptografia Assimétrica

Em contraste da simétrica, a criptografia assimétrica, também conhecida como criptografia de chave pública, utiliza um par de chaves distintas: uma chave pública e uma chave privada. De acordo com a referência [10], a chave pública é divulgada amplamente e usada para criptografar os dados, enquanto a chave privada, mantida em segredo pelo emissor, é utilizada para a decifragem. Este método oferece uma solução elegante para desafios associados à distribuição segura de chaves em grandes redes, mas é mais custoso e complexo comparado à sua contraparte. A criptografia assimétrica é frequentemente empregada em conjunto com a criptografia simétrica, criando uma abordagem híbrida que combina a eficiência da simétrica com a segurança da assimétrica[10].

Um exemplo de uso dessa combinação é vista no protocolo HTTPS, onde o protocolo TLS subjacente primeiro usa criptografia assimétrica para verificar a identidade de uma ou mais partes, depois a usa para trocar uma chave simétrica entre os envolvidos, isso garante a segurança do processo e permite que o resto da comunicação seja por criptografia simétrica, que é mais rápida e eficiente[8].

Abaixo, temos um exemplo dum diagrama para criptografia assimétrica, nele Bob gerou o par de chaves e divulgou a sua chave pública. Alice cifrou uma mensagem com ela e depois Bob a decifrou usando sua chave privada.

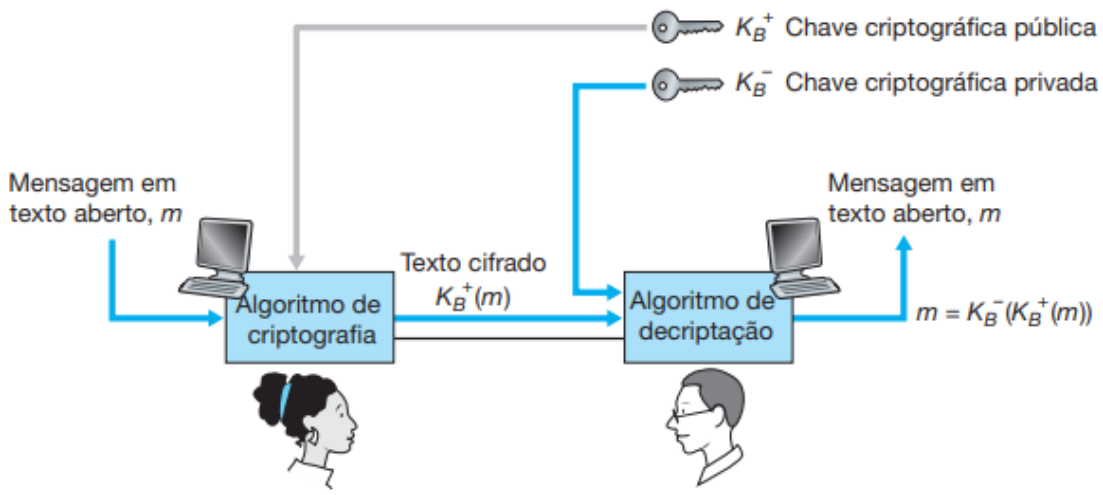

Figura 2.3: Diagrama de criptografia assimétrica, fonte: [1]

RSA (Rivest-Shamir-Adleman)

O RSA[16] é um dos algoritmos de criptografia assimétrica mais conhecidos e amplamente utilizados. Ele é utilizado para criptografia de dados, assinaturas digitais e estabelecimento de chaves para troca segura de informações em redes. Segundo [1], o RSA funciona de uma maneira geral executando as seguintes funções:

- **Geração de Chaves:** A geração de chaves RSA envolve os seguintes passos:

  1. Escolha dois números primos grandes, p e q. Quanto maiores os valores, mais difícil é quebrar o RSA, mas mais tempo leva para realizar a codificação e decodificação. O recomendado é que o produto de p e q seja da ordem de no mínimo 2.048 bits.
  2. Calcule $n = p * q$ e $z = (p-1) * (q-1)$.
  3. Escolha um número, e, menor que n, que não tenha fatores comuns (além de 1) com z ,neste caso, e e z são ditos primos entre si. Na maioria das aplicações, é escolhido o número primo 65,537 devido a ser um número grande o suficiente, mas que não prejudica muito o desempenho do algoritmo.
  4. Calcule um número, d, de forma que $e * d - 1$ seja exatamente divisível , ou seja, sem resto, por z. Em outras palavras, dado e, escolhemos d de modo que:
     $$e * d mod z = 1.$$
     Para achar o número d, é usado o teorema de Euler.
  5. A chave pública pronta para ser distribuída vai ser o par de números (n,e) e a chave privada vai ser o par de números (n,d)

- **Criptografia:** Digamos que tenhamos uma sequência de bits que é igual um número m, sendo $m < n$, o valor criptografado cm de m vai ser igual a:
  $$cm = m^c mod n$$
  Sendo c e n os valores da chave pública.

- **Descriptografia:** Para decifrar cm, é feito:
  $$m = (cm)^d mod n$$

A segurança do RSA é baseada na dificuldade de resolver o problema da fatoração de números inteiros grandes em seus fatores primos. Até o momento, não existe um algoritmo eficiente conhecido que possa fazer isso em tempo polinomial[1].

Sendo assim, mesmo que o RSA atualmente seja seguro, cada vez mais protocolos estão abandonando a sua criptografia em favor do Diffie-Hellman de Curvas Elípticas (ECDH), outro algoritmo assimétrico. Algumas áreas como BIG DATA e Blockchains podem se beneficiar do ECDH sob o RSA, devido ao fato que o ECDH fornece chaves públicas mais curtas e, benefícios de forma geral, como padrões mais modernos, performance mais rápida e implementações muito mais seguras[17][18].

2.2.3 Troca de Chaves Diffie–Hellman (DH)

Um dos desafios críticos na criptografia assimétrica é a troca segura de chaves entre duas entidades que desejam comunicar-se de maneira privada. O Protocolo de Troca de Chaves de

Diffie-Hellman[19], é uma solução para esse problema, neste protocolo, duas partes podem concordar sobre uma chave secreta compartilhada sem trocar a chave diretamente pela rede. Ele funciona com base no problema dos logaritmos discretos, que é considerado computacionalmente difícil de quebrar por causa da natureza exponencial dele e da falta de um algoritmo eficiente conhecido para resolvê-lo em tempo polinomial. O processo geralmente ocorre da seguinte maneira[19]:

1. Primeiro, antes de qualquer comunicação, são escolhidos e divulgados publicamente dois números primos, p e g, onde p deve ser um número primo grande e g é um gerador para o grupo multiplicativo modulo p.

2. Segundo, cada parte, digamos Alice e Bob, geram uma chave privada (a para Alice e b para Bob). Essas chaves privadas são mantidas em segredo.

3. Em terceiro, ambas as partes calculam suas chaves públicas e as trocam publicamente, onde Alice calcula $A = g^a mod p$ e Bob calcula $B = g^b mod p$.

4. Por ultimo, com as chaves públicas recebidas, cada parte pode calcular a mesma chave compartilhada. Para Alice, a chave compartilhada K seria $K = B^a mod p$, e para Bob, seria $K = A^b mod p$. Devido às propriedades matemáticas envolvidas, esses cálculos resultam na mesma chave para ambos.

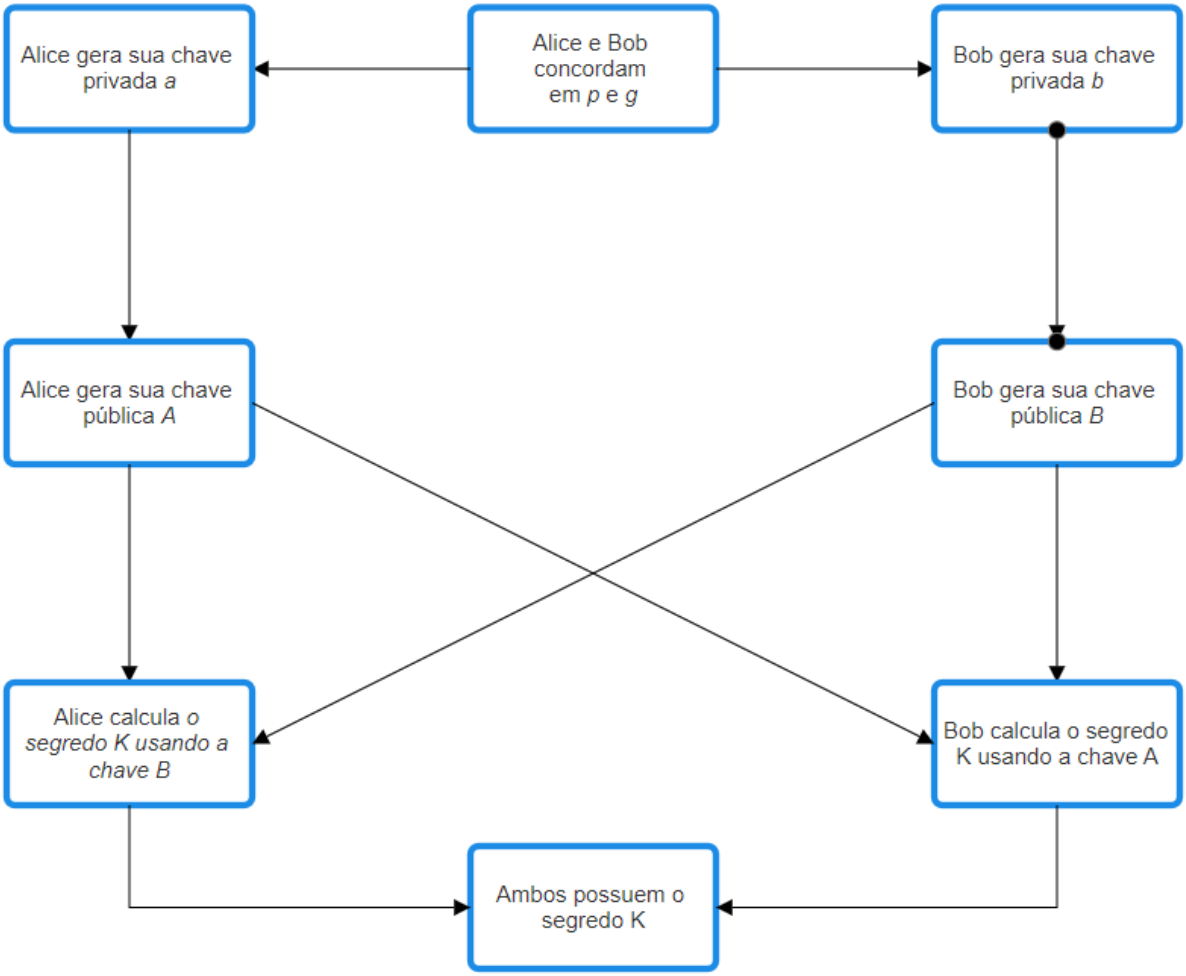

Figura 2.4: Diagrama para o Diffie-Helmann.

O método Diffie-Hellman permite que duas partes estabeleçam uma chave compartilhada mesmo que um atacante tenha conhecimento das chaves públicas trocadas. A chave compartilhada permanece segura, pois ela só pode ser calculada com a posse da chave privada correspondente[19].

Devido ao estudo dos melhores ataques sob o problema do logaritmo discreto, foi possível obter uma ideia de qual valores de p e g podem ser usados com segurança. As melhores práticas conhecidas atualmente são usar um número primo p de 2.048 bits e um gerador g de tamanho 2[20].

O ataque principal sob o DH que ajudou na definição desses valores foi o Logjam [21] em 2015, neste ataque, cientistas se aproveitaram de vulnerabilidades em DHs de 512 bits, que na época era amplamente utilizados na internet, e atacaram uma variedade de serviços. De acordo com [21], Esse ataque custou milhares de CPUs calculando por 1 semana, mas no fim foi provado a necessidade de valores altos para os parâmetros do DH. Os cientistas também estimaram que custaria em torno de 100 milhões de doláres para quebrar a versão de 1024 bits do DH, um valor baixo para órgãos de inteligências como a NSA (National Security Agency)

dos Estados Unidos. Por causa de tudo isso, os autores do Logjam recomendam o ECDH para substituir o DH, mas caso ele não possa ser usado, é recomendado o valor de 2048 bits devido a sua maior robustez[21].

2.2.4 Diffie-Hellman de Curva Elíptica (ECDH)

A abordagem tradicional do Diffie-Hellman é baseada em logaritmos discretos, onde o cálculo da chave compartilhada envolve exponenciação modular. Enquanto essa abordagem ainda é segura, segundo a referência [2], ela pode ser computacionalmente intensiva, especialmente com tamanhos de chave maiores. Por outro lado, o protocolo Diffie-Hellman de Curva Elíptica (ECDH)[22] utiliza propriedades matemáticas específicas de curvas elípticas, que dependendo da curva escolhida, pode alcançar um nível de segurança maior e serem mais resistentes a pré-computação com tamanhos de chave consideravelmente menores, como 256 bits em comparação com 2048 bits do RSA ou DH, isso melhora o desempenho[2].

As curvas elípticas são representações geométricas de equações matemáticas que possuem propriedades únicas e são amplamente utilizadas na criptografia. Essas curvas são definidas sobre corpos finitos, como campos primos, e sua forma geral é dada por uma equação elíptica[23]. Infelizmente, as curvas para o ECDH mais amplamente suportadas, especificadas pela NIST, são vistas com suspeita devido à influência da NSA em seu design, apesar de não haver fraquezas conhecidas. A NSA na época se voluntariou para "contribuir"em certas curvas, que depois foram publicadas e usadas em várias aplicações. Um exemplo da manipulação da NSA em algoritmos padronizados pela NIST foi o caso do Dual_EC _DRBG, um gerador de números pseudo-aleatórios que utiliza a curva P-256 que, de acordo com documentos vazados por Edward Snowden, contém uma vulnerabilidade proposital para benefício da NSA[24].

Tendo isso, outras curvas, como a Curve25519 e a Curve448, foram padronizadas pela IRTF (Internet Research Task Force) e atualmente são usadas em protocolos da Internet[21].

Segundo [2], um exemplo geral de uso da ECDH seria o seguinte:

1. **Geração de Par de Chaves ECDH:** Cada entidade, acorda em uma curva elíptica, um número primo p, que atuará como um corpo finito e um gerador g, que é um ponto na curva geralmente chamado de ponto base. A partir disso, cada um gera uma chave privada única pertencente à curva elíptica, a partir de sua chave privada, cada entidade calcula sua chave pública usando operações específicas da curva.

2. **Troca de Chaves Públicas:** As entidades trocam suas chaves públicas de forma aberta.

3. **Cálculo da Chave Compartilhada:** A chave compartilhada é derivada matematicamente pelo algoritmo usando a chave privada da parte que a possui e a chave pública da outra parte.

Uma das curvas mais populares do ECDH é a Curve25519[25], ou X25519, ela é uma curva de Montgomery e oferece aproximadamente 128 bits de segurança. É uma das curvas mais rápidas e nunca foi patenteada, sua implementação é pública. Ela é regida pela seguinte fórmula[25]:

$$y^2 = x^3 + 486662x^2 + xmod\,p, \; onde \; p = 2^{255} - 19$$

Nessa equação, p é o corpo finito, o número primo que dá nome a curva. O ponto base, ou g, dessa curva é[25]:

$$g = (9,14781619447589544791020593568409986887264606134616475288964881837\ 755586237401)$$

E por fim, essa curva tem uma ordem n, que significa o número de pontos na curva, de[25]:

$$n = 2^{252} + 27742317777372353535851937790883648493$$

2.2.5 Função de Derivação de Chave (KDF)

As Funções de Derivação de Chave (KDF) desempenham um papel fundamental na segurança de sistemas criptográficos. Segundo a referência [26], Elas servem para gerar chaves para o processo de criptografia a partir de chaves mais fracas, como uma senha ou código. As chaves geradas por KDFs são mais seguras e robustas, devido ao processo de criptografia e embaralhamento das funções. Ter uma chave complexa é essencial contra ataques criptográficos, como força bruta, criptoanálise e ataques de dicionário. Uma chave forte e adequadamente protegida dificulta significativamente os esforços de um atacante para comprometer a segurança do sistema[26].

De acordo com a NIST[26], as KDFs tem alguns princípios básicos:

1. **Entrada Inicial (Seed):** As KDFs recebem uma entrada inicial, frequentemente chamada de "seed"ou "contexto inicial". Essa entrada pode incluir senhas, códigos compartilhados ou outros dados.

2. **Parâmetros:** KDFs incluem vários tipos de parâmetros como iterações, "salting"e parâmetros de uso de CPU e memória. Iterações referem-se ao número de vezes que o algoritmo é aplicado, aumentando a complexidade computacional. *Salting* envolve a adição de dados aleatórios à entrada, evitando ataques de dicionário e melhorando a segurança. Já parâmetros de CPU/Memoria ajudam contra ataques de exaustão.

3. **Saída Derivada:** A KDF gera uma saída derivada, que é uma chave ou conjunto de chaves criptograficamente seguras. Essas chaves derivadas são então utilizadas em algoritmos criptográficos para diversos propósitos, como cifragem, autenticação e geração de chaves temporárias.

Uma KDF popular é a PBKDF2 (Password-Based Key Derivation Function 2)[27], ela é amplamente utilizada para derivar chaves de forma segura a partir de senhas em várias aplicações criptográficas, incluindo o hashing de senhas para armazenamento seguro em bancos de dados, derivação para chaves de criptografia e protocolos de acordo de chaves. No entanto, é importante escolher parâmetros apropriados, especialmente o número de iterações, para equilibrar considerações de segurança e desempenho[27].

Tendo isso, segundo [28], a PBKDF2 é vulnerável a ataques de força-bruta, devido a sua simplicidade de arquitetura e leveza computacional. Hoje em dia, é recomendado o uso de KDFs como Argon2 e Scrypt , ambas resistentes a ataques de exaustão[29].

Scrypt

O algoritmo Scrypt[28] é uma função moderna de derivação de chave (KDF) projetada para ser *memory-hard*, o que significa que requer uma quantidade significativa de memória para ser executado, tornando-o resistente a ataques de força bruta e ASICs (Application-Specific Integrated Circuit)[30], que são circuitos customizados, efetivamente projetados para algum propósito específico, nesse caso ataques criptográficos. O Scrypt possui os seguintes parâmetros[31]:

- Passphrase (P): O *passphrase*, também conhecido como senha, é a entrada fornecida pelo usuário. É uma sequência de caracteres escolhida pelo usuário para proteger seus dados, como uma palavra-chave ou código.
- Salt: O *salt* é uma sequência de bytes aleatórios que é misturada com o *passphrase* antes de ser usada como entrada para o algoritmo. O objetivo do *salt* é aumentar a entropia da entrada e tornar cada execução do Scrypt única, mesmo para o mesmo *passphrase*. Isso ajuda a prevenir ataques de dicionário e torna mais difícil a pré-computação de tabelas de hash.
- BlockSize (r): O parâmetro r, também conhecido como *blockSize*, especifica o tamanho do bloco de operações dentro do algoritmo Scrypt. Ele influencia diretamente a quantidade de memória necessária para executar o algoritmo e, portanto, afeta sua resistência a ataques de força bruta.
- CPU/Memory Cost Parameter (N): O parâmetro N, também conhecido como *costParameter*, determina a quantidade de tempo e recursos necessários para executar o algoritmo. Quanto maior o valor de N, mais intensiva em recursos será a computação e mais difícil será para um atacante realizar um ataque de força bruta.

- Parallelization Parameter (p): O parâmetro p, também conhecido como *parallelizationParameter*, especifica o número de iterações que podem ser executadas simultaneamente. Isso afeta a quantidade de trabalho paralelo que pode ser realizado durante a execução do algoritmo, um valor maior de p é útil para a execução em hardware paralelo, como CPUs com vários núcleos.

- Intended Output Length (dkLen): O parâmetro dkLen, também conhecido como *keyLength*, especifica o tamanho em bytes da saída da função, no caso, a chave derivada produzida pelo Scrypt, varia de acordo com a necessidade do usuário.

De acordo com a referência [32], os valores dos parâmetros recomendados atualmente para geração de chaves para criptografia de dados são: *Salt* de 16 bytes, *blockSize* tamanho 8, *costParameter* tamanho 2^{20} e *parallelizationParameter* tamanho 1. Com o tempo, estima-se que esses valores aumentem para acompanhar a sempre crescente capacidade de processamento e memória computacional[32].

2.2.6 Hashing

O hashing é uma técnica fundamental em criptografia que converte dados de tamanho variável em uma sequência de tamanho fixo, conhecida como hash, de forma determinística e eficiente. Essa técnica é amplamente utilizada em diversas aplicações de segurança de dados, incluindo verificação de integridade, autenticação de informações e armazenamento seguro de senhas[2]. Os princípios básicos de hashing são[33]:

1. **Função de Hash:** Uma função de hash é responsável por transformar os dados de entrada em uma sequência de bits de tamanho fixo, conhecida como hash ou resumo. Essa transformação é unidirecional, ou seja, não é possível recuperar os dados originais a partir do hash.

2. **Propriedades do Hashing:** As funções de hash devem possuir algumas propriedades fundamentais, como:

    - Determinismo: Dados idênticos sempre produzem o mesmo hash.
    - Unicidade: Diferentes conjuntos de dados devem produzir hashes distintos.
    - Eficiência: O cálculo do hash deve ser rápido e eficiente.
    - Resistência a Colisões: Deve ser computacionalmente inviável encontrar dois conjuntos de dados diferentes que produzam o mesmo hash.
    - Resistência a Pré-Imagem e Segunda Pré-Imagem: Deve ser computacionalmente inviável recuperar os dados originais a partir do hash (pré-imagem), bem como encontrar outro conjunto de dados que produza o mesmo hash (segunda pré-imagem).

De acordo com [34], hashing é frequentemente utilizado em vários contextos, por exemplo, verificar a integridade dos dados durante a transmissão ou armazenamento, comparando hashes antes e após a transferência de dados, é possível detectar qualquer alteração não autorizada nos dados, hashing também é utilizado em sistemas de autenticação para armazenar senhas como hashes em vez de texto simples, isso ajuda a proteger as senhas contra vazamentos de dados e ataques de força bruta, já que os hashes não podem ser revertidos para recuperar a senha original.

Alguns algoritmos populares de hashing são o MD5 e o SHA-1 (Secure Hash Algorithm 1), ambos atualmente são considerados inseguros devido a sua falta de resistência a colisões[35][36], mas ainda são usados em sistemas desatualizados devido ao custo de infraestrutura necessário para troca-los para um novo algoritmo, como por exemplo, o SHA-2[37], que foi criado para substituir o SHA-1 sem ter os problemas de colisão. O SHA-2 é baseado na arquitetura de Merkle–Damgård[37], que utiliza uma função de compressão e várias iterações para fazer o hashing. Mesmo sendo mais seguro que seu predecessor, o SHA-2 é vulnerável para outro tipo de ataque, o lenght-extension attack, no qual o atacante intercepta e adiciona dados ao hash, que causa problemas de checagem de integridade e autenticidade. Tal ataque levou a criação de um novo algoritmo, o SHA-3[2].

SHA-3

Embora o SHA-2 seja uma função de hash perfeitamente adequada para vários usos, segundo [2], ela não é adequado para hashing de segredos, como chaves criptográficas. Devido a isso, o algoritmo recomendado atualmente é o SHA-3[38], que tem alta segurança contra todos tipos de ataques contra hashes. Ele oferece as seguintes variantes: SHA-3-224, SHA-3-256, SHA-3-384 e SHA-3-512, o número no fim representa o tamanho da saída do algoritmo em bits, o que também muda a quantidade de dados processados por entrada e a performance e segurança de cada variação algoritmo, ficando ao usuário a escolha dependendo de sua necessidade.

O SHA-3 usa uma arquitetura diferente da Merkle–Damgård, ele usa a arquitetura de esponja, que foi inventada como parte da competição SHA-3, um torneio para decidir o sucessor do SHA-2. Esse método de esponja é baseado em uma permutação específica chamada *Keccak-f* que recebe uma entrada e retorna uma saída do mesmo tamanho[39].

A permutação Keccak-f opera em uma matriz de estado de tamanho 1600 bits, inicialmente zerados, que vão ser gradualmente preenchidos com o resultado de operações de XOR com blocos da entrada a ser hasheada, e depois passados pela permutação de Keccak-f. Esse processo se repete várias vezes dependendo da versão do SHA-3, no fim, extraímos da matriz os dados em hash[2].

Na figura abaixo, temos o processo de permutação do SHA-3, na esquerda temos a matriz inicial e na direita a hash, no topo temos a entrada sendo usada com operações XOR e sendo permutada em *f*.

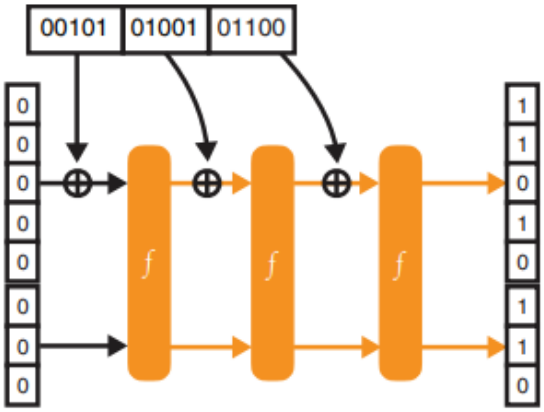

Figura 2.5: Permutação Keccak-f, fonte: [2]

2.3 Esteganografia

Segundo a referência [40], a esteganografia é uma disciplina antiga com raízes profundas na história da comunicação secreta, que emergiu como uma área de estudo significativa no contexto da era digital. Diferentemente da criptografia, que se concentra na codificação de mensagens para proteger seu conteúdo contra acesso não autorizado, a esteganografia busca ocultar a própria existência da comunicação secreta, integrando-a de forma imperceptível em outros tipos de dados.

Desde tempos antigos, a esteganografia tem sido empregada como uma ferramenta poderosa para comunicação secreta. Um dos exemplos mais notáveis remonta ao século V a.C., quando Heródoto descreveu como mensagens foram tatuadas na cabeça de mensageiros e, em seguida, deixadas para crescer o cabelo de volta antes de serem enviadas. Essa prática histórica ilustra o impulso humano de ocultar informações de olhos curiosos[40].

Já atualmente, no meio digital, a esteganografia é aplicada em uma variedade de cenários, desde comunicações seguras até proteção de direitos autorais e marca d'água digital. Em ambientes forenses, a esteganografia é uma ferramenta valiosa para investigar e detectar comunicações secretas, enquanto em campos como inteligência e segurança nacional, pode ser usada para transmitir informações sensíveis de forma discreta[41].

Tipos de Esteganografia

Com o advento da era digital, a esteganografia evoluiu para explorar técnicas inovadoras de ocultação de dados em arquivos digitais, como imagens, áudio, vídeo e texto. De acordo com [42], essas técnicas podem ser especificadas da seguinte maneira:

- **Esteganografia em Texto:**

  Na esteganografia em texto, as mensagens são ocultas dentro de documentos de texto ou conteúdo textual. Isso pode ser alcançado através de técnicas como a utilização de

espaços extras, modificação de formatação ou até mesmo a substituição de palavras por sinônimos ou letras específicas. A esteganografia em texto é frequentemente utilizada em comunicações online e em documentos digitais para transmitir informações de forma discreta.

- **Esteganografia em Áudio:**

  A esteganografia em áudio envolve a incorporação de dados ocultos em arquivos de áudio, como músicas ou gravações de voz. Técnicas comuns incluem a modificação de amplitudes, frequências ou fase do sinal de áudio para ocultar informações. A esteganografia em áudio é frequentemente utilizada em aplicações de marca d'água digital, proteção de direitos autorais e transmissão de mensagens secretas através de mensagens de áudio ou chamadas de voz.

- **Esteganografia em Vídeo:**

  A esteganografia em vídeo envolve a ocultação de dados dentro de arquivos de vídeo, como filmes ou clipes. Isso pode ser realizado através de técnicas semelhantes às usadas na esteganografia em imagem, como a modificação de frames individuais, a alteração de valores de pixels ou a inserção de dados em áreas específicas do vídeo. A esteganografia em vídeo é comumente utilizada em sistemas de vigilância, transmissões ao vivo e proteção de conteúdo de vídeo contra cópias não autorizadas.

2.3.1 Esteganografia em Imagens

A esteganografia em imagens, segundo [41], é uma técnica sofisticada que envolve a ocultação de mensagens secretas dentro de arquivos de imagem, como fotografias digitais, ilustrações ou gráficos. Este método aproveita as propriedades dos pixels e do domínio da imagem para integrar dados ocultos de forma imperceptível, mantendo a aparência visual da imagem original.

Pixels, são os pontos individuais que compõem uma imagem digital. Em termos simples, um pixel é a menor unidade de uma imagem digital, representando uma cor específica em uma grade. Cada pixel contém informações sobre a cor e intensidade luminosa que deve ser exibida naquela posição específica da imagem. As imagens digitais são compostas por uma matriz de pixels organizados em linhas e colunas. Quanto maior a resolução da imagem, mais pixels estão presentes, proporcionando uma representação mais detalhada e nítida[43].

A cor de um pixel é geralmente representada usando modelos de cor como o RGB (Red, Green, Blue), onde a combinação dessas três cores primárias em diferentes intensidades cria uma ampla gama de cores. A intensidade luminosa ou opacidade do pixel pode ser representada usando diferentes sistemas, como o RGBA (Red, Green, Blue, Alpha), onde o componente "Alpha"controla a transparência do pixel[44].

Os sistemas RGB e RGBA atuam com valores de 0 a 255, ou 1 byte, para cada cor primária e para a o valor de transparência da imagem, por exemplo, um pixel com valor RGB de (255,0,0)

vai ser totalmente vermelho, já outro com (255,255,0) vai ser totalmente amarelo, uma mistura de vermelho com verde. Tendo isso, um pixel de RGB vai ter um tamanho de 3 bytes e um RGBA tamanho 4, ou seja, uma imagem RGB de resolução 1200x1200 vai ter um total de 1.440.000 pixels, e com isso um tamanho de 4.320.000 bytes ou 4.32 Mb[45].

Técnicas de Esteganografia em Imagem

Existem várias técnicas usadas na esteganografia de imagens, cada uma com suas próprias vantagens e desvantagens. Aqui está uma visão geral de algumas técnicas comuns.

- **Técnicas no Domínio do Espaço:**

  Segundo a referência [41], essas técnicas atuam no domínio espacial onde operam diretamente nos valores de pixel da imagem, ou seja, escondem dados modificando levemente as cores de cada pixel, algo imperceptível ao olho humano mas que pode ser detectado por certos algoritmos, dependendo da técnica de modificação espacial escolhida.

  Algumas técnicas do domínio espacial são o Least Significant Bit (LSB)[46] e o Random Pixel Embedding (RPE)[47]. No LSB, a mensagem a ser escondida é transforma em uma sequencia binária e codificada nos bits na imagem, por exemplo, caso a letra "A"precise ser escondida ela primeiro é transformada no seu equivalente binário ASCII, no caso, 01000001. Como todo símbolo na tabela ASCII, "A"tem valor de 8 bits, necessitando 3 pixels RGB para oculta-lo pois cada pixel tem 3 valores[46]. Como exemplo, digamos que temos 3 pixels:

  (121, 20, 234), (239, 55, 149), (12, 40, 182)

  Para esconder a sequência 01000001, o algoritmo vai navegar a sequência e trocar o valor de cada número dos pixels ao depender se temos um 1 ou um 0, caso seja 1 o valor deve ser impar e caso seja 0 o valor deve ser par, então no fim os 3 pixels modificados ficam assim:

  (122, 21, 234), (240, 56, 150), (12, 41, 182)

  O último valor dos 3 pixels serve para dizer quando a mensagem escondida acaba, caso a mensagem continue o ultimo valor deve ser par, caso ela acabe o último valor deve ser impar.

  Para decodificar, o programa navega os pixels da imagem lendo se os valores são par ou impar, o que vai decidir se ele encontrou um 0 ou um 1 , e quando chegarmos num valor impar no fim de uma tripla de pixels significa que a mensagem acabou.

  O LSB é vulnerável a detecção devido a sua simplicidade ao modificar os pixels, como cada pixel é mudado em sequência isso pode ser descoberto por uma análise de padrões. Por outro lado, o algoritmo RPE modifica os pixels de maneira aleatória, ele funciona da mesma maneira que o LSB exceto na parte de escolher quais pixels vão ser alterados, no

RPE isso é feito baseado num padrão aleatório o que protege mais a imagem codificada contra ataques de análise[47] .

- **Técnicas no Domínio Transformado:**

  De acordo com a referência [41], as técnicas de domínio de transformação envolvem a conversão da imagem de sua representação no domínio espacial para um domínio diferente, como o domínio de frequência ou de *wavelet*, usando transformações matemáticas. Nestes domínios, os dados da imagem são representados de maneiras variadas dependendo do tipo do domínio, que então podem ser manipuladas para incorporar informações ocultas.

  O processo de incorporar dados em diferentes domínios de uma imagem é mais seguro, mas mais complexo, do que os princípios de incorporação que operam no domínio espacial. A maioria dos sistemas esteganográficos robustos hoje opera dentro do domínio de transformação. As técnicas desse domínio têm uma vantagem sobre as técnicas de domínio espacial, pois escondem informações em áreas da imagem que estão menos expostas à compressão, *cropping* e processamento de imagem[42].

  Uma técnica de domínio transformado popular é a Discrete Cosine Transform (DCT)[42]. Na DCT, a imagem é convertida para seu domínio de frequência. O domínio de frequência de uma imagem se refere a uma representação da imagem onde as variações de intensidade em toda a imagem são expressas em termos de suas frequências, em vez de suas localizações, em outras palavras, ele descreve como a intensidade da imagem varia entre diferentes posições espaciais. Nesse domínio, as baixas frequências representam variações suaves na intensidade, enquanto as altas frequências representam mudanças rápidas ou detalhes na imagem[48].

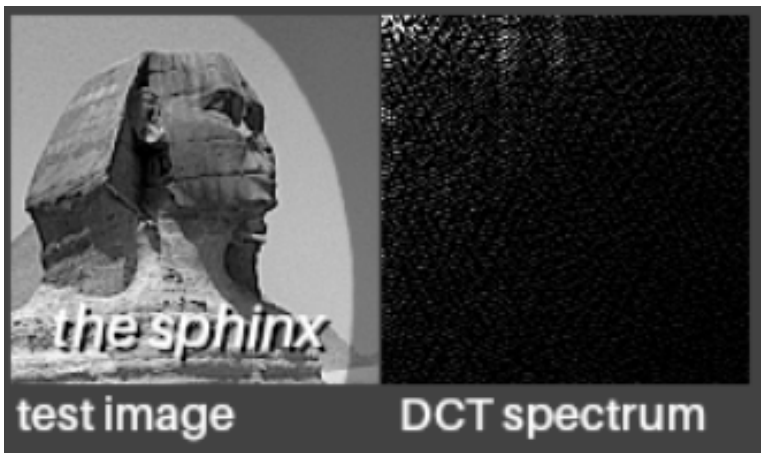

Figura 2.6: Imagem no domínio espacial na esquerda e sua versão no domínio de frequência por DCT. Imagem retirada de: [3]

Tendo isso, na esteganografia de DCT os coeficientes resultantes da imagem são modificados para incorporar a mensagem secreta. Tipicamente, segundo [42], os coeficientes

menos significativos (aqueles que representam componentes de alta frequência) são selecionados para incorporação para minimizar a distorção visual. Depois que a mensagem é incorporada no domínio da frequência da imagem, ela é transformada de volta para o domínio espacial e está pronta a imagem codificada.

2.4 Tipos de Ataques Criptográficos

Os ataques criptográficos visam comprometer a segurança dos sistemas de criptografia, seja revelando informações confidenciais, comprometendo a integridade dos dados ou quebrando algoritmos de criptografia. Aqui estão alguns tipos comuns de ataques criptográficos[10]:

- **Ataque de Força Bruta:**

  Também chamados de ataques de exaustão, eles envolvem testar todas as possíveis combinações de chaves para quebrar uma criptografia. Esse método é geralmente usado para quebrar senhas ou chaves de criptografia de forma sistemática e geralmente envolvem o uso de programas automatizados que testam uma grande quantidade de combinações de caracteres em rápida sucessão até encontrar a senha correta. Esse ataque é bastante simples, porém demorado e pode ser eficaz contra chaves fracas ou mal protegidas.

- **Ataque de Dicionário:**

  Nesse tipo de ataque, o invasor utiliza uma lista de palavras comuns, senhas comuns ou chaves predefinidas para tentar encontrar a chave correta. É uma variação mais eficiente do ataque de força bruta, pois se concentra em testar combinações mais prováveis primeiro.

- **Ataque de Criptoanálise:**

  A criptoanálise envolve o estudo dos sistemas criptográficos para encontrar fraquezas ou padrões que possam ser explorados para quebrar a criptografia de forma mais eficiente do que ataques de força bruta. Existem várias técnicas de criptoanálise, incluindo ataques de texto simples conhecido, ataques de texto cifrado conhecido e ataques de escolha de texto.

- **Ataque de Texto Simples Conhecido:**

  É um tipo de ataque criptográfico onde o invasor possui acesso a pares de texto plano (mensagem original) e texto cifrado (mensagem criptografada) correspondentes. Com esses pares conhecidos, o invasor tenta deduzir informações sobre a chave secreta ou o algoritmo de criptografia.

- **Ataque de Texto Cifrado Conhecido:**

Nesse tipo de ataque, o invasor só possui acesso ao texto cifrado. O objetivo é deduzir informações sobre o texto plano original ou a chave de criptografia. Esse tipo de ataque é geralmente considerado o mais difícil de realizar, pois o invasor não tem acesso direto ao texto plano.

- **Ataque de Escolha de Texto:**

  Nesses tipos de ataques, o invasor pode escolher um texto plano e observar a correspondente saída criptografada, ou um texto cifrado e obter o texto plano correspondente, com isso ele tenta obter informações sobre a chave secreta ou o algoritmo de criptografia. Esses ataques são mais poderosos do que ataques de texto simples conhecido e texto cifrado conhecido.

3

Ferramenta

3.1 Introdução a Ferramenta

Apresentamos a ferramenta que exemplifica a combinação de técnicas de criptografia e esteganografia para ocultar, recuperar e detectar erros de mensagens em arquivos de imagem. Embora seja uma versão demonstrativa sem opções avançadas de customização ou interface completa, a ferramenta oferece uma abordagem didática para compreensão dos princípios fundamentais envolvidos em segurança de comunicações digitais.

O propósito principal desta ferramenta é proporcionar uma visão prática dos conceitos de esteganografia e criptografia em processo de ocultação de informação utilizando arquivos de imagem.

3.2 Especificações da Ferramenta

A ferramenta demonstrativa foi desenvolvida utilizando a linguagem de programação Python[49], devido a sua alta flexibilidade e a vasta gama de bibliotecas disponíveis na comunidade de desenvolvedores. Para atender aos requisitos específicos de criptografia, esteganografia e manipulação de imagens, utilizei as seguintes bibliotecas:

- **PyCryptodome:** É uma biblioteca de criptografia que oferece uma ampla variedade de algoritmos criptográficos. Utilizamos esta biblioteca para implementar os algoritmos de criptografia necessários para proteger as mensagens antes de ocultá-las nas imagens[50].
- **PySimpleGUI:** É uma biblioteca Python que simplifica a criação de interfaces gráficas do usuário (GUI) de forma intuitiva e rápida. Utilizamos o PySimpleGUI para criar uma interface básica que permite o usuário interagir com a ferramenta de forma simples e direta[51].

- **Pillow:** Pillow é uma biblioteca de processamento de imagem em Python que oferece uma ampla gama de funcionalidades para manipulação de imagens, como abrir, salvar, modificar e converter diversos formatos de imagem. A ferramente utiliza o Pillow para carregar as imagens de entrada, modificar os pixels conforme necessário e salvar as imagens resultantes após a ocultação ou extração das mensagens[52].

- **Base64:** A Base64 é uma biblioteca nativa do Python e fornece funções para codificação e decodificação de dados usando o esquema de codificação Base64. A ferramenta utiliza a codificação Base64 para converter as mensagens criptografadas para bits para ocultação em arquivos de imagem e, posteriormente, para decodificar os bits extraídos das imagens[49].

No âmbito de algoritmos criptográficos, foram implementados os seguintes métodos no programa:

- **AES:** É utilizado na criptografia simétrica do programa, é empregado para criptografar as mensagens antes de serem ocultadas nas imagens. A escolha do AES foi devido a sua alta segurança e eficiência.

- **ECDH:** É utilizado na criptografia assimétrica do programa, é empregado para gerar uma chave compartilhada que poderá ser usada como uma chave simétrica para o AES. Utilizamos o ECDH na ferramenta devido a sua superioridade comparada ao DH básico e seu melhor desempenho.

- **SHA-3-256:** O SHA-3 é utilizado para hashing da mensagem antes de ser criptografada, na ferramenta, o SHA-3 é utilizado para verificar a integridade das mensagens, garantindo que não tenham sido alteradas ou corrompidas durante a transmissão ou o armazenamento. Ele foi escolhido diante de outros algoritmos devido a sua superioridade e a versão de 256 bits foi escolhida por ser uma boa balança entre segurança e desempenho.

- **Scrypt:** Na ferramenta, o Scrypt é empregado para gerar chaves criptográficas a partir de senhas ou frases-chave fornecidas pelos usuários, garantindo uma proteção adicional contra ataques de força bruta. Ele foi selecionado devido a sua modernidade e alta defesa contra ataques.

Já na área de esteganografia, foi implementado o algoritmo RPE que se baseia na aleatoriedade para ocultar os dados dentro de imagens. O RPE utiliza uma semente aleatória, baseada na chave criptográfica, para determinar quais pixels serão modificados, tornando a detecção da mensagem oculta mais difícil. Ele foi escolhido pois é uma escolha entre segurança e eficiência.

3.3 Uso da Ferramenta

A ferramenta demonstrativa oferece uma interface simples e direta para facilitar o processo de esteganografia e criptografia. Composta por quatro botões distintos, cada um projetado para uma função específica, a ferramenta permite aos usuários realizar as seguintes operações:

1. **Esteganografia de Texto em Imagem:**

   O primeiro botão da ferramenta permite aos usuários ocultarem um texto criptografado em uma imagem, seguindo os seguintes passos:

   - O texto a ser ocultado é cifrado usando o algoritmo AES e a chave fornecida, garantindo sua segurança.

   - O texto cifrado é então ocultado na imagem selecionada, utilizando técnicas de esteganografia.

   - As entradas necessárias para este processo são o texto a ser cifrado, a chave de criptografia, que pode ser uma senha ou uma chave gerada pelo ECDH, a imagem que se deseja esconder os dados e o nome da imagem resultante.

Figura 3.1: Primeira opção da ferramenta.

2. **De-Esteganografia de Imagem:**

   O segundo botão possibilita a extração e decifração do texto oculto em uma imagem, revertendo o processo descrito anteriormente:

   - A imagem contendo o texto oculto é selecionada.

   - O texto oculto é extraído da imagem e decifrado utilizando a chave de criptografia fornecida.

- A entrada necessária para este processo é a imagem contendo o texto oculto e a chave de criptografia.

Figura 3.2: Segunda opção da ferramenta.

3. **Geração de Par de Chaves ECDH:**

   O terceiro botão é responsável por gerar um par de chaves ECDH, permitindo aos usuários iniciar uma troca segura de chaves públicas com outros usuários:

   - Um par de chaves ECDH (chave pública e chave privada) é gerado pela ferramenta.

   - As chaves pública e privada são disponibilizadas ao usuário para compartilhamento seguro com outros participantes da comunicação.

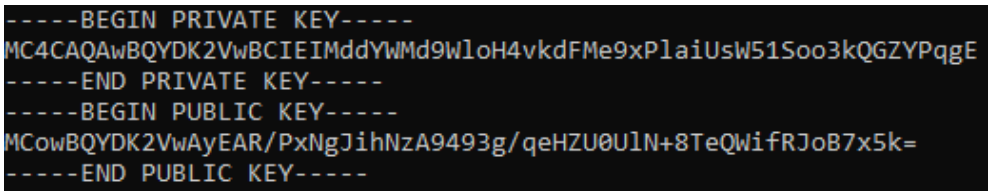

Figura 3.3: Par de chaves ECDH gerado pela ferramenta.

4. **Geração do Segredo Compartilhado:**

   O quarto botão possibilita a geração do segredo compartilhado entre duas partes que trocaram chaves públicas ECDH, permitindo o estabelecimento de uma chave de criptografia comum:

   - A chave pública recebida do outro participante é combinada com a chave privada do usuário para calcular o segredo compartilhado.

   - O segredo compartilhado resultante pode ser usado como chave de criptografia em comunicações futuras.

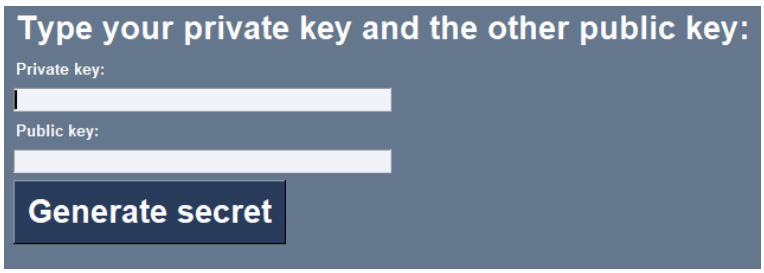

Figura 3.4: Quarta opção da ferramenta.

4

Testes e Discussões

4.1 Introdução

Neste capítulo, conduziremos uma análise do desempenho da esteganografia da ferramenta em diferentes cenários, utilizando três imagens de tamanhos variados como base de teste. O objetivo é avaliar a capacidade da ferramenta em ocultar informações de forma eficaz, mantendo a qualidade visual das imagens e a segurança dos mecanismos de criptografia e esteganografia.

Para isso, utilizaremos imagens de teste geradas por [53] no formato de .png de 512x512, 896x896 e 1256x1256 pixels, sendo assim, uma imagem pequena, média e grande. Nos testes, cada imagem será carregada com mensagens de 5.148, 40.080 e 80.080 bytes, sendo 1 byte por caractere de texto. Esses valores foram escolhidos devido a especificações do programa.

Depois de terem as mensagens ocultadas, as imagens passarão por avaliações dadas por métricas de processamento de imagem. São elas:

1. **MSE (Mean Squared Error):** O MSE é uma métrica que quantifica a média dos erros quadrados entre os valores de intensidade dos pixels da imagem original e da imagem esteganografada. Quanto mais perto de 0 o valor do MSE, mais similar a imagem esteganografada está em relação à original[54].

2. **PSNR (Peak Signal-to-Noise Ratio):** O PSNR é uma medida que compara a qualidade da imagem esteganografada com a imagem original, considerando o nível de ruído introduzido durante o processo de esteganografia. Quanto maior o valor do PSNR, menor o nível de distorção percebida na imagem esteganografada em relação à original. Valores ideais para esteganografia em pixels são acima de 60db[55].

3. **SSIM (Structural Similarity Index):** O SSIM é uma medida de similaridade estrutural entre duas imagens, considerando a luminância, o contraste e a estrutura. Ele quantifica a qualidade perceptual da imagem esteganografada em relação à original, levando em conta

as características visuais e estruturais das imagens. Quanto mais perto de 1, mais similar são as imagens[55].

4. **Payload Capacity:** A capacidade de carga refere-se à quantidade máxima de dados que podem ser ocultados de forma imperceptível em uma imagem esteganografada. No contexto de esteganografia de imagem espacial, a capacidade de carga é medida em bits por pixel (bpp) e depende do tamanho da mensagem ocultada em bits e de quantos pixels tem a imagem que vai oculta-la. Quanto maior a capacidade de carga, mais dados podem ser ocultados na imagem, mas por outro lado, mais vulnerável fica a imagem a ser detectada por sistemas de estegoanálise[56].

4.2 Teste Imagem 512x512

Neste teste, iremos avaliar o desempenho da ferramenta de esteganografia na ocultação de dados em uma imagem colorida de 512x512 pixels. Abaixo temos a imagem na esquerda antes e depois da ocultação de 80.080 bytes, como podemos ver, não existe nenhuma alteração visual perceptível.

Figura 4.1: Antes e depois da imagem de teste 512x512

Agora veremos os resultados das métricas para os 3 possíveis tamanhos de mensagem a ser ocultada:

Ocultado	MSE	PSNR	SSIM	Payload Capacity
5148B	0.0294	63.4471	0.9997	0.1571
40.080B	0.2295	54.5212	0.9986	1.2231
80.080B	0.4584	51.5181	0.9981	2.4438

Tabela 4.1: Tabela de avaliação da esteganografia da imagem 512x512

Pelos resultados, na mensagem ocultada de 5.148 bytes tivemos um MSE baixo e um PSNR alto, o que significa uma imagem esteganográfica robusta[57], o SSIM também foi muito alto o que indica uma alta qualidade estrutural da imagem resultante. A capacidade de carga também foi muito baixa, devido ao pequeno tamanho da mensagem ocultada, um resultado de 0.1571 é muito bom, pois é menor que 0.4 que foi demonstrado como um limite no qual a maioria dos algoritmos de esteganografia em imagem começam a ser detectados por ferramentas de estegoanálise[58].

Nos outros dois casos de mensagem ocultada de 40.080 e 80.080 bytes, podemos observar uma perda na qualidade da esteganografia, os valores do MSE aumentaram para níveis não-ideais e os do PSNR diminuiram para níveis aceitáveis, o SSIM também diminuiu, mas a capacidade de carga aumentou significamente, ultrapassando o limite de 0.4, o que deixa esses casos mais vulneráveis a detecção. Tudo isso se da pelo grande aumento do tamanho da mensagem ocultada, que por sua vez, faz com que mais pixels da imagem sejam modificados para esconde-la, isso acaba afetando a segurança da esteganografia.

4.3 Teste Imagem 896x896

Neste próximo teste, iremos avaliar o desempenho da nossa ferramenta de esteganografia na ocultação de dados em uma imagem colorida de 896x896 pixels. Abaixo temos a imagem na esquerda antes e depois da ocultação de 80.080 bytes, como na imagem anterior, podemos ver que não existe nenhuma alteração visual perceptível.

Figura 4.2: Antes e depois da imagem de teste 896x896

Agora os resultados das métricas para os 3 possíveis tamanhos de mensagem a ser ocultada:

Ocultado	MSE	PSNR	SSIM	Payload Capacity
5148B	0.0096	68.2733	0.99993	0.0512
40.080B	0.0748	59.3908	0.99955	0.3993
80.080B	0.1495	56.3827	0.99916	0,7979

Tabela 4.2: Tabela de avaliação da esteganografia da imagem 896x896

De acordo com os testes, a mensagem ocultada de 5.148 bytes teve um desempenho muito bom, com um MSE muito baixo e ambos PSNR e SSIM bem altos, e também uma capacidade carga muito baixa. Esses resultados demonstram uma esteganografia bastante segura e foram conseguidos através da discrepância entre tamanho da mensagem e tamanho da imagem, quanto maior essa diferença maior a segurança de ocultação da ferramenta.

Nos caso de 40.080 bytes escondidos, a imagem teve resultados beirando o limite da segurança, com um MSE razoavelmente baixo e um PSNR perto de 60db, junto de uma capacidade muito próxima de 0.4bpp, a imagem alcançou uma balança boa balança entre robustez e quantidade de dados escondidos.

Por último, no caso de 80.080 bytes ocultados, temos uma perda da qualidade da esteganografia, com um MSE fraco e um PSNR razoável, com também uma perca no SSMIM e um aumento na capacidade de carga, ultrapassando o limite de segurança. Com isso, podemos concluir que essa última mensagem é muito grande para essa imagem.

4.4 Teste Imagem 1256x1256

Neste último teste, iremos avaliar o desempenho da nossa ferramenta de esteganografia na ocultação de dados em uma imagem colorida de 1256x1256 pixels. Abaixo temos a imagem na esquerda antes e depois da ocultação de 80.080 bytes, como nas outras imagens, podemos ver que não existe nenhuma alteração visual perceptível.

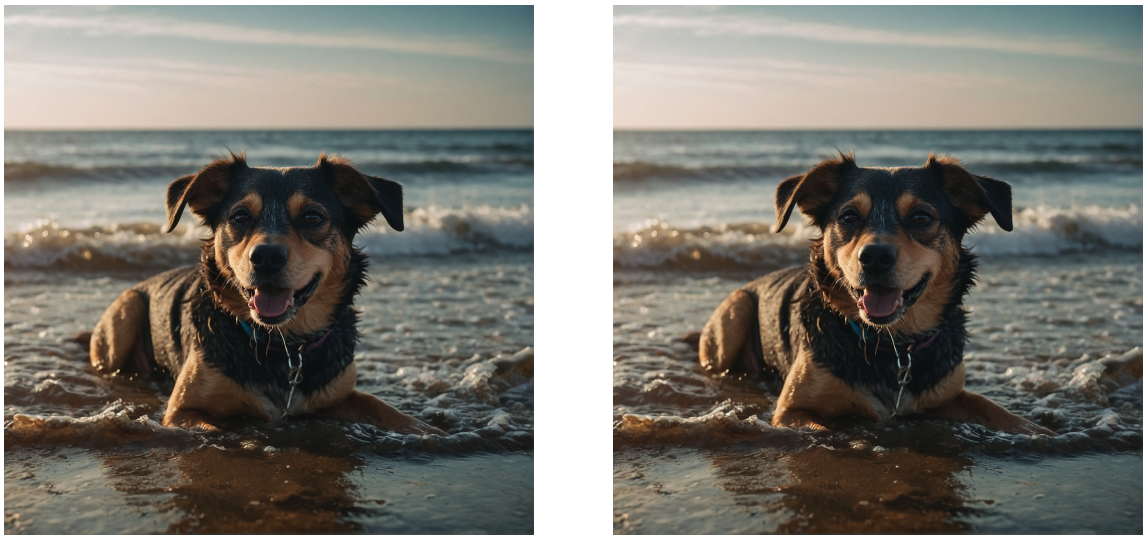

Figura 4.3: Antes e depois da imagem de teste 1256x1256

Agora os resultados das métricas para os 3 possíveis tamanhos de mensagem a ser ocultada:

Ocultado	MSE	PSNR	SSIM	Payload Capacity
5148B	0.0049	71.2140	0.99995	0.0261
40.080B	0.0380	62.3225	0.99963	0.2032
80.080B	0.0760	59.3174	0.99929	0,4061

Tabela 4.3: Tabela de avaliação da esteganografia da imagem 1256x1256

Interpretando os testes, temos que no caso de 5.148 bytes o algoritmo teve um desempenho excepcional, com um MSE bem baixo e um PSNR acima de 70db, um SSIM bem alto e a capacidade muito baixa. Como dito anteriormente, quanto maior a imagem e menor a mensagem a ser escondida, melhor o desempenho da segurança do algoritmo.

No caso de 40.080 bytes, tivemos um resultado muito bom, com um MSE baixo e um PSNR acima de 60db, um SSIM alto e uma capacidade baixa, ao todo, temos uma ótima balança entre segurança e quantidade de ocultamento de dados. Similarmente, no caso de 80.080 bytes, temos uma boa balança de segurança, mas já com valores mais próximos do limite de qualidade, com um PSNR perto dos 60db e uma capacidade um pouco maior que 0.4.

4.5 Considerações Finais

No geral, observamos que a ferramenta apresentou um bom desempenho quando ocultando mensagens de menor tamanho em imagens maiores. Na imagem de 512x512 pixels, a esteganografia foi eficaz na ocultação de mensagens de 5.148 bytes, demonstrando valores métricos bons, indicativos de uma esteganografia robusta e de alta qualidade. No entanto, à medida que o tamanho da mensagem ocultada aumentou para 40.080 e 80.080 bytes, houve uma degradação na qualidade da esteganografia, refletida por um aumento no MSE e na capacidade de carga e uma redução no PSNR e SSIM.

Nas imagens de 896x896 e 1256x1256, vemos uma conclusão similar, de que quanto maior for a mensagem a ser ocultada, para que seja alcançada uma boa segurança, maior deve ser o tamanho da imagem. Tendo isso, ambas imagens, apresentaram resultados bons na ocultação dos dados, a imagem de 896x896 conseguiu ocultar os dados seguramente nos casos de 5.148 e 40.080 bytes, mas falhou na ocultação de 80.080 bytes, e a imagem de 1256x1256 conseguiu esconder os 3 casos com uma boa robustez, tendo um ótimo desempenho em 5.148 e 40.080 bytes.

Podemos concluir que, na ferramenta, a escolha da imagem depende do tamanho dos dados a serem escondidos, caso haja necessidade de transmitir uma grande quantidade de dados é necessário escolher uma imagem grande para garantir a segurança do processo. Por outro lado, caso seja preciso transmitir uma pequena quantidade de dados, pode-se escolher uma imagem menor e mais inconspícua, mas que também vai ter limitações na possível quantidade de carga.

A partir disso, temos uma base sólida para futuras melhorias e otimizações na ferramenta de esteganografia. Estratégias para melhorar a segurança da esteganografia em imagens menores, como a implementação de técnicas de compressão e codificação adaptativa, podem ser exploradas para equilibrar a capacidade de carga da imagem e a qualidade da esteganografia.

No entanto, é importante ressaltar que, embora os resultados desses testes forneçam um entendimento da capacidade da ferramenta, há outros fatores a serem considerados, como diferentes ambientes e algoritmos de detecção de esteganografia. Portanto, testes adicionais e validações práticas são necessários para uma avaliação completa do desempenho e da segurança da ferramenta em cenários do mundo real.

Por fim, mesmo que um atacante consiga descobrir que uma imagem possui conteúdo esteganográfico, ele ainda precisa extrair esse conteúdo e decifra-lo caso deseje acessar seu conteúdo, isso só será possível através da criptoanálise, que não será abordada nesse trabalho.

5

Conclusão

Ao longo deste trabalho, exploramos os fundamentos teóricos da esteganografia e da criptografia, abordando conceitos-chave relacionados à ambos num ambiente de transmissão de dados. Aprofundamos nosso entendimento sobre protocolos de comunicação, algoritmos criptográficos e técnicas de ocultação de dados esteganográficas, estabelecendo uma base sólida para o desenvolvimento de nossa ferramenta de ocultação.

Na seção dedicada à ferramenta, detalhamos sua arquitetura e funcionalidades, demonstrando como ela foi implementada para oferecer uma demonstração na ocultação segura de dados em imagens. Foi descrito os algoritmos criptográficos utilizados, bem como as técnicas de esteganografia empregadas, fornecendo uma visão de seu funcionamento e usabilidade.

Os testes e discussões realizados com a ferramenta proporcionaram um entendimento sobre seu desempenho e eficácia na ocultação de dados em imagens de diferentes tamanhos. Ao avaliar métricas como MSE, PSNR, SSIM e capacidade de carga, pudemos identificar padrões e tendências que contribuíram para uma compreensão mais profunda do impacto da esteganografia na qualidade das imagens e na segurança dos dados ocultados.

Em resumo, este trabalho estuda conceitos atuais de criptografia e esteganografia, fornecendo uma ferramenta prática para a ocultação segura de dados. Por meio da combinação de conceitos teóricos sólidos, e do desenvolvimento da ferramenta e seus testes, conseguimos oferecer um conhecimento aprofundado no âmbito de proteção da confidencialidade dos dados e garantia da integridade da comunicação digital.

No entanto, reconhecemos que há espaço para melhorias e refinamentos contínuos na ferramenta. No futuros podemos explorar a implementação de técnicas mais avançadas, análise de vulnerabilidades e integração com outros sistemas de segurança digital para fornecer uma solução mais forte e abrangente.

Referências bibliográficas

[1] James Kurose and Keith Ross. *Computer Networking- A Top Down Approach-Pearson*. Pearson Education Limited, 2020. ISBN 978-1292405469.

[2] David Wong. *Real-World Cryptography*. Manning, 2021. ISBN 978-1617296710.

[3] Drummyfish. Comparison of diference dct filter effects, 2019. URL https://commons.wikimedia.org/wiki/File:DCT_filter_comparison.png.

[4] MIT Technology Review. Data literacy: a importância da alfabetização em dados em um mundo big data. 2022.

[5] Higher School of Networks. 9 main threats to corporate information security! https://esr.rnp.br/seguranca/ameacas-seguranca-da-informacao/.

[6] NordLayer. Network security basics. URL https://nordlayer.com/learn/network-security/basics/#what-is-network-security-and-why-is-it-important.

[7] David Coss. The cia strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, 10, 01 2014.

[8] Roy T. Fielding, Mark Nottingham, and Julian Reschke. HTTP Semantics. RFC 9110, June 2022. URL https://www.rfc-editor.org/info/rfc9110.

[9] IBM. What is cryptography? URL https://www.ibm.com/topics/cryptography.

[10] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing, 2010. ISBN 978-0470474242.

[11] National Institute of Standards and Technology. Data encryption standard (des). Technical Report FIPS PUB 46-3, 1999.

[12] ToorCon Information Security Conference. The world's fastest des cracker. https://crack.sh/.

[13] Fabio Maino, Uri Blumenthal, and Keith McCloghrie. The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model. RFC 3826, June 2004. URL https://www.rfc-editor.org/info/rfc3826.

[14] Commerce secretary announces new standard for global information security. *NIST*, December 2001. URL https://www.nist.gov/news-events/news/2001/12/commerce-secretary-announces-new-standard-global-information-security.

[15] John Schwartz. U.s. selects a new encryption technique. *The New York Times*, October 2000. URL https://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html.

[16] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, November 2016. URL https://www.rfc-editor.org/info/rfc8017.

[17] I. Bhargavi, D. Veeraiah, and T. Maruthi Padmaja. Securing big data: A comparative study across rsa, aes, des, ec and ecdh. In Suresh Chandra Satapathy, Vikrant Bhateja, K. Srujan Raju, and B. Janakiramaiah, editors, *Computer Communication, Networking and Internet Security*, pages 355–362, Singapore, 2017. Springer Singapore. ISBN 978-981-10-3226-4.

[18] Ashok Kumar Yadav. Significance of elliptic curve cryptography in blockchain iot with comparative analysis of rsa algorithm. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 256–262, 2021. **DOI** 10.1109/ICCCIS51004.2021.9397166.

[19] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. **DOI** 10.1109/TIT.1976.1055638.

[20] Daniel Kahn Gillmor. Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS). RFC 7919, August 2016. URL https://www.rfc-editor.org/info/rfc7919.

[21] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 5–17, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. **DOI** 10.1145/2810103.2813707. URL https://doi.org/10.1145/2810103.2813707.

[22] Daniel R. L. Brown. Sec 1: Elliptic curve cryptography. *Certicom Research*, May 2009.

[23] Joseph Silverman. An introduction to the theory of elliptic curves, July 2006.

[24] Nicole Perlroth. Government announces steps to restore confidence on encryption standards. *The New York Times*, September 2013. URL https://archive.nytimes.com/bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/.

[25] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, pages 207–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33852-9.

[26] National Institute of Standards and Technology. Recommendation for password-based key derivation: Part 1: Storage applications. Technical Report NIST SP 800-132, 2010.

[27] Kathleen Moriarty, Burt Kaliski, and Andreas Rusch. PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018, January 2017. URL https://www.rfc-editor.org/info/rfc8018.

[28] Colin Percival. Stronger key derivation via sequential memory-hard functions. 01 2009.

[29] OWASP. Password storage cheat sheet. URL https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.

[30] Arm Limited. What is asic? https://www.arm.com/glossary/asic.

[31] Colin Percival and Simon Josefsson. The scrypt Password-Based Key Derivation Function. RFC 7914, August 2016. URL https://www.rfc-editor.org/info/rfc7914.

[32] Colin Percival. scrypt: A new key derivation function. In *The Technical BSD Conference*, 2009.

[33] National Institute of Standards and Technology. Recommendation for applications using approved hash algorithms. Technical Report NIST SP 800-107 Rev. 1, 2012.

[34] National Institute of Standards and Technology. Recommendation for key management: Part 1 – general. Technical Report NIST SP 800-57 Part 1 Rev. 5, 2020.

[35] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 19–35, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32055-5.

[36] Gaëtan Leurent and Thomas Peyrin. SHA-1 is a shambles: First Chosen-Prefix collision on SHA-1 and application to the PGP web of trust. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1839–1856. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL https://www.usenix.org/conference/usenixsecurity20/presentation/leurent.

[37] Wouter Penard and Tim van Werkhoven. On the secure hash algorithm family. 2008.

[38] National Institute of Standards and Technology. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical Report FIPS PUB 202, 2015.

[39] Nist selects winner of secure hash algorithm (sha-3) competition. *NIST*, October 2012. URL https://www.nist.gov/news-events/news/2012/10/nist-selects-winner-secure-hash-algorithm-sha-3-competition.

[40] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999. **DOI** 10.1109/5.771065.

[41] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 90(3): 727–752, 2010. ISSN 0165-1684. **DOI** https://doi.org/10.1016/j.sigpro.2009.08.010. URL https://www.sciencedirect.com/science/article/pii/S0165168409003648.

[42] Mehdi Hussain. A survey of image steganography techniques. *International Journal of Advanced Science and Technology. (IJAST),*, 54:113–125, 05 2013.

[43] Lenovo. What is a pixel? URL https://www.lenovo.com/us/en/glossary/pixel/.

[44] Andrew Zola. Rgb (red, green and blue). URL https://www.techtarget.com/whatis/definition/RGB-red-green-and-blue.

[45] Leonard Rosenthol, Pierre-Anthony Lemieux, Chris Lilley, Chris Seeger, and Chris Blume. Portable network graphics (PNG) specification (third edition). Candidate recommendation, W3C, September 2023. https://www.w3.org/TR/2023/CR-png-3-20230921/.

[46] Ashwin Goel. Image based steganography using python. URL https://www.geeksforgeeks.org/image-based-steganography-using-python/.

[47] Muhammad Khaerul Anam, Eko Adi Sarwoko, Edy Suharto, and Kharis Khasburrahman. Random pixel embedding for hiding secret text over video file. In *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*, pages 41–46, 2017. **DOI** 10.1109/ICICOS.2017.8276335.

[48] Deep AI. Frequency domain. URL https://deepai.org/machine-learning-glossary-and-terms/frequency-domain.

[49] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[50] Helder Eijs. A self-contained cryptographic library for python. *GitHub repository*, 2014. URL https://github.com/Legrandin/pycryptodome.

[51] PySimpleSoft. Python guis for humans! pysimplegui is the top-rated python application development environment. *GitHub repository*, 2018. URL https://github.com/PySimpleGUI/PySimpleGUI.

[52] Jeffrey Clark. Python imaging library. *GitHub repository*, 2010. URL https://github.com/python-pillow/Pillow.

[53] Leonardo Interactive Pty Ltd. Leonardo.ai. https://leonardo.ai/.

[54] May Alanzy, Razan Alomrani, Bashayer Alqarni, and Saad Almutairi. Image steganography using lsb and hybrid encryption algorithms. *Applied Sciences*, 13(21), 2023. ISSN 2076-3417. **DOI** 10.3390/app132111771. URL https://www.mdpi.com/2076-3417/13/21/11771.

[55] De Rosal Igantius Moses Setiadi. Psnr vs ssim: imperceptibility quality assessment for image steganography. *Multimedia Tools and Applications*, 80(6):8423–8444, Mar 2021. ISSN 1573-7721. **DOI** 10.1007/s11042-020-10035-z. URL https://doi.org/10.1007/s11042-020-10035-z.

[56] shreyasgangopadhyay. Performance metrics for image steganography. URL https://www.geeksforgeeks.org/performance-metrics-for-image-steganography/.

[57] Oluwakemi Christiana Abikoye, Roseline Oluwaseun Ogundokun, Sanjay Misra, and Akasht Agrawal. Analytical study on lsb-based image steganography approach. In Amit Kumar, Jacek M. Zurada, Vinit Kumar Gunjan, and Raman Balasubramanian, editors, *Computational Intelligence in Machine Learning*, pages 451–457, Singapore, 2022. Springer Nature Singapore. ISBN 978-981-16-8484-5.

[58] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: High capacity image steganography with gans, 2019.