UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

MATHEUS SCHENATTO

CRIPTOGRAFIA E ESTEGANOGRAFIA APLICADAS EM IMAGENS MÉDICAS NO PADRÃO DICOM

CAXIAS DO SUL

2021

MATHEUS SCHENATTO

CRIPTOGRAFIA E ESTEGANOGRAFIA APLICADAS EM IMAGENS MÉDICAS NO PADRÃO DICOM

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dra. Helena Graziottin Ribeiro

CAXIAS DO SUL

2021

MATHEUS SCHENATTO

CRIPTOGRAFIA E ESTEGANOGRAFIA APLICADAS EM IMAGENS MÉDICAS NO PADRÃO DICOM

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 30/11/2021

BANCA EXAMINADORA

Prof. Dra. Helena Graziottin Ribeiro
Universidade de Caxias do Sul - UCS

Prof. Dr. Andre Luis Martinotto
Universidade de Caxias do Sul - UCS

Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul - UCS

AGRADECIMENTOS

Agradeço professores que passaram valiosos conhecimentos durante a caminhada da graduação, em especial à minha orientadora Helena Graziotin Ribeiro, que me guiou na construção deste trabalho. Aos professores da banca Andre Luis Martinotto e Ricardo Vargas Dorneles que contribuíram com valiosas revisões.

Aos meus pais Alderi Schenatto e Conceição Domingas Schenatto que acompanharam todo o processo da graduação, me apoiando da forma como puderam, inclusive em com atitudes simples de motivação.

Agradeço aos amigos e colegas, pelas palavras de apoio e motivação além das trocas de experiência durante a jornada, que em momentos difíceis tornavam a caminhada mais leve.

RESUMO

O presente trabalho buscou combinar técnicas de criptografia e esteganografia para utilizar em uma aplicação de um protótipo que utilizasse imagens médicas no padrão DICOM. O processo que o protótipo segue é embutir texto criptografado dentro da imagem, assim como o processo de extração, ao identificar o texto criptografado dentro da imagem, retirá-lo e descriptografar seu conteúdo. Deste modo, foram realizadas algumas avaliações calculando métricas de similaridade, para testar diferente algoritmos de esteganografia e identificar o mais adequado para a aplicação. Identificou-se que o algoritmo LSB manteve índices melhores para similaridade entre a imagem original e a imagem que passou por esteganografia. Para criptografia foi utilizado o algoritmo RSA, por ser de chave pública e possuir um nível alto de segurança.

Palavras-chave: Criptografia. Esteganografia. DICOM. LSB. RSA.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ASCII *American Standard Code for Interchange Information*

BMP *Bitmap*

BPCS *Bit-Plane Complexity Segmentation*

CIA *Confidentiality, Integrity and Availability*

DCT *Discrete Cosine Transform*

DES *Data Encryption Standard*

DICOM *Digital Imaging and Communications In Medicine*

DSA *Digital Signature Algorithm*

ECC *Elliptical Curve Cryptography*

ECDH *Elliptic-curve Diffie–Hellman*

ECDSA *Elliptic Curve Digital Signature Algorithm*

IODs *Information Object Definitions*

JPEG *Joint Photographic Experts Group*

LSB *Least Significant Bits*

MAD *Max of Absolute Differences*

MDC Máximo Divisor Comum

MIT *Massachusetts Institute of Technology*

MSE *Mean Squared Error*

NIST *National Institute of Standards and Technology*

OSI *Open System Interconnection*

PNG *Portable Network Graphics*

RMSE *Root Mean Squared Error*

RSA Rivest-Shamir-Adleman

SAD *Sum of Absolute Differences*

SSD *Sum of Square Differences*

SSIM *Structural Similarity Index Measure*

SUMÁRIO

1 INTRODUÇÃO

Cada vez mais são trocados dados através da internet, pois o acesso à informação está mais acessível e difundido globalmente. Com isso, informações pessoais e sigilosas também são enviadas pela rede, facilitando uma série de operações, como transações bancárias, trocas de mensagens, distribuição de áudios e vídeos entre outros. Porém, esses dados, caso não estejam protegidos, podem ser roubados ou interceptados por *hackers*, assim como os dispositivos eletrônicos também podem ser danificados se não tiverem algum tipo de proteção contra ataques. Por esse motivo, estudos relacionados a criptografia são pertinentes na atualidade, pois muito se discute em garantir a privacidade dos usuários, e a segurança de informações sensíveis de sistemas e pessoas. A esteganografia é uma área que pode ser utilizada de forma aliada à criptografia, que consiste em "esconder" dados dentro de imagens, vídeos, áudios, documentos de textos dentro outros tipos de arquivos digitais. Nesse contexto, os dados não são necessariamente criptografados, assim como nem toda informação criptografada passa pelo processo de esteganografia. Recentemente, a empresa de segurança eletrônica *Check Point Security* descobriu no ano de 2017, uma falha no aplicativo *Whatsapp*, que permitia que os dados da conta do usuário fossem hackeados através de um *malware* escondido dentro de imagens.

Esteganografia consiste na arte ou ciência de se escrever mensagens ocultas de tal forma que ninguém saiba que essa mensagem exista. É diferente da criptografia, pois nesta sabemos que a mensagem existe, só não conseguimos decifrá-la. No contexto digital, a esteganografia possui inúmeras aplicações práticas. Ela é uma das técnicas utilizadas para implementar mecanismos de verificação de direitos autorais em imagens e outras mídias. Pode também ser utilizada para a divulgação de mensagens ocultas, passando despercebidas àqueles que não devem recebê-la. Acredita-se que o próprio Bin Laden tenha usado esse recurso para enviar ordens aos seus subordinados, porém, isso é apenas uma especulação. Discute-se também, o uso de esteganografia em Imagiologia Médica, onde existe um padrão estabelecido para o armazenamento, impressão e transmissão de informações de pacientes em imagens médicas, o *Digital Imaging and Communications In Medicine* (DICOM). Então é importante que essas informações sejam protegidas de vazamentos e inclusive que essas informações sejam vinculadas ao arquivo correto.

Imagens são a mídia digital mais utilizada em esteganografia, possuem diversos formatos e podem ser armazenadas em formato *Bitmap* (BMP) ou em um formato comprimido como o *Joint Photographic Experts Group* (JPEG) ou *Portable Network Graphics* (PNG). A ocultação de informação nas imagens, é realizado no domínio de frequência ou no domínio espacial (SULLIVAN *et al.*, 2004).

1.1 OBJETIVOS

Normalmente, criptografia e esteganografia não são abordadas de maneira conjunta, mesmo pertencendo ao mesmo ramo de estudo, a criptologia. O trabalho tem como objetivo geral desenvolver um protótipo de *software* que será responsável por realizar a inserção/extração de mensagens criptografadas em arquivo de imagem através de esteganografia. A finalidade de criar tal *software*, é para aplicar os conhecimentos adquiridos durante o desenvolvimento do trabalho, aplicando as técnicas selecionadas em uma aplicação real, e analisar o seu funcionamento.

Para atingir o objetivo geral, alguns objetivos específicos foram estabelecidos. Identificar se existe uma restrição ou recomendação de algoritmos de esteganografia e criptografia que podem ser utilizados em conjunto para a construção de uma aplicação, para isso serão estudadas e classificadas técnicas de ambos os campos de estudos.

Focado em esteganografia, algumas das técnicas estudadas serão validadas para identificar o nível de alteração que causam nas imagens utilizadas, afim de estabelecer qual dos algoritmos é o ideal para ser implementado no *software* desenvolvido.

Em criptografia, os objetivos são classificar os algoritmos criptográficos, identificar seu nível de segurança assim como sua popularidade, ou seja, identificar alguns dos mais utilizados atualmente. Essas validações também serão levadas em consideração para a escolha do algoritmo que será utilizado no *software* desenvolvido.

Como o trabalho é focado em uma aplicação para imagens, um objetivo relacionado com esse tipo de arquivo, é identificar como as imagens digitais são obtidas e como são representadas computacionalmente. É importante realizar essas validações, pois aplicar esteganografia a uma imagem, não deixa de se tratar de uma transformação que ela sofrerá, de tal maneira que devemos identificar sua estrutura para entender como será possível esconder informações dentro dela.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta um *overview* dos três objetos principais de estudo desse trabalho, criptografia, esteganografia e imagem. Nesse capítulo é abordado a conceituação histórica assim como alguns conceitos iniciais de cada área, que serão aprofundados nos capítulos posteriores. Os conceitos básicos sobre imagem, sua representação computacional e a forma como é obtida, são abordados nesse capítulo.
- O Capítulo 3 aprofunda os estudos em criptografia, apresenta algumas técnicas utilizadas, assim como sua classificação referente ao tipo de chave que utiliza, privada ou pública.

É feita a diferenciação entre cifras simétricas e assimétricas, apresentando as principais diferenças e identificando como classificar um algoritmo dentro de uma dessas definições.

- O Capítulo 4 aprofunda os estudos em esteganografia, apresentando os principais conceitos, os principais meios de utilização, assim como algumas técnicas de esteganografia em imagem são apresentadas. O assuntos de esteganálise também é abordado, referente aos ataques em esteganografia.
- O Capítulo 5 apresenta a proposta de desenvolvimento de *software*, assim como propõe as métricas que serão utilizadas para identificar quais algoritmos utilizar na construção da aplicação.
- O capítulo 6 apresenta o desenvolvimento relacionado com a proposta. Nele são apresentados os algoritmos de esteganografia estudados, assim como apresenta a coleta de dados para a análise e como as métricas foram desenvolvidas para gerar os dados de análise. Também é apresentado como o algoritmo de criptografia é utilizado.
- No capítulo 7 os dados obtidos através do que foi apresentado no capítulo 6 são analisados e é realizado uma explanação sobre o que foi obtido, analisando números e imagens.
- No capítulo 8, após a análise de dados sobre os algoritmos de esteganografia estudados é apresentado o protótipo de criptografia e esteganografia criado. É explicado de maneira mais detalhada o funcionamento do algoritmo de esteganografia escolhido para o projeto.
- O Capítulo 9 contempla as conclusões do trabalho.

2 CRIPTOGRAFIA, ESTEGANOGRAFIA E IMAGEM

Este capítulo tem como objetivo introduzir os conceitos iniciais sobre criptografia, esteganografia e imagem digital, assim como um pouco de conceituação histórica sobre os temas abordados para conhecer as origens de tais técnicas e por consequência, a sua transformação para os meios digitais. É importante que os conceitos básicos sobre cada item do capítulo sejam entendidos de forma isolada, para que posteriormente, quando eles forem unificados, se possa ter uma compreensão global do estudo desenvolvido.

2.1 CRIPTOGRAFIA

Essa seção apresenta a percepção histórica da criptografia. Também é definido o conceito de segurança da informação, pois segundo Stallings (2015), essa área possui medidas para desviar, prevenir e detectar violações de segurança e a criptografia está amplamente relacionada, pois é aplicada em uma gama extensa de aplicações em segurança. É feita uma breve elucidação sobre alguns comportamentos de ataque aos sistemas computacionais, para ilustrar a importância da privacidade. Então, para entender a criptografia, é importante estudar sobre os conceitos de segurança de computadores.

2.1.1 Conceituação Histórica

A palavra criptografia tem sua origem vinda do grego, é derivada de *cryptos*, que significa secreto, oculto. Ela estuda os métodos para codificar mensagens, para que somente o remetente e o destinatário consigam interpretar o significado do texto. Existem diversas técnicas, uma das mais simples é a transladação do alfabeto, que consiste em substituir uma letra pela próxima ocorrência alfabética. Outra técnica simples, que acredita-se que foi uma das primeiras utilizadas na história que se tem registro, ficou conhecida como Cifra de César. Caio Júlio César, era um importante líder militar romano, e utilizava dessa técnica criptográfica para se comunicar com suas tropas de combate através da Europa. A técnica tinha como fundamento gerar uma rotação à direita ou à esquerda no alfabeto, assim, gera-se um novo alfabeto, com as mesmas letras, mas com posições diferentes, então para escrever a mensagem, compara-se a letra do alfabeto normal com a do alfabeto rotacionado, escrevendo a letra do segundo alfabeto. Posteriormente essas duas técnicas ficariam enquadradas nos modelos de substituição (COUTINHO, 2005).

2.1.2 Criptografia e Segurança

A definição de Segurança de Computadores é sugerida pelo Manual de Segurança de Computadores da *National Institute of Standards and Technology* (NIST) por Guttman & Ro-

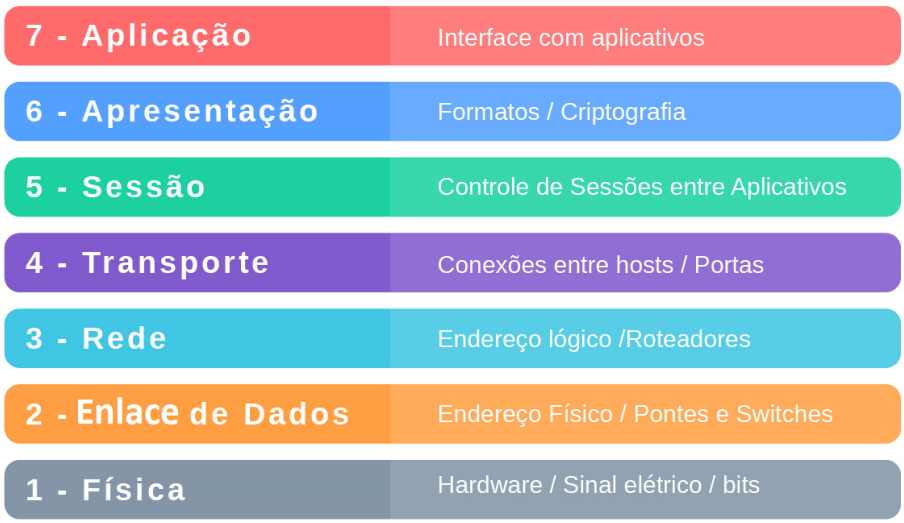
back (1995), da seguinte forma: a proteção oferecida para um sistema de informação automatizado a fim de alcançar os objetivos de preservar a integridade, a disponibilidade e a confidencialidade dos recursos do sistema de informação (incluindo *hardware*, *software*, *firmware*, informações/dados e telecomunicações).

Para Stallings (2015) existem três objetivos que se almejam alcançar com segurança, formando a tríade para *Confidentiality, Integrity and Availability* (CIA) do acrônimo em inglês:

1. Confidencialidade (dados e usuários): que se assegura que informações privadas e confidenciais não fiquem disponíveis a pessoas não autorizadas.
2. Integridade: entende-se que os dados devam ser manipulados somente por agentes autorizados, assim como a utilização dos sistemas seja livre de manipulações.
3. Disponibilidade: os sistemas devem operar de forma que os serviços fiquem sempre disponíveis para os usuários autorizados.

Foi criado um padrão de arquitetura internacional para comunicação segura entre computadores, conhecido como arquitetura *Open System Interconnection* (OSI). Essa arquitetura é dividida em sete camadas que estão representadas na Figura 1. Ela também prevê uma visão geral útil e abstrata dos conceitos de ataque à segurança, mecanismos de segurança e serviços de segurança, que são conceitos que permeiam o assunto abordado (STALLINGS, 2015).

Figura 1 – Arquitetura OSI



Fonte: O Autor (2021).

Ataques à segurança podem ser divididos em ativos e passivos. Os ataques passivos possuem a natureza de olhar ou monitorar transmissão de dados, onde não se tem a permissão para tal. O objetivo é obter as informações para vazá-las e também para analisá-las. Ataques

ativos, por sua vez, envolvem modificação do fluxo de dados, ou a criação de um fluxo falso. Eles envolvem: disfarce, repasse, modificação de mensagens e negação de serviço (STALLINGS, 2015).

Serviços de segurança são fornecidos pelas camadas do protocolo de comunicação entre sistemas, que garantem a transferência adequada das informações, prezando a segurança e a integridade dos dados. Os serviços descrevem diretrizes de segurança que são implementadas pelos mecanismos de segurança. Os serviços são divididos em cinco, como definido por Stallings (2015):

1. Autenticação: certifica-se que a comunicação é autêntica, ou seja, garante ao destinatário que a mensagem tem a origem de que afirma ter vindo. Também deve garantir que a conexão não sofra interferência, de modo que um terceiro não finja ser uma das partes, para transmitir ou receber de forma não autorizada.

2. Controle de Acesso: refere-se à capacidade de limitar o acesso ao sistema por meio de regras e privilégios de acesso.

3. Confidencialidade dos dados: proteção dos dados contra ataques.

4. Integridade dos dados: garante que as mensagens sejam recebidas do mesmo modo como foram enviadas, sem duplicação, inserção, modificação ou reordenação. Deste modo, sendo por meio de ataques ou de erros na aplicação, a mensagem enviada deve ser a mesma no momento da recepção.

5. Irretratabilidade: impede a negação da recepção ou emissão de uma nova mensagem a ser transmitida.

A criptografia entra no aspecto dos mecanismos de segurança. Dentro da arquitetura OSI, esses mecanismos são implantados principalmente nas camadas de transporte e aplicação. Eles podem ser divididos entre mecanismos de codificação reversíveis e irreversíveis. O reversível é um algoritmo de encriptação que permite que os dados sejam encriptados e depois decriptados. Os irreversíveis incluem algoritmos de *hash* e códigos de autenticação de mensagens, que são usados em assinatura digital principalmente (STALLINGS, 2015) .

As redes de comunicação abertas, como a Internet, não possuem segurança intrínseca para os usuários. A criptografia é capaz de garantir a autenticidade das informações nos meios eletrônicos. Desse modo, ela é importante linha de defesa contra *snooping* (bisbilhotagem) e *spoofing* (falsificação) (STALLINGS, 2015).

Historicamente a criptografia está basicamente relacionada ao sigilo das informações, porém atualmente ele tem usos voltados para autenticação, integridade e irrefutabilidade. Geralmente, estes itens são utilizados em conjunto. Por exemplo, um *e-mail* pode ser criptografado

e assinado digitalmente, assim, confidencialidade e autenticação estão garantidos. Sendo que a assinatura digital é única por *e-mail*, a integridade também é preservada. Esses conceitos são definidos por Braga & Dahab (2015) a seguir:

1. Confidencialidade: obtida através do uso de criptografia, para tornar uma informação secreta. Envio de *e-mail* encriptados e manter arquivos encriptados no disco, são exemplos de confidencialidade.

2. Autenticação: obtida pela utilização da criptografia para validar a identidades das entidades. Assinaturas digitais para identificar a autoria de documentos eletrônicos, é um bom exemplo do seu uso.

3. Integridade: garante que uma porção de dados não foi modificada. Podemos ver isso nos códigos de detecção de erros, mecanismo de verificação de integridades.

4. Irrefutabilidade: meio de garantir que o autor de uma mensagem não possa negar a sua autoria.

2.2 ESTEGANOGRAFIA

A esteganografia, atualmente, pode ser aplicada em vários tipos de mídias digitais, e pode ser utilizada juntamente com a criptografia. Para Sampaio & Jackowski (2014), por se tratar de uma ferramenta a favor da privacidade, é natural que as pessoas e as autoridades se preocupem com o conteúdo que está sendo trafegado de forma que não possa ser identificada. Então, assim como na criptografia existe a criptoanálise, para decifrar as mensagens codificadas, na esteganografia existe a esteganálise, responsável por identificar se existe informação oculta dentro dos mais diversos formatos de arquivo.

2.2.1 Conceituação Histórica

A esteganografia é uma arte antiga, os gregos já a utilizavam para mandar mensagens em tempos de guerra. Nas Histórias de Herodotus, existe um trecho que mostra o uso da esteganografia. Um mensageiro se disfarçou de caçador para enviar ao rei uma mensagem, que estava escondida dentro de uma lebre. Disfarçado, conseguiu passar pelos portões do castelo e entregar a mensagem ao rei (JULIO; BRAZIL; ALBUQUERQUE, 2007).

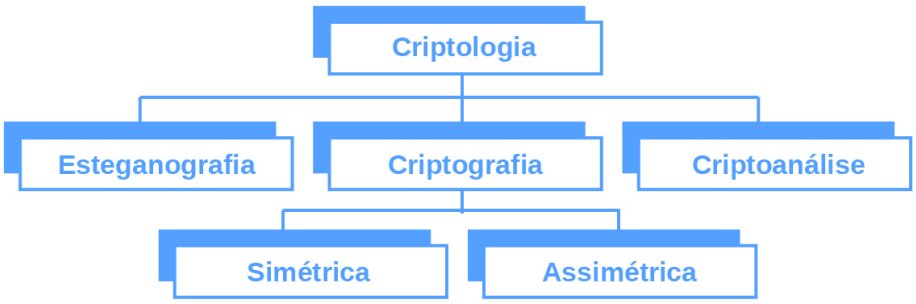
Foi durante a Idade Média que a esteganografia foi estudada e desenvolvida. No ano de 1499, um monge de nome Tritheimus escreveu uma série de livros chamados *Steganographia*, onde ele descreveu algumas técnicas. Uma técnica desenvolvida na idade média por Girolamo Cardano, consistia numa grade de lâmina, que randomicamente definia retângulos. A quantidade e o posicionamento dos retângulos era o segredo. A grade era removida e nos espaços remanescentes, o remetente preenchia com palavras. Ao chegar no destinatário, deveria colocar

a mesma grade sobre o papel, e então podia ler a mensagem sem problemas. Os primeiros experimentos com tintas invisíveis também começaram nesta Era (JULIO; BRAZIL; ALBUQUERQUE, 2007).

2.2.2 Esteganografia moderna e sua relação com a Criptografia

A esteganografia possui muitas desvantagens quando comparada com a criptografia. Exige um *overhead* para esconder relativamente poucos bits de informação, porém é possível torná-la mais eficaz. Quando o sistema é descoberto, se torna totalmente inútil, mas isso também pode ser melhorado quando o método de inserção depender de chave. Como alternativa, a mensagem pode ser criptografada antes de ser ocultada por esteganografia. A vantagem é que ela pode ser utilizada por aqueles que se prejudicam caso sua comunicação secreta (não necessariamente o conteúdo) for descoberta. A Figura 2 apresenta hierarquicamente os ramos da Criptologia (STALLINGS, 2015).

Figura 2 – Hierarquia da criptologia



Fonte: O Autor (2021).

As técnicas apresentadas na conceituação história, possuem seus equivalentes contemporâneos. Um exemplo, propõe ocultar uma mensagem utilizando os bits menos significativos dos *frames* em um CD. A resolução máxima do formato *Kodak Photo CD* é de 3096×6144 pixels, cada pixel contendo 24 bits de informações de cor RGB. O bit menos significativo de cada pixel pode ser modificado sem que cause muita perda de qualidade na imagem. Deste modo é possível esconder 130Kb em uma foto digital (WAYNER, 2008).

Novas técnicas de esteganografia são produzidas para interagir com os meios de comunicação atuais. Muitos artistas e gravadoras utilizam marca d'água para proteger suas obras, justamente com o aumento da pirataria e de sites de internet, onde é possível baixar todo tipo de

mídia, está técnica se mostra aliada na proteção dos direitos autorais. O uso da esteganografia em *software* tem enorme potencial, pois tem a capacidade de ser usada em diversos formatos de mídia, fotos, áudios, vídeos, textos, dentre outros (JULIO; BRAZIL; ALBUQUERQUE, 2007).

A criptografia e a esteganografia unidas, são uma boa tecnologia eficiente para aumentar a privacidade online dos usuários. Porém tal nível de privacidade preocupa as autoridades políticas e policiais. Planos terroristas podem ser montados totalmente em sigilo, por isso, existem muitas propostas para controle de privacidade em andamento, como *PATRIOT*, *Carnivore*, *DMCA*, *CAPS II* entre outros. Grande parte da informação que circula na *internet* é constantemente vigiada pelo projeto *Echelon*. Esse projeto visa filtrar toda informação em busca de terroristas, e é aplicado principalmente por países como Estados Unidos, Reino Unido, Austrália e Canada (ROCHA *et al.*, 2004).

2.3 IMAGEM

Existem diversas formas de se obter imagem a partir do ambiente. Segundo Gonzalez & Woods (2010), é possível gerar utilizando um único sensor, sensores por varredura de linha e sensores matriciais. Nesta seção, será abordado apenas a aquisição de imagens por meio de sensores matriciais, pois é a forma como a maioria dos dispositivos fotográficos digitais funcionam atualmente, e consequentemente, se relaciona com os algoritmos que serão estudados nos próximos capítulos.

2.3.1 Aquisição de imagens

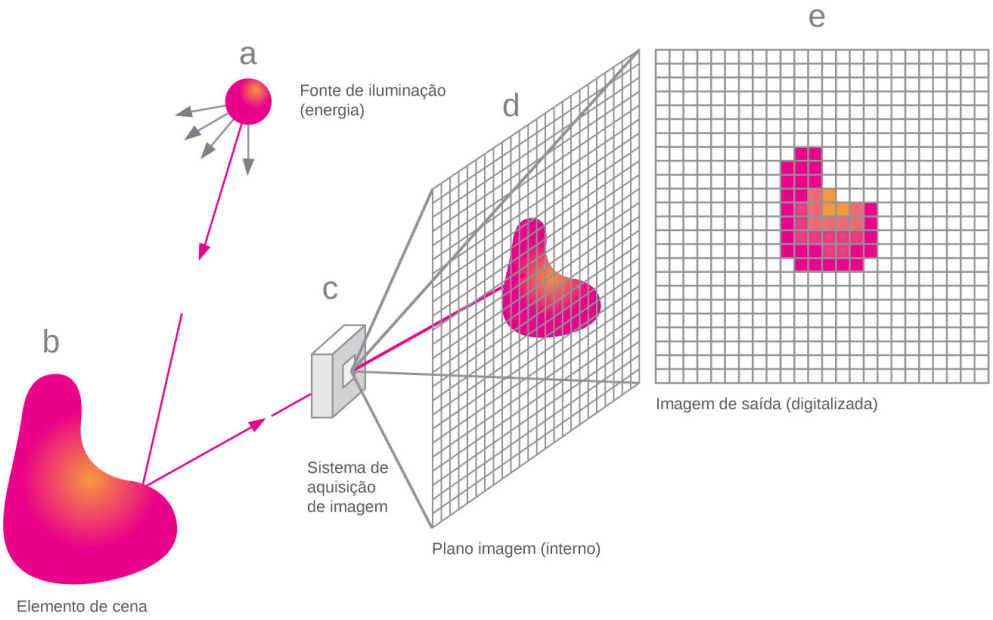
A maioria das imagens que conhecemos, são geradas pela combinação de uma fonte de iluminação e a reflexão ou absorção de energia dessa fonte pelos elementos da cena. A iluminação pode ter origem através de energia eletromagnética, como um sistema de raios X, de radar ou infravermelho. Outras maneiras menos tradicionais de geração de imagens são por meio de ultrassom ou até mesmo por um padrão de iluminação gerado computacionalmente. De acordo com a forma de geração de iluminação, ela pode ser refletida pelos objetos ou transmitida através deles. No primeiro caso, a luz pode ser refletida por uma superfície plana, e no segundo caso, é quando os raios X passam pelo corpo humano para gerar uma imagem radiográfica. A ideia para transformar energia de iluminação em imagens digitais por meio de sensores é relativamente simples: a energia que entra é transformada em tensão pela combinação de energia elétrica de entrada e do material do sensor, sensível ao tipo de energia que está sendo detectado. A resposta dos sensores é a forma de onda da tensão de saída, então, uma quantidade digital é obtida por cada sensor através da digitalização da resposta (GONZALEZ; WOODS, 2010).

Uma das formas de aquisição de imagens, predominantemente encontradas nas câmeras digitais, é através de sensores matriciais. Um sensor típico, é representado por uma matriz CCD, possuindo uma grande variedade de propriedades sensoras, que ficam dispostas em ar-

ranjos matriciais de 4000×4000 elementos ou mais. Cada sensor responde proporcionalmente à integral da energia projetada sobre o sensor. A matriz de sensores é bidimensional, então sua principal vantagem é que uma imagem inteira pode ser capturada refletindo o padrão de energia sobre a matriz (GONZALEZ; WOODS, 2010).

A Figura 3 representa o processo de aquisição de imagem digital, caracterizando os elementos que formam a cena e a imagem: (a) a fonte de luz, (b) objeto da cena, (c) câmera digital ou algum outro sistema de aquisição de imagens, (d) projeção da cena no plano imagem e (e) a imagem digitalizada.

Figura 3 – Aquisição de imagem digital por sensor matricial



Fonte: O Autor (2021).

As imagens são expressas através de funções bidimensionais na forma de $f(x, y)$. O valor ou amplitude f nas coordenadas (x, y) é uma valor escalar positivo cujo significado físico é obtido através da origem da imagem. Os valores de intensidade de uma imagem são proporcionais à energia irradiada pela fonte. Por esse motivo, $f(x, y)$ deve ser diferente de zero e finito:

$$0 < f(x, y) < \infty \tag{2.1}$$

A função pode ser entendida por dois parâmetros: (1) a quantidade de iluminação que incide na cena observada; e (2) a quantidade de iluminação que os objetos da cena refletem. Tais elementos são conhecidos como iluminação e refletância, expressos por $i(x, y)$ e $r(x, y)$. As duas funções se combinam e formam $f(x, y)$:

$$f(x, y) = i(x, y)r(x, y) \tag{2.2}$$

onde

$$0 < i(x, y) < \infty \tag{2.3}$$

e

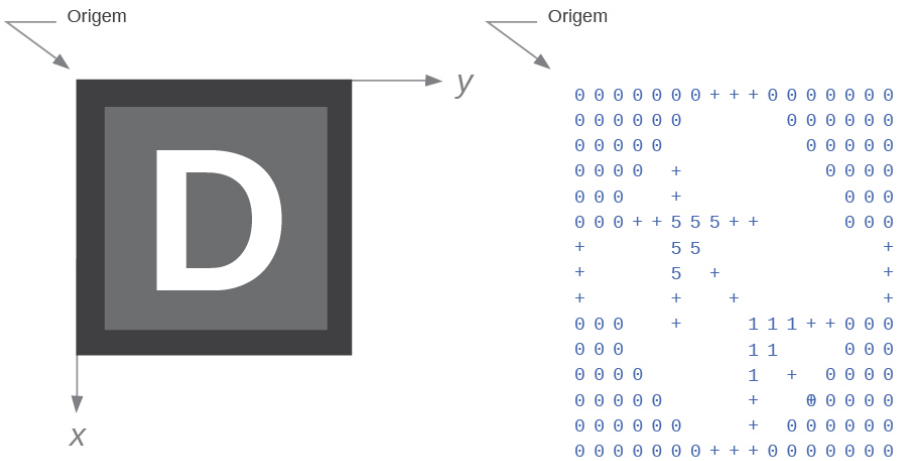
$$0 < r(x, y) < 1 \tag{2.4}$$

A Equação 2.4 mostra que a refletância está entre 0 (absorção total) e 1 (refletância total). A fonte de iluminação determina $i(x, y)$, e as características dos objetos da cena determinam $r(x, y)$ (GONZALEZ; WOODS, 2010).

2.3.2 Representação de imagens digitais

Suponhamos que uma imagem seja obtida através de um sensor matricial, e após, sejam realizadas as funções de amostragem e quantização, processos para a digitalização da imagem. Podemos estabelecer uma matriz 2-D, $f(x, y)$ contendo M linhas e N colunas, onde (x, y) são coordenadas discretas. O valor da origem da imagem (onde ela começa), é representado por $f(0, 0)$, e o próximo valor na mesma linha é $f(0, 1)$.

Figura 4 – Representações gráficas de imagem



Fonte: O Autor (2021).

A Figura 4, nos mostra as duas formas mais utilizadas de representar uma imagem digitalmente. A primeira mostra como uma imagem seria visualizada em um monitor. O nível de cinza é proporcional ao valor da função f para cada ponto. Nesse caso, temos apenas três valores de intensidade, então cada ponto possui o valor 0, 0.5 ou 1. Então quando o computador

faz a conversão desses valores, são exibidas as cores preto, cinza e branco respectivamente. A segunda representação está apenas em valores numéricos para $f(x, y)$. Nesse caso, mostra uma matriz de 600×600 elementos, ou seja, 360000 números. Essa representação é muito importante para o desenvolvimento de algoritmos de processamento de imagens, pois é através da alteração dos valores da matriz, que realizamos as transformações necessárias para cada situação. Na forma de equação, representamos uma matriz numérica M $\times$ N como representado na Figura 5.

Figura 5 – Matriz M x N

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Fonte: Gonzalez & Woods (2010).

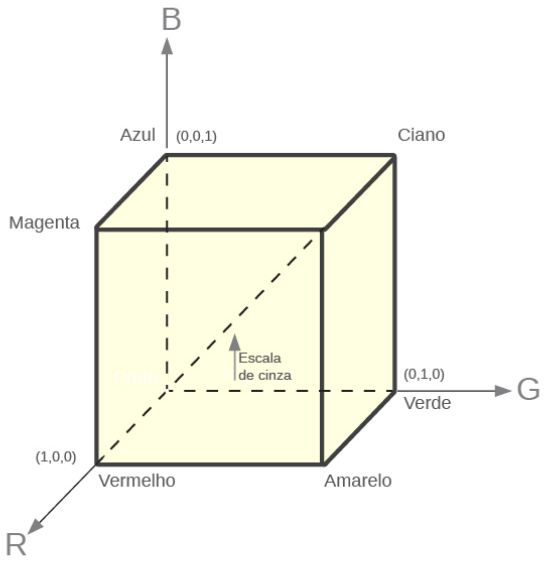
Ambos os lados da equação são equivalentes a fim de representar quantitativamente uma imagem digital. O lado direito é uma matriz de números reais, onde cada elemento é chamado de pixel (GONZALEZ; WOODS, 2010).

2.3.3 Modelo RGB de cores

O objetivo de um modelo de cores, é a padronização da especificação das cores. Basicamente, um modelo é uma especificação em um sistema de coordenadas, em um subespaço do sistema, onde cada ponto representa uma cor. Um dos modelos mais utilizados é o RGB. Nesse modelo, cada cor é representada pela mescla de componentes espectrais primários de vermelho, verde e azul em um sistema de coordenadas cartesianas. O subespaço do sistema é um cubo, como representado na Figura 6. Imagens representadas no modelo RGB, são formadas por três componentes de imagem, uma para cada cor primária. Em um monitor, essas três informações de imagem, se combinam para produzir uma cor composta. A profundidade de pixel é o número de bits utilizados para representar cada pixel da imagem. Cada componente de cor da imagem é expresso por 8 bits, então cada pixel de cor é representado por um trio de valores com profundidade de 24 bits, 3 planos de imagem, multiplicado pela quantidade de bits utilizada para sua representação. No modelo RGB de cores, para imagem de 24 bits de profundidade, o número de cores possíveis de se representar é $(2^24) = 16.777.216$ (GONZALEZ; WOODS, 2010).

Na imagem, os valores RGB primários estão representados em cada um dos três vértices, a cor preta está na origem, a cor branca está no ponto mais distante da origem. A escala de cinza vai do preto até o branco, no segmento de reta que une esses pontos. As demais cores possíveis de serem representadas são pontos dentro do cubo e são definidas por um vetor que tem início na origem (GONZALEZ; WOODS, 2010).

Figura 6 – Esquema do cubo de cores RGB



Fonte: O Autor (2021).

3 CRIPTOGRAFIA

O criptógrafo é responsável por transformar um texto comum em um texto cifrado, isto é, em uma mensagem criptografada. O trabalho do criptoanalista é inverso, tenta tornar legível, algo que estava criptografado. Até a década de 70, o seu uso era predominantemente militar, e após, com a popularização do computador e principalmente da internet, a criptografia tomou papel importante na vida diária das pessoas, pois existem diversas ferramentas que trabalham com informações sigilosas do indivíduo, tais como aplicativos de banco e sistemas do governo, entre outros (FIERRASGA, 2010).

Texto cifrado significa que o texto passou por um processo de codificação, também conhecido como *ciphertext*. O processo de codificação pela qual a mensagem passa, é conhecido como cifração ou encriptação (STALLINGS, 2015).

Existem dois tipos de sistemas criptográficos mais conhecidos e utilizados, chamados de criptografia de chave simétrica, e criptografia de chave assimétrica. Basicamente, no sistema simétrico, a chave de codificação é a mesma da decodificação. No sistema assimétrico, existe uma chave pública para encriptar a informação, e uma outra chave privada diferente, utilizada no processo de decodificação. Essas chaves são matematicamente relacionadas (BRAGA; DAHAB, 2015).

3.1 TÉCNICAS CRIPTOGRÁFICAS

Nas subseções a seguir, serão apresentadas algumas técnicas e algoritmos de criptografia. Existem diversas classes de técnicas, geralmente divididas entre cifras simétricas e assimétricas. Será apresentada a lógica das técnicas assim como apresentados alguns códigos para exemplificar o funcionamento dos algoritmos de criptografia.

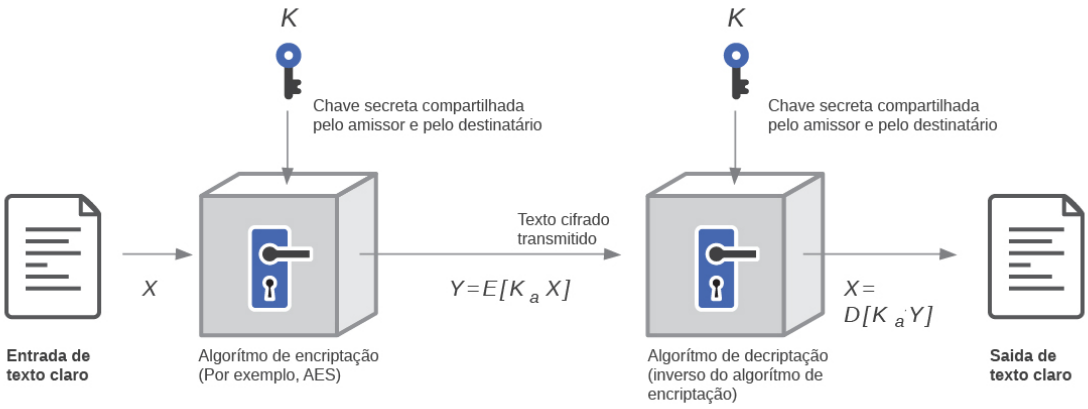
3.1.1 Cifras Simétricas

A chamada encriptação convencional, ou de chave única possui técnicas de cifra simétrica, que foram as primeiras a serem desenvolvidas e ainda continuam sendo as mais utilizadas, mesmo após a criação da encriptação por chave pública (STALLINGS, 2015).

Dentro das técnicas de criptografia simétrica, os algoritmos podem ser divididos entre algoritmos de substituição e de transposição. Nesta seção, serão estudados apenas os algoritmos com ênfase em substituição, pois são técnicas clássicas e amplamente utilizadas nos estudos do campo criptoanalítico. A Figura 7 representa de forma geral, como pode ser representado um sistema simétrico. De acordo com Stallings (2015), um sistema de encriptação simétrica deve possuir cinco requisitos:

1. Texto claro: a mensagem com os dados originais.
2. Algoritmo de encriptação: realiza as transformações na mensagem original.
3. Chave secreta: valor independente do texto, chaves diferentes geram saídas diferentes, e depende do algoritmo como as transformações e substituições ocorrerão baseadas na chave.
4. Texto cifrado: é a mensagem embaralhada após passar pelo algoritmo de encriptação.
5. Algoritmo de decriptação: algoritmo que transforma a mensagem embaralhada em texto legível. Utiliza o texto cifrado e a chave secreta.

Figura 7 – Política de criptografia simétrica



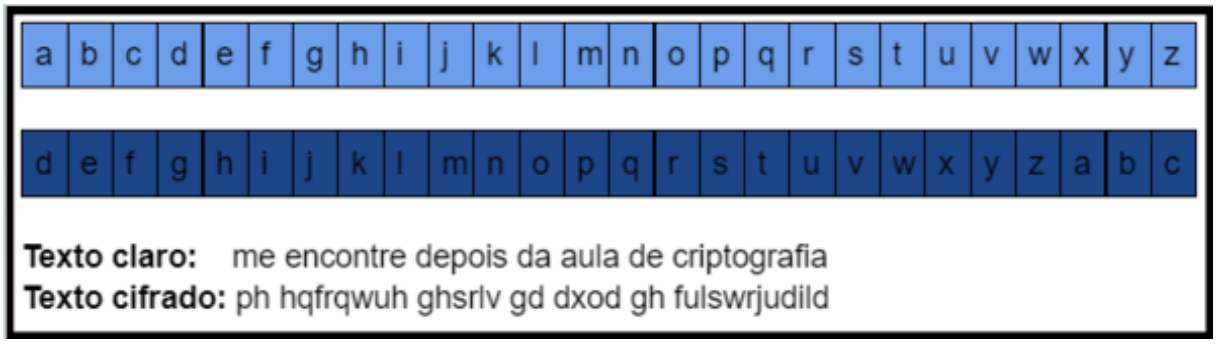
Fonte: O Autor (2021).

3.1.1.1 Cifra de César

Considerado o exemplo de uso mais antigo de uma cifra de substituição, é relativamente simples, utilizada por Júlio César. Envolve substituir cada letra do alfabeto por aquela que vem três posições à frente (STALLINGS, 2015).

No exemplo da Figura 8, podemos perceber que é gerado um novo alfabeto iniciado pela letra D, assim definindo a escrita do texto cifrado.

Figura 8 – Alfabeto para Cifra de César



Fonte: O Autor (2021).

Para criarmos um algoritmo deste método, devemos criar uma função baseada em valores numéricos, então podemos atribuir um número a cada letra do alfabeto, como apresentado na Figura 9.

Figura 9 – Atribuição de números a letras

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Fonte: O Autor (2021).

Deve-se substituir cada letra do texto claro p, pela letra do texto cifrado C

$$C = E(3, p) = (p + 3)\ mod\ 26 \tag{3.1}$$

O deslocamento, pode ser de qualquer valor, então de forma geral, podemos representar por:

$$C = E(k, p) = (p + k)\ mod\ 26 \tag{3.2}$$

onde k é um valor de 1 a 25. Dessa forma, o algoritmo de decriptação pode ser expresso simplesmente por

$$p = D(k, C) = (C - k)\ mod\ 26 \tag{3.3}$$

Uma desvantagem da Cifra de César, é que em termos computacionais, um algoritmo de força bruta quebra facilmente qualquer código, é só testar cada uma das chaves do modelo, neste caso, testando as 25 possibilidades. A Figura 10 apresenta um algoritmo em linguagem C implementando a Cifra de César (STALLINGS, 2015).

Figura 10 – Cifra de César - Linguagem C

```c
#include <stdio.h>
int main()
{
    char palavra[30], aux[30];
    int chave=1, i;
    scanf(" %s", palavra);

    while(chave < 26)
    {
        i = 0;
        while(palavra[i] != '\0')
        {
            aux[i] = palavra[i] + chave;
            if((palavra[i] + chave) > 122)
            {
                aux[i] -=26;
            }
            if((palavra[i] + chave) < 97)
            {
                aux[i] += 26;
            }
            i++;
        }
        aux[i] = '\0';
        printf("chave %d: %s ", chave, aux);

        if(chave%5 == 0)
        {
            printf("\n\n");
        }
        chave++;
    }
    printf("\n");
    return 0;
}
```

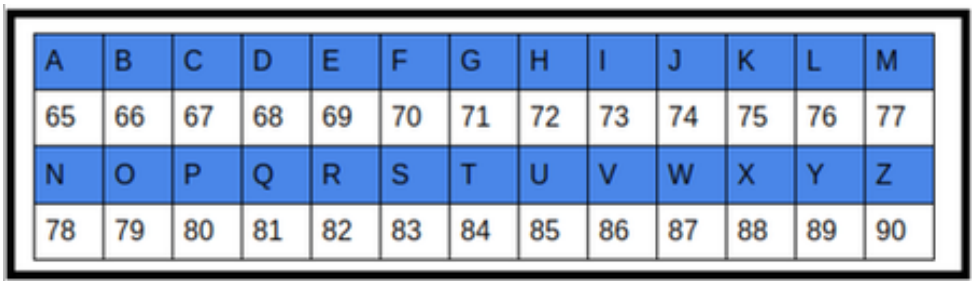
Fonte: Esperanca (2016).

3.1.1.2 Cifra de Hill

Esta cifra foi criada por Lester Hill, no ano de 1929, o que contribuiu para tornar a criptografia mais algébrica. Consiste em transformar uma matriz quadrada inversível $n \times n$ módulo 26 de forma que cada entrada $a(i, j)$ seja um número inteiro no intervalo de 1 a 26, ou seja, a matriz chave do processo (COSTA; CAETANO, 2017).

A cifra de Hill é considerada insegura e é do tempo em que a criptografia era feita apenas com papel e lápis, é facilmente quebrado e decodificado via ataque computacional. O algoritmo de encriptação é fortemente baseado em princípios de álgebra linear, por este motivo, a seguir é apresentado um passo a passo proposto por Barbosa & Cornelissen (2017), que torna o processo mais explicativo:

Figura 11 – Alfabeto para Cifra de Hill



A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

Fonte: O Autor (2021).

1. Converte-se as letras do alfabeto em número baseados na tabela *American Standard Code for Interchange Information* (ASCII), que é uma tabela amplamente utilizada em computação, como mostra a Figura 11.

Quadro 1 – Mensagem a ser criptografada

C	**I**	**F**	**R**	**A**	**D**	**E**	**H**	**I**	**L**	**L**
67	73	70	82	65	68	69	72	73	76	76

Fonte: O Autor (2021).

2. A sequência numérica deve ser agrupada em vetores coluna de tamanho n. Se o último vetor não tiver tamanho n, deve-se repetir o último número do vetor até completar o tamanho.

$$\begin{pmatrix}67\\73\\70\end{pmatrix}, \begin{pmatrix}82\\65\\68\end{pmatrix}, \begin{pmatrix}69\\72\\73\end{pmatrix}, \begin{pmatrix}76\\76\\76\end{pmatrix}$$

3. Escolher a matriz $A = (a_{i,j})_{n\ x\ n}$ que será a chave de codificação. A escolha da matriz A é feita de forma que $mdc(det, A, k) = 1$. O número de símbolos possíveis de acordo com a tabela utilizada é representado por k, neste caso $k = 26$.

$$\text{A} = \begin{pmatrix}1 & 4 & 6\\0 & 1 & 5\\3 & -1 & 8\end{pmatrix}$$

4. Inicia-se a codificação, multiplicando a matriz chave A por uma matriz B, cujas colunas são formadas pelo resultado do passo 2, com entrada subtraída por 65, pois assim conseguimos extrair o módulo de 26.

$$\text{A} \begin{pmatrix}1 & 4 & 6\\0 & 1 & 5\\3 & -1 & 8\end{pmatrix} \text{ x B } \begin{pmatrix}2 & 17 & 4 & 11\\8 & 0 & 7 & 11\\5 & 3 & 8 & 11\end{pmatrix} = \begin{pmatrix}64 & 35 & 80 & 121\\33 & 15 & 47 & 66\\38 & 75 & 69 & 110\end{pmatrix}$$

5. Depois de efetuar o produto das matrizes, caso alguma entrada da matriz final seja maior ou igual a 26, deve-se trocar esse número pelo seu resto na divisão por 26.

$$\begin{pmatrix} 12 & 9 & 2 & 17 \\ 7 & 15 & 21 & 14 \\ 12 & 23 & 17 & 6 \end{pmatrix}$$

6. Soma-se 65 a cada uma das entradas dessa matriz obtida e transforme seus vetores coluna em letras.

$$\begin{pmatrix} 77 & 74 & 67 & 82 \\ 72 & 80 & 86 & 79 \\ 77 & 88 & 82 & 71 \end{pmatrix}$$

Após realizar os 6 passos descritos, ao fazer a conversão dos números obtidos em letras de acordo com a Figura 11, é gerada a mensagem codificada MHMJPXCVRROG que será enviada ao destinatário juntamente com chave secreta, que é a matriz A.

Para Hossem *et al.* (2018), a chave do algoritmo é uma matriz C de ordem $n\ x\ n$, com determinante diferente de zero, multiplicada pela matriz X cifrada, que gera a nova matriz codificada Y. Para decodificar, o processo inicia-se em encontrar a matriz inversa de C^{-1} e realizar o produto $Y.C^{-1}$.

3.1.1.3 *Advanced Encryption Standard* - AES

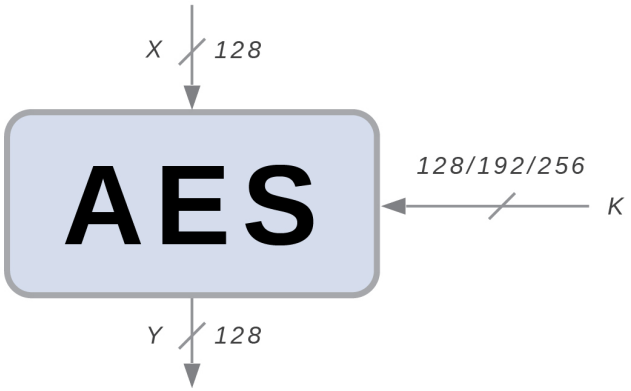
Segundo pesquisas de Dobbertin, Knudsen & Robshaw (2004), em janeiro de 1997, o NIST iniciou uma pesquisa para substituir o algoritmo antigo de criptografia utilizado pelo governo dos Estados Unidos, o *Data Encryption Standard* (DES)[1]. O algoritmo viria a ser chamado de AES, e deveria atender os seguintes requisitos:

1. A entrada deveria ser um bloco de texto sem formatação de tamanho 128 bits, com a escolha de três tamanhos para a chave, 128, 192 ou 256 bits.
2. Design público e flexível.
3. Mais seguro que o padrão DES.
4. Disponível de forma livre em todo o mundo.

O AES utiliza chave de tamanho entre 128, 192 e 256 bits, porém, para bloco de entrada utiliza apenas blocos de tamanho 128 bits. A Figura 12 representa por x o bloco de entrada, y o bloco de saída e k os possíveis tamanhos de chave (PAAR; PELZL, 2010).

[1] <https://ieeexplore.ieee.org/document/563518>

Figura 12 – AES - parâmetros de entrada/saída



Fonte: O Autor (2021).

A chave também é representada como uma matriz quadrada de bytes. A chave é expandida para um conjunto de palavras de chave. Cada palavra possui 4 bytes, e para a chave de 128 bits, são geradas 44 palavras. A ordenação dos bytes dentro da matriz é por coluna. A cifra se baseia em N rodadas (rounds), a quantidade de rodadas depende do tamanho da chave, como mostra o Quadro 2. Nas primeiras $N - 1$ rodadas, ocorrem quatro funções de transformação: *SubBytes*, *ShiftRows*, *MixColumns* e *AddRoundKey*. A rodada final possui apenas três transformações e aquela que pode ser considerada a rodada 0 possui apenas a transformação de *AddRoundKey* (STALLINGS, 2015).

Quadro 2 – *Rounds* por tamanho de chave

Rounds	**Tamanho chave (bits)**
10	128
12	192
14	256

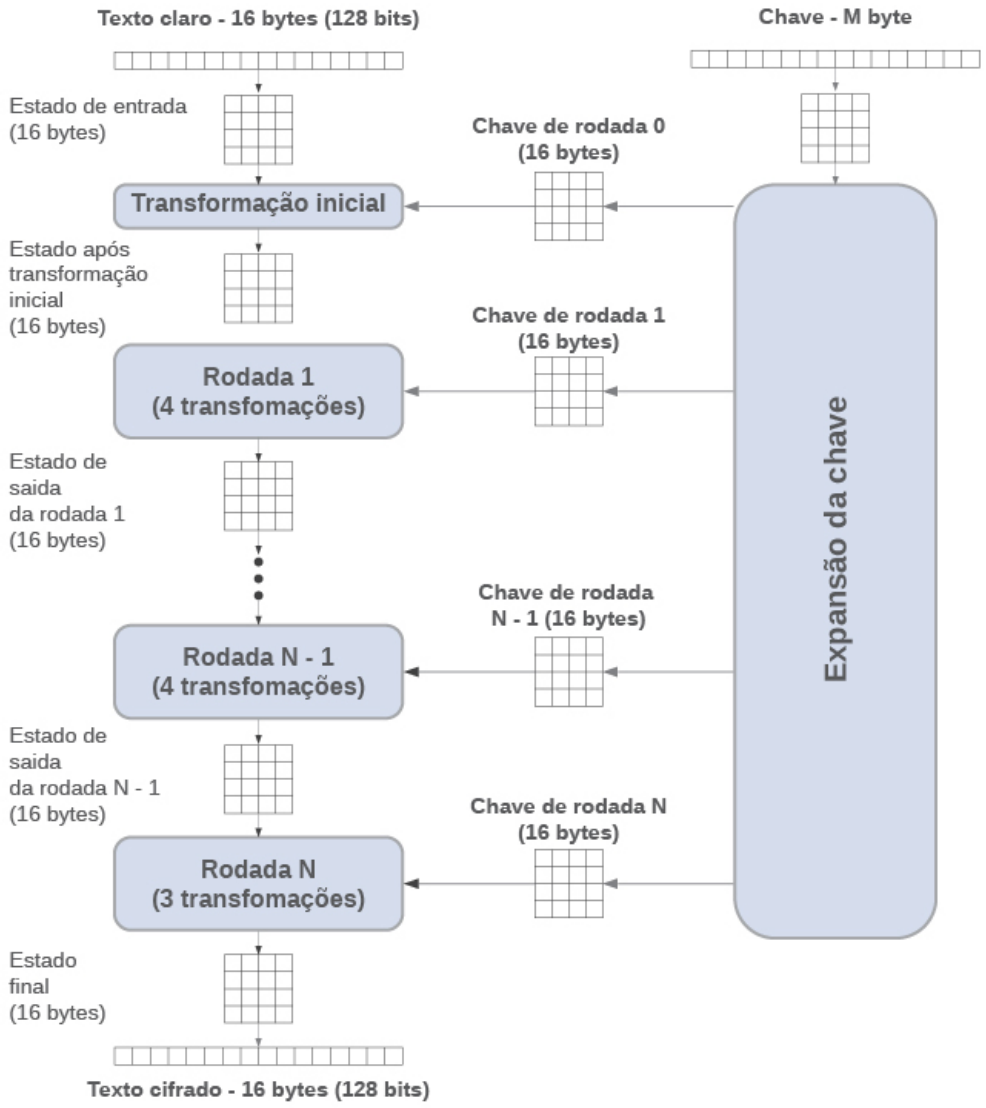
Fonte: O Autor (2021).

A Figura 13 mostra que uma matriz 4×4 é gerada como saída de cada *round*, sendo que no *round* final, a saída é o texto cifrado. A função de expansão de chave gera $N + 1$ chaves a cada rodada, sendo matrizes 4×4 diferentes. A chave do *round* é utilizada para a função de transformação *AddRoundKeyem* em cada *round*.

AES possui camadas (*layers*). Cada camada manipula todos os 128 bits de dados. Os dados podem ser referenciados também como o estado do algoritmo. Existem três camadas principais, onde cada rodada consiste na aplicação dessas camadas. Para Paar & Pelzl (2010) as camadas e suas principais funções são:

- *Key Addition layer*: uma chave de 128 bits da rodada, ou sub-chave, que foi derivada da chave principal na rotina da chave, recebe a operação de XOR para o estado do algoritmo.

Figura 13 – Processo de encriptação do AES



Fonte: O Autor (2021).

- *Byte Substitution layer (S-Box)*: cada elemento do estado é não-linearmente transformado utilizando tabelas com propriedades matemáticas. Isso gera o embaralhamento dos dados, o que garante as mudanças nos bits que são propagados através dos estados do algoritmo.
- *Diffusion layer*: responsável pelo espalhamento sobre todos os bits do estado. Possui duas subcamadas, responsáveis por operações lineares. A subcamada *ShiftRows* permuta os dados a nível de byte. A subcamada *MixColumn* é uma matriz de operações que mistura blocos de quatro bytes.

Figura 14 – Pseudoalgoritmo AES

```
Cifra(byte entrada[4*Nb], byte saida [4*Nb], palavra p[Nb*(Nr+1)])
  inicio
    byte estado[4,Nb]

    estado = entrada

    AddRoundKey(estado, p[0,Nb-1])

    para rodada = 1 faça 1 até Nr-1
      SubBytes(estado)
      ShiftRows(estado)
      MixColumns(estado)
      AddRoundKey(estado, p[rodada*Nb, (rodada+1) *Nb-1])
    fim para

    SubBytes(estado)
    ShiftRows(estado)
    AddRoundKey(estado, p[Nr*Nb, (Nr+1) *Nb-1)

    saida = estado
  fim
```

Fonte: Adaptado de NIST (2001)

3.1.2 Cifras Assimétricas

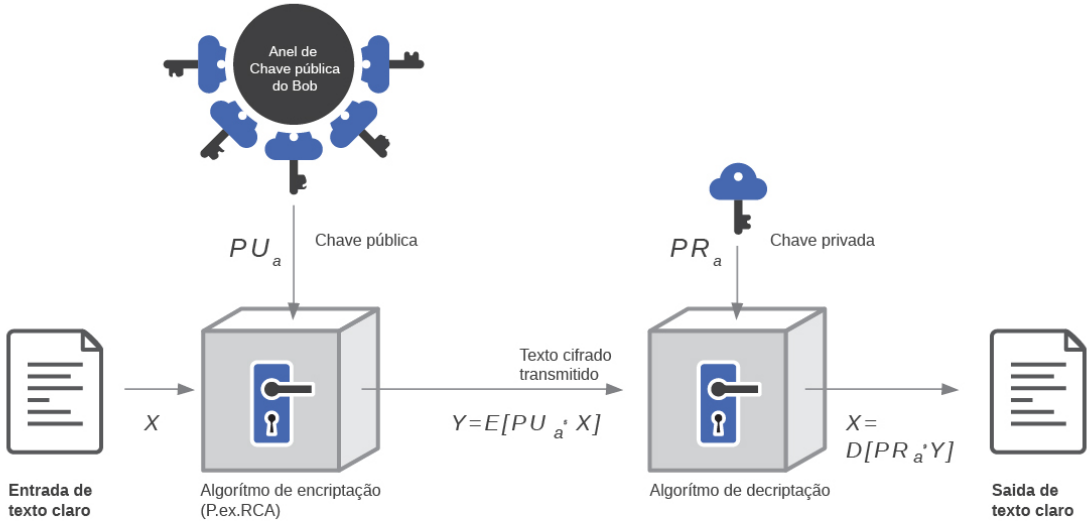
Algoritmos de cifra simétrica são muito utilizados e consideravelmente seguros, porém, eles possuem alguns pontos fracos. A distribuição das chaves deve ser feita por um canal seguro, no entanto, deve-se lembrar que o *link* de comunicação pode não ser seguro, ou seja, sendo "escutado", o que comprometeria a segurança da criptografia, esse problema é conhecido como *Key Distribution Problem*. O número de chaves pode ser considerado um problema, pois pode ser necessário lidar com um grande número de chaves, se a cada dois usuários for necessário um par de chaves, então em uma rede com n usuários existem

$$(n * (n - 1))/2 \tag{3.4}$$

pares de chaves, e cada usuário deve armazenar de modo seguro $n - 1$ chaves. A questão de "trapaça" também pode ocorrer, ou seja, aquele que envia a mensagem ou o receptor, podem divulgar a chave secreta, uma vez que possuem a mesma chave (PAAR; PELZL, 2010).

Visando vencer essas desvantagens Diffie, Hellman and Merkle propuseram algo revolucionário, baseado na ideia de que não é necessário que a chave utilizada para criptografar os dados seja secreta. A parte importante é que o receptor da mensagem criptografada, tenha uma chave secreta, que somente ele conhece. De forma análoga, funciona como uma antiga caixa de correio, onde qualquer pessoa pode colocar uma carta, mas somente o dono da caixa tem a chave para abri-la e pegar o seu conteúdo. A Figura 15 mostra uma representação de como esse sistema funciona (PAAR; PELZL, 2010).

Figura 15 – Política de criptografia de chave pública



Fonte: O Autor (2021).

Para Stallings (2015), um sistema de encriptação de chave pública possui cinco requisitos:

1. Texto claro: mensagem em dados legíveis, entrada do algoritmo.
2. Algoritmo de encriptação: realiza as transformações no texto claro.
3. Chaves pública e privada: chaves selecionadas de modo que uma é utilizada para encriptação e outra é utilizada para decriptação. A geração das chaves depende das funções aplicadas em cada algoritmo.
4. Texto cifrado: mensagem criptografada, elemento de saída.
5. Algoritmo de decriptação: recebe o texto cifrado e transforma em texto claro.

Essa arquitetura supre o ponto fraco do sistema simétrico da necessidade do envio da chave secreta por algum meio, porém os métodos de chave pública utilizam números de grande magnitude nas funcionalidades de encriptação e decriptação, nesse caso, sendo um processo mais lento que os de cifra simétrica (PERIN, 2011).

Para Perin (2011), existem três classes de algoritmos de chave pública que possuem maior relevância prática:

1. Esquema de Fatoração de Inteiros: parte do princípio que fatorar número inteiros é uma tarefa custosa. O algoritmo mais conhecido para essa classe, é o Rivest-Shamir-Adleman (RSA).

2. Esquema do Logaritmo discreto: baseado no problema do logaritmo discreto para campos finitos. Os mais conhecidos dessa classe são Diffie-Hellman, Elgamal e *Digital Signature Algorithm* (DSA)

3. Esquema de Curvas Elípticas: se enquadram nessa classe o *Elliptic-curve Diffie–Hellman* (ECDH), *Elliptic Curve Digital Signature Algorithm* (ECDSA) e o *Elliptical Curve Cryptography* (ECC). Baseado nas generalizações do esquema do logaritmo discreto.

3.1.2.1 RSA

Dos mais conhecidos algoritmos de criptografia de chave pública, o mais popular é o RSA. Foi criado em 1978 por R.L Rivest, A. Shamir e L. Adleman, na época em que trabalhavam no *Massachusetts Institute of Technology* (MIT). O significado da sigla RSA tem origem na inicial dos nomes de seus criadores (COUTINHO, 2005).

Geralmente, esse algoritmo é aplicado em sistemas que requerem cifração de pequenas porções de dados e assinaturas digitais. A segurança deste método é baseada na fatoração de números inteiros grandes (PERIN, 2011).

O algoritmo RSA não substitui as cifras simétricas, pois por diversas vezes ele se demonstra mais lento do que o algoritmo AES, por exemplo. Isso se deve aos muitos cálculos que os algoritmos de chave pública utilizam. A função unilateral subjacente desse algoritmo é multiplicar dois números primos grandes, o que computacionalmente pode ser considerado fácil, porém fatorar o seu produto resultante é relativamente difícil. O teorema de Euler e a função *phi* de Euler possuem papéis fundamentais no RSA (PAAR; PELZL, 2010).

Dado um conjunto de inteiros $Z_n = \{0, 1, ..., n-1\}$, a função *phi* de Euler é responsável por determinar quantos números dentro de um conjunto são primos relativos para um determinado valor n. Porém, calcular essa função para cada elemento do conjunto e calcular o gcd é extremamente lento se os números forem muito grandes. Existe uma relação para realizar esse cálculo de forma mais eficiente se conhecermos a fatoração de n, que é dada pelo teorema de Euler (PAAR; PELZL, 2010).

De forma genérica, para implementar o RSA, necessitamos de dois parâmetros: dois números primos p e q. Para codificar um texto é necessário conhecer o produto dos dois primos, que podemos considerar n. Para decodificar um texto, é necessário conhecer os primos p e q. A constituição da chave de codificação se dá pelo número $n = pq$. Teoricamente seria muito

fácil quebrar a criptografia deste método, porém existe um obstáculo de natureza tecnológica. Ao utilizar chaves de codificação de números muito grandes, 150 algarismos ou mais, fatorar n para achar p e q, com a tecnologia atual, levaria milhares de anos (COUTINHO, 2005).

O método pode ser entendido de uma maneira simplificada, como sugere Coutinho (2005), dividindo os processos em três partes: pré-codificação, codificação e decodificação.

Figura 16 – Alfabeto para RSA

A	B	C	D	E	F	G	H	I	J	K	L	M
10	11	12	13	14	15	16	17	18	19	20	21	22
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
23	24	25	26	27	28	29	30	31	32	33	34	35

Fonte: o Autor (2021)

O primeiro passo a executar é converter a mensagem em números. Para esse exemplo, utiliza-se a tabela de conversão representada pela imagem Figura 16, onde cada letra é representada por um número de dois algarismos para evitar ambiguidade, e o espaço entre as palavra é representado por 99. Ao converter a frase "Paraty é linda", obtemos a sequência de números $25102710293499149921182313 10$. Determina-se os parâmetros de entrada do sistema RSA, que são dois números primos distintos denotados por p e q. Definimos n como o produto dos números primos $n = pq$. A última etapa da pré-codificação realiza-se a quebra do número anteriormente obtido em blocos menores que n. Ao escolheremos $p = 11$ e $q = 13$, então $n = 143$ e a mensagem numérica pode ser quebrada nos seguintes blocos: 25 - 102 - 7 - 102 - 93 - 49 - 91 - 49 - 92 - 118 - 23 - 13 - 10.

Para codificar a mensagem é necessário obter n e um número inteiro positivo que seja inversível módulo $\Phi(n)$, ou seja $mdc(e, \Phi(n)) = 1$. Para calcular $\Phi(n)$, é necessário conhecer p e q e aplicar a fórmula

$$\Phi(n) = (p-1)(q-1). \tag{3.5}$$

Identifica-se o par (n, e) como chave de codificação do RSA. Codifica-se cada bloco anteriormente definido separadamente, e a mensagem se torna uma sequência de blocos codificados, porém, esses blocos não podem ser reunidos de forma a montar um número único. Neste ponto, define-se b como um bloco, sendo b um número inteiro positivo menor que n. Denota-se o bloco codificado por $C(b)$, desta forma, a fórmula para calcular pode ser definida como

$$C(b) = b^e (mod \quad n) \tag{3.6}$$

No ponto atual do processo de codificação, temos $p = 11$, $q = 13$, $n = 143$, $\Phi(n) = 120$ e ainda é necessários escolher e. Nesse exemplo o único valor possível para e é 7, por ser

o menor número primo não divide 120, deste modo, o bloco 102 é codificado como o resto da divisão de 102^7 por 143, obtém-se $C(102) = 119$, que fora calculado da forma reduzida $102^7 \equiv (-41)^7 \equiv -41^7 \equiv -81 * 138 \equiv -24 \equiv 119 \pmod{143}$. Codificando a mensagem obtemos a sequência de blocos: 64 - 119 - 6 - 119 - 102 - 36 - 130 - 36 - 27 - 79 - 23 - 117 - 10.

Para decodificar a mensagem, é necessário ter dois número, n e o inverso de e em $\Phi(n)$, que é denotado por d. O par (n, d) é considerado a chave da decodificação. Sendo a um bloco de mensagem codificada, então $D(a)$ é o resultado do processo de decodificação, desta forma, a fórmula para calcular a decodificação do bloco é

$$D(a) = a^d (mod \quad n). \tag{3.7}$$

Além de n é necessário conhecer o inverso d de e módulo $\Phi(n)$ para decodificar. É possível calcular d apenas aplicando o algoritmo estendido a e e $\Phi(n)$. Neste exemplo tem-se $n = 143$ e $e = 7$, aplicando o algoritmo euclidiano estendido para calcular d. Os cálculos se tornam simples pois dividindo $\Phi(143) = 120$ por 7 obtemos $120 = 7 * 17 + 1$, onde $1 = 20 + (-17) * 7$. Logo o inverso de 7 módulo 120 é -17. Usa-se d como expoente de potências, é necessário que d seja positivo, portanto $d = 120 - 17 = 103$, que é o menor inteiro positivo congruente a -17 módulo 120. Assim, para decodificar o bloco 119, calcula-se a forma reduzida de 119^{103} módulo 143. Usando um sistema de computação algébrica, verifica-se que $119^{103} \equiv 102 \ (mod \ 143)$.

RSA está fortemente ligado à Teoria dos Números, sendo baseado em pilares como as operações de resto e fatoração por números primos. O algoritmo pode ser resumido nos passos descritos abaixo (RIVEST; SHAMIR; ADLEMAN, 1978):

1. Obter dois números primos p e q;
2. Calcular $n = pq$;
3. Calcular $\Phi(n) = (p - 1)(q - 1)$;
4. Escolher e | Máximo Divisor Comum (MDC) entre e e $\Phi(n)$ seja igual a 1, ou seja, e e $\Phi(n)$ são coprimos (primos relativos);
5. Calcular $d \ | \ de \equiv 1 \ (mod \ \Phi(n))$, ou seja, de mod $\Phi(n) = 1$;
6. Chave pública: (e, n); chave privada: (d, n);
7. Função para cifrar uma mensagem $m : C(m) = m^e \ mod \ n = c$;
8. Função para decifrar uma mensagem $c : D(c) = c^d \ mod \ n = m$;
9. $D(C(m)) = m$.

Os valores d, e estão matematicamente relacionados. Isso pode ser verificado pela propriedade inverso multiplicativo, ou seja, multiplicando o resultado de $d\ mod\ \Phi(n)$ pelo resultado de $e\ mod\ \Phi(n)$ obtém-se $\Phi(n) + 1$. Então, entende-se que $\Phi(n) + 1\ mod\ \Phi(n) = 1$.

O RSA é seguro devido ao processo custoso computacionalmente de fatorar um número grande (n) em números primos (p e q). Se b é o número de bits de n, então existem $\sqrt{(2^b - 1)}$ possibilidades a serem testadas em um eventual pior caso, o que resulta em complexidade de tempo de $O(\sqrt{(2^b)})$. Considerando que $b = 2048$, $\sqrt{2^b}$ resulta em um valor próximo a 179.10^{308}, e uma máquina é capaz de processar 1 bilhão (10^9) de tentativas por segundo, seriam necessários mais de 5.10^{291} anos (LADEIRA; RAUGUST, 2017).

Existem alguns ataques contra o sistema RSA, porém eles não são considerados muito relevantes, pois normalmente exploram as fraquezas de como o método é implementado, e não na arquitetura do algoritmo. Existem cinco técnicas possíveis para atacar o RSA, de acordo com Stallings (2015):

1. Força bruta: no ataque, testa-se todas as chaves privadas possíveis.
2. Ataques matemáticos: utilizam-se técnicas matemáticas para fatorar o produto dos números primos de uma maneira mais rápida.
3. Ataques de temporização: se aproveita do tempo que o algoritmo utiliza para decriptar uma mensagem.
4. Ataques baseados em falha de hardware: este tipo de ataque aproveita falhas de *hardware* do processador que está gerando assinaturas digitais.
5. Ataques de texto cifrado escolhido: tenta explorar as propriedades algorítmicas do RSA.

3.2 ALGORITMOS DE CRIPTOGRAFIA ESTUDADOS

A Cifra de César é a mais simples de ser implementada, mas ao mesmo tempo ela tem um nível de segurança consideravelmente baixo, como estudado, por ser uma cifra simétrica e utilizar chave secreta, existe a possibilidade de utilizar 25 chaves diferentes, então um simples algoritmo que utilize força bruta para testar todas as combinações, pode facilmente quebrar a criptografia.

A Cifra de Hill por ser baseada em conceitos matemáticos complexos de álgebra linear, é mais difícil de implementar do que a Cifra de César. Referente à segurança, possui um ponto fraco, a sua criptografia é facilmente quebrada com um ataque de texto claro conhecido, como afirma Stallings (2015).

Os algoritmos AES e RSA certamente são alguns dos algoritmos de criptografia mais utilizados na atualidade. São amplamente seguros e difíceis de quebrar. O AES é utilizado pelo

governo dos Estados Unidos por ser considerado seguro, além de que teve investimento na sua criação. Esses dois algoritmos diferem na sua arquitetura, sendo o RSA de chave pública, é tão seguro quanto o AES, tem a vantagem de não ser necessário o envio de chave por meios secundários, e baseia sua segurança na incapacidade que os processadores atuais possuem em fatorar números primos com muitos dígitos, porém, justamente isso pode vir a ser seu ponto fraco, pois com o avanço da computação quântica logo teremos máquinas capazes de quebrar esse tipo de criptografia em pouco tempo.

Quadro 3 – Algoritmos estudados

Algoritmo	**Cifra**	**Complexidade**	**Chave**	**Segurança**
Cifra de César	Simétrica	Baixa	Privada	Baixa
Cifra de Hill	Simétrica	Média	Privada	Baixa
AES	Simétrica	Alta	Privada	Alta
RSA	Assimétrica	Alta	Pública	Alta

Fonte: O Autor (2021).

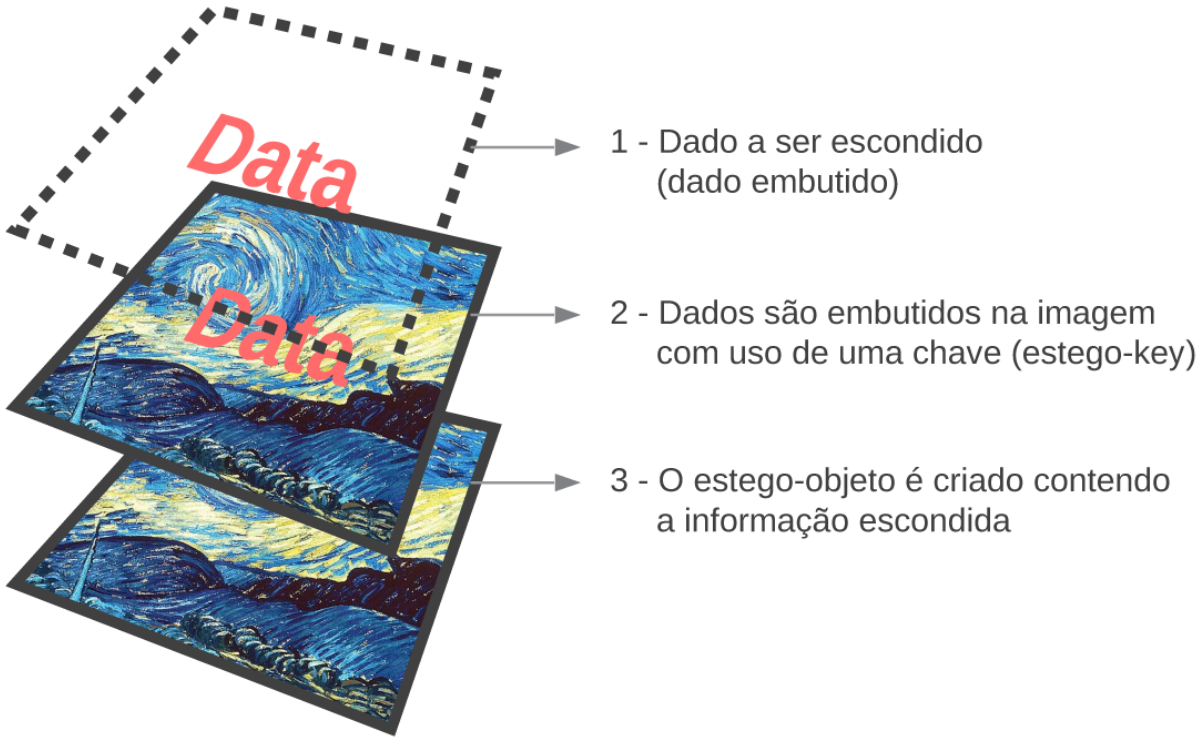
4 ESTEGANOGRAFIA

Ao falar de escrita secreta, ela pode se dividir em duas partes: criptografia e esteganografia. Enquanto a criptografia deixa evidente que existe um dado secreto sendo transmitido, a esteganografia se propõe a transmitir esse dado de forma que não seja detectado. A confidencialidade de informação é um tema muito importante e deve ser de preocupação de todos aqueles que gostariam de minimizar ataques e vazamento de informações.

> A segurança da informação é adotar controles físicos, tecnológicos e humanos personalizados, que viabilizem a redução e administração dos riscos, levando a empresa a atingir o nível de segurança adequado ao seu negócio (SEMOLA, 2002).

Cada vez mais existe um interesse no estudo da Esteganografia, principalmente por parte da indústria que utiliza marca d'água e seriação digital, dentre outras. É comum que ocorra distorção na terminologia relacionada com essa área de conhecimento. Por isso alguns conceitos devem ser estabelecidos. Abaixo são apresentados esses conceitos e a Figura 17 ilustra tais termos, como propõem Julio, Brazil & Albuquerque (2007):

Figura 17 – Formação do estego-objeto



Fonte: O Autor (2021).

1. *Embedded data*: informação que será enviada de maneira secreta. Também podemos referenciar como dado embutido.

2. *Cover-message*: é o arquivo que servirá de máscara para que o dado secreto seja enviado. Esses arquivos podem ser de áudio (*cover-audio*), de texto (*cover-text*) ou uma imagem *cover-image*).

3. *Stego-object*: quando o *embedded* data é inserido no *cover-message*, o produto do processo é chamado de stego-objeto.

4. *Stego-key*: não é obrigatório, mas em alguns casos pode-se utilizar uma chave de inserção dos dados embutidos.

5. *Fingerprinting*: uma série de números protegidos, com finalidade de provar a autoria do documento.

4.1 CAMPOS DE APLICAÇÃO

Essa sessão tem o objetivo de apresentar algumas formas de como a esteganografia pode ser aplicada no mundo real.

4.1.1 Aplicações militares

Para as agências militares, é questão de estratégia e sobrevivência realizar comunicação de forma discreta em áreas de conflito. Em algumas situações, a transmissão de conteúdo criptografado não é eficiente, uma vez que o emissor do sinal é localizado, pode ser localizado e atacado. Técnicas de esteganografia como a modulação por espalhamento de espectro são bastante utilizadas pelos militares, dificultando a detecção da transmissão pelo inimigo (PETITCOLAS; ANDERSON; KUHN, 1999).

4.1.2 Proteção na Internet

Não é incomum que em movimentações financeiras pela internet sejam utilizadas técnicas de esteganografia para suporte na comunicação. O mesmo vale para sistemas eleitorais, onde a comunicação anônima é importante para garantir a integridade da votação (ROCHA *et al.*, 2004).

4.1.3 Direitos Autorais

O criador de um documento, com a finalidade de garantir a sua autoria, pode inserir de forma oculta dentro do arquivo, uma mensagem de direitos autorais *copyright*, deste modo ao revelar o conteúdo, prove que a propriedade intelectual do documento lhe pertence. Se um terceiro atribuir a si os direitos sobre o documento, o autor pode provar o contrário, já que só ele

sabe como resgatar o conteúdo oculto. Esse processo é conhecido como marca d'água digital ou *watermarking* (PETITCOLAS; ANDERSON; KUHN, 1999).

4.1.4 Aplicações médicas

A forma de comunicação entre o médico e o exame de um paciente é conhecida por DICOM. Essa comunicação separa o exame em duas partes, a imagem em si (um raio-X por exemplo), e a segunda parte as informações do paciente. Existe a possibilidade do exame ou das informações do paciente se perderem, então o ideal é que todo o conteúdo estivesse em um único arquivo. Os estudos relacionados envolvem que a aplicação de esteganografia não prejudique a qualidade dos arquivos e por consequência o diagnóstico do médico (SAMPAIO; JACKOWSKI, 2014).

4.1.5 *Fingerprinting*

Além de ocultar *copyright*, em caso de *softwares* ou mídias de fácil reprodução no meio digital, é possível associar um número serial único para cada arquivo. Logo, caso uma mesma cópia seja distribuída, é possível identificar o usuário que está fazendo uso indevido, e possivelmente reprimi-lo. Isso é conhecido como impressão digital ou *fingerprinting* (PETITCOLAS; ANDERSON; KUHN, 1999).

4.1.6 Aplicações Ilegais

Um lado negativo da esteganografia, é que ela pode ser utilizada em aplicações fraudulentas, como em uso indevido de dados e também na espionagem industrial. Além de esconder dados ilegalmente, criminosos também podem se comunicar, a fim de que suas mensagens não sejam interceptadas. No ano de 2000, foi amplamente divulgado pela imprensa norte-americana, que terroristas estariam se comunicando secretamente com o auxílio da esteganografia (PROVOS; HONEYMAN, 2001).

4.2 TÉCNICAS ESTEGANOGRÁFICAS EM IMAGENS

Existem diversas formas diferentes de se aplicar esteganografia por meios digitais, e em diversos tipos de arquivos, tais como imagens, áudio, vídeo, documentos de texto entre outros. O estudo a seguir será focado em técnicas voltadas para esteganografia em imagens.

As imagens são mídias digitais bastante populares para aplicação de esteganografia, e podem ser utilizados diversos formatos como JPEG, GIF, PNG entre outros. A inserção dos dados na imagem, é realizado no domínio da frequência ou no domínio espacial (SULLIVAN *et al.*, 2004).

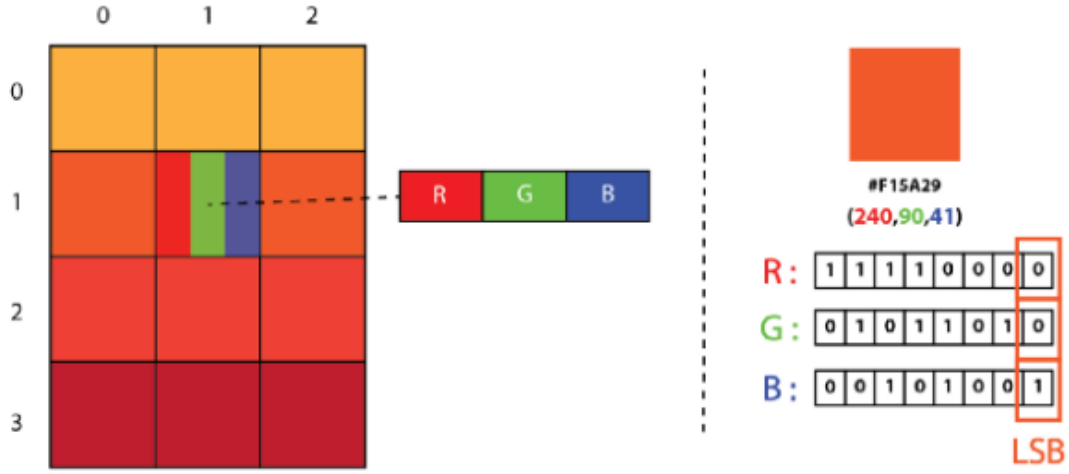
Mídias digitais como fotografias, geralmente possuem quantidade significativa de ruído, que é gerado na conversão de uma cena real para o meio digital. Desta forma, é possível esconder informações no ruído. Técnicas que utilizam esse conceito são as mais utilizadas (WAYNER, 2008).

4.2.1 Inserção no bit menos significativo

O método de inserção baseado em *Least Significant Bits* (LSB), é considerado muito comum e o mais conhecido para ocultar informação. A alteração no bit menos significativo do pixel no domínio espacial torna a alteração basicamente imperceptível (SAMPAIO; JACKOWSKI, 2014).

Essa técnica é bastante utilizada em arquivos de imagens, utilizando o ruído para esconder informação. Utiliza-se os bits menos significativos para esconder informação. A Figura 18 apresenta de forma simplificada onde a alteração é realizada. Isso é uma ótima solução, uma vez que a imagem aparentemente está inalterada ao olho humano (WAYNER, 2008).

Figura 18 – Inserção no bit menos significativo



Fonte: Jain (2021).

Muitas fotografias digitais coloridas possuem 32 bits para representar cada pixel. Desses 32 bits, existem 8 bits para guardar as quantidades de vermelho, azul e verde, no modelo RGB, por exemplo. Desta forma, são utilizados 24 bits. Se um bit de cada uma das cores for modificado para esconder informação, ao final temos que o arquivo será afetado em 10% . Cada 8 bits representa um número entre 0 e 255. O bit mais significativo equivale a 128 caso seu valor seja igual a 1, e o bit menos significativo, altera a imagem com valores entre 0.5% a 1%. É possível considerar que utilizar 10% do tamanho em bits de uma imagem e o resultado visual ser afetado em aproximadamente 1% é uma solução eficiente (WAYNER, 2008).

Esse método é vulnerável a transformações geométricas, filtros e sistemas de compressão, como o do formato JPEG, uma vez que essas transformações também aplicam alterações nos bits menos significativos, destruindo qualquer informação que esteja oculta (SAMPAIO; JACKOWSKI, 2014).

4.2.2 Bit-Plane Complexity Segmentation

O *Bit-Plane Complexity Segmentation* (BPCS) baseia-se na natureza da visão humana, onde uma pessoa não consegue perceber um rastro de informações em um padrão binário complexo, o que é característica de ruído. A ideia é dividir uma imagem, PNG por exemplo, em regiões com ruídos e sem ruídos, para ocultar os dados nas regiões que possuem ruído. Dessa forma é difícil localizar as mensagens escondidas, uma vez que as regiões complexas da imagem foram substituídas por padrões binários invisíveis aos olhos. Uma região complexa pode ser definida como um bloco que tenha uma variedade maior de informações visuais, e uma região de pouca complexidade com menor variedade de informações, por exemplo, um céu azul limpo que tenha somente a cor azul (KAWAGUCHI; EASON, 1999).

Uma imagem com valores múltiplos (P) consistindo em pixels de n bits pode ser decomposta em um conjunto de imagens binárias n. Por exemplo, se a imagem é uma imagem cinza de n bits, ela é mostrada como

$$P = (P1, P2, ..., Pn) \tag{4.1}$$

e se a imagem for representada no modelo RGB, ela é mostrada por

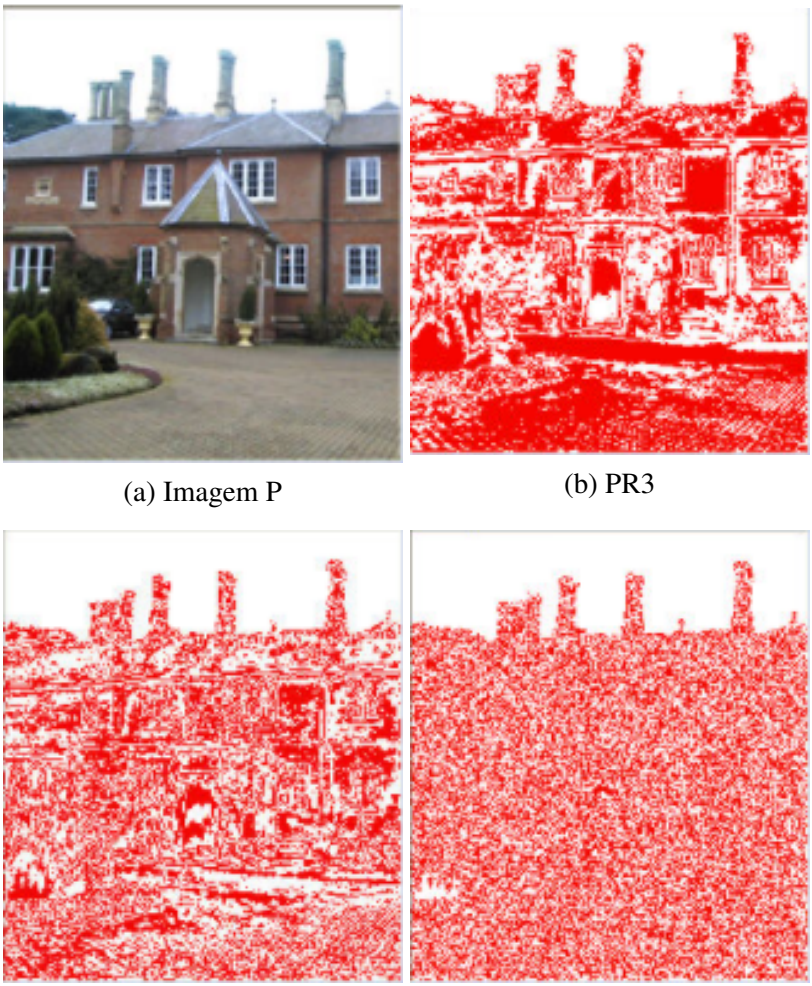
$$P = (PR1, PR2, .., PRn, PG1, PG2, .., PGn, PB1, PB2, .., PBn) \tag{4.2}$$

onde PR1, PG1, PB1 são os planos de bits mais significativos (MSB), enquanto PRn, PGn, PBn são os bits menos significativos (LSB) (KAWAGUCHI; EASON, 1999).

Os dados da imagem são representados em número por um sistema de código binário puro. No que diz respeito às camadas de bits de uma imagem, a complexidade de cada camada de bits aumenta monotonicamente do MSB (P1) para o LSB (Pn). Visualmente, a maioria das camadas LSB parecem estar em um padrão aleatório de bits. A Figura 19 ilustra os planos de bits (b) PR3, (c) PR4 e (d) PR6 de uma imagem colorida (a) P de 24 bits (KAMATA; EASON; KAWAGUCHI, 1995).

Os métodos tradicionais de esteganografia, possuem uma capacidade relativamente pequena de ocultação de dados. A capacidade é de 5 a 15% dos dados do arquivo utilizado como máscara. O BPCS é diferente das técnicas tradicionais pois possui uma grande capacidade de incorporação, podendo chegar em alguns casos a 50% de aproveitamento no caso de uma imagem de 24 bits (KAWAGUCHI, 2005).

Figura 19 – Camadas de bits.



(a) Imagem P (b) PR3

(c) PR4 (d) PR6

Fonte: Kawaguchi (2015)

4.2.3 Filtragem e Mascaramento

Essas técnicas são mais robustas que as técnicas baseadas em inserção LSB. Elas criam um stego-objeto protegido contra compressão e recorte, porém são mais fáceis de serem detectadas. As técnicas de filtragem e mascaramento funcionam de forma inversa ao LSB, uma vez que as alterações aplicadas são nos bits mais significativos dos pixels. Essas técnicas devem ser aplicadas apenas em imagens em tons de cinza, pois não são eficazes em imagens coloridas (JULIO; BRAZIL; ALBUQUERQUE, 2007).

As técnicas de filtragem e mascaramento são semelhantes à marca d'água visível, pois de acordo com uma porcentagem os valores dos pixels em áreas mascaradas são aumentados ou diminuídos. A marca fica invisível ao reduzir o incremento em um certo grau. No método de retalhos (*patchwork*), pares de remendos (*patches*) são selecionados pseudo-aleatoriamente. Os valores de pixel em cada par são aumentados por um valor constante pequeno em um remendo e diminuídos pela mesma quantia no outro (JULIO; BRAZIL; ALBUQUERQUE, 2007).

4.2.4 Algoritmos de Transformações

Um dos principais problemas do LSB é a compressão, que pode ocasionar perda de dados ocultos. Os algoritmos de transformação são mais eficientes nesse sentido, para isso utilizam: a transformada de Fourier discreta, a transformada de cosseno discreta e a transformada Z (GONZALEZ; WOODS, 2010).

Nessas técnicas, os dados são ocultos no domínio de transformação e são espalhados por toda a imagem, oferecendo maior proteção contra processamento de sinal. São amplamente utilizados em marca d'água robusta (JULIO; BRAZIL; ALBUQUERQUE, 2007).

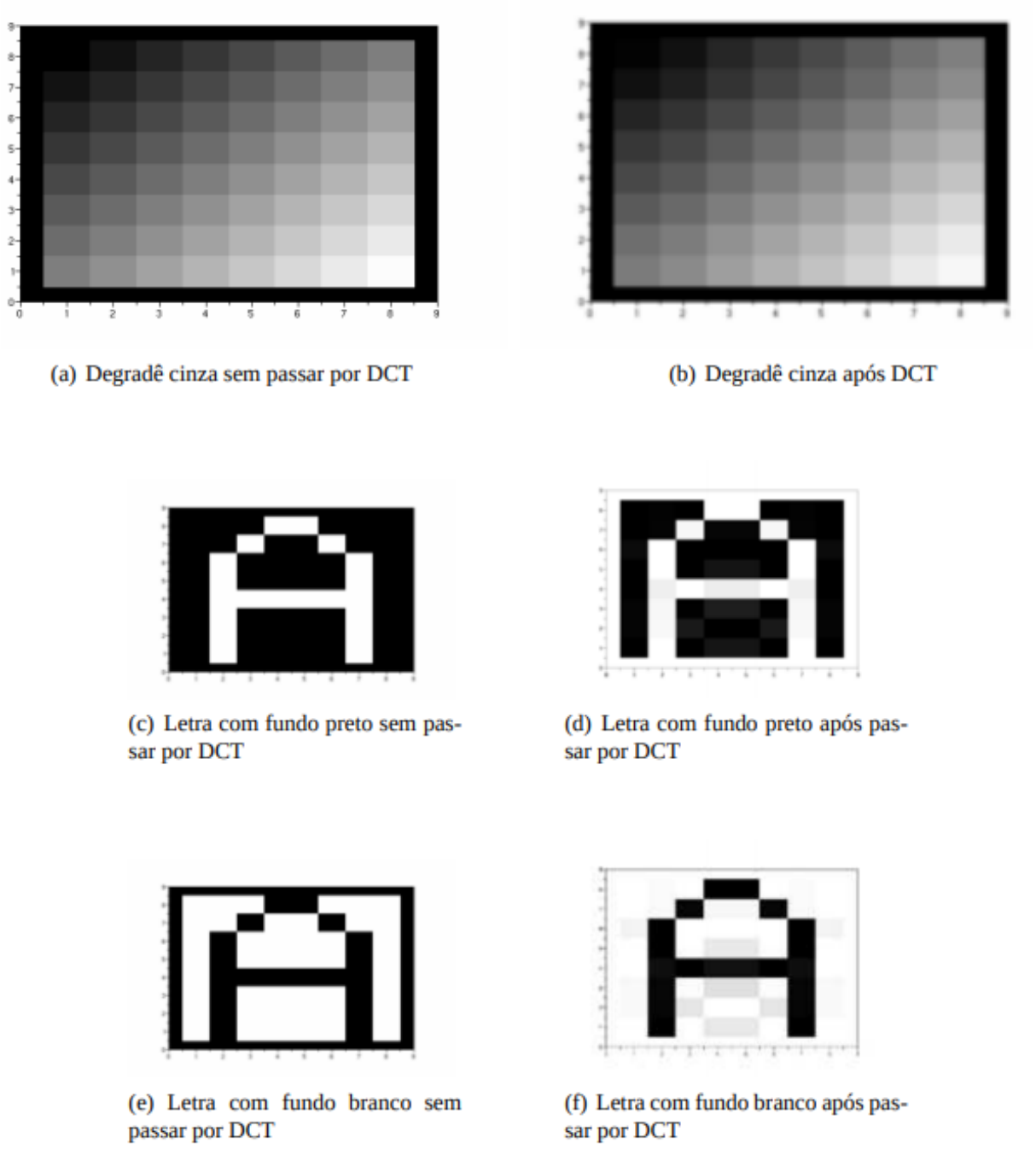
Os algoritmos baseados em transformações, de modo genérico, aplicam algum tipo de transformação em blocos de 8×8 pixels na imagem. São selecionados os coeficientes redundantes ou de menor importância em cada bloco. Após, o coeficiente escolhido é substituído por um valor pré definido para o bit 0 ou 1 (POPA, 1998).

Um método popular utilizado para ocultar informação no domínio da frequência, é a modulação do tamanho relativo de dois ou mais coeficientes *Discrete Cosine Transform* (DCT) em um bloco de imagem. Durante o processo de ocultação, é necessário dividir a cover-imagem em blocos de 8×8 pixels. Cada bloco codifica um bit de mensagem secreta. O processo inicia com a seleção de um bloco pseudo aleatório b_i, que será utilizado para codificar o enésimo bit da mensagem. Então $B_i = D\{b_i\}$ é a transformada DCT do bloco da imagem (KATZENBEISSER; PETITCOLAS, 1999).

O primeiro passo nos sistemas de compressão de imagens é identificar redundância espacial, isso é a semelhança entre um pixel e os pixels de sua vizinhança. Isso pode ser feito através da Transformada Discreta dos Cossenos (DCT) ao longo da imagem. É um processo sem perda de informação. A transformada DCT é relativamente simples de entender. Para cada dimensão de blocos a ser usados (a mais usada é de 8×8 pixels), existe uma matriz de DCT fixa. Realizar a transformação implica simplesmente em recolher os 64 pixels deste bloco da imagem, fazer o cálculo da DCT para estes valores, obtendo-se novos 64 valores, que são chamados coeficientes da DCT. Este processo é repetido para todos os blocos da imagem. A Figura 20 mostra alguns experimentos realizados aplicando essa técnica (SALOMON, 2006).

Um acordo que o remetente e o receptor da mensagem devem fazer, é decidir a localização dos coeficientes DCT. Esses coeficientes devem corresponder às funções cosseno com frequências médias, deste modo, a informação é armazenada nos bit mais significativos do pixel, portanto, a informação não é perdida no caso de trabalharmos com arquivos JPEG, devido ao seu processo de compressão que afeta os bits menos significativos dos pixels (KATZENBEISSER; PETITCOLAS, 1999).

Figura 20 – Transformação por DCT



(a) Degradê cinza sem passar por DCT

(b) Degradê cinza após DCT

(c) Letra com fundo preto sem passar por DCT

(d) Letra com fundo preto após passar por DCT

(e) Letra com fundo branco sem passar por DCT

(f) Letra com fundo branco após passar por DCT

Fonte: Julio, Brazil & Albuquerque (2007).

4.2.5 Espalhamento de Espectro

Um exemplo de espalhamento de espectro é o espalhamento de frequência, onde os dados são espalhados ao longo da imagem utilizando uma estego-chave para selecionar de forma aleatória os canais de frequência. As técnicas, geralmente são aplicadas em uma faixa de frequência larga em um nível baixo de força, inserindo os dados em pseudo ruídos gerados pelo algoritmo, isso garante que a técnica seja quase imperceptível (JULIO; BRAZIL; ALBUQUERQUE, 2007).

4.3 ESTEGANÁLISE

Assim como na criptografia existem maneiras de descobrir e atacar dados criptografados, na esteganografia também existem ataques e maneiras de detectar informações ocultas, pois todas as técnicas possuem alguma falha que possa ser explorada. Em algumas situações, basta fazer um exame mais detalhado para identificar padrões gerados para identificar se existe mensagem oculta. A pesquisa por técnicas para descobrir informação oculta é chamada de esteganálise (JULIO; BRAZIL; ALBUQUERQUE, 2007).

Para Wayner (2008) existem muitas abordagens para identificar conteúdo oculto em imagens digitais, que são divididas em três grupos: ataques aurais, estruturais e estatísticos:

- Ataques aurais e visuais: alguns ataques separam partes importantes da imagem, de forma que seja possível ao olho humano identificar qualquer anomalia. Um teste comum exibe os bits menos significativos da imagem. Esconder a informação dos olhos é o primeiro desafio. Alguns algoritmos básicos podem cometer erros graves ao realizar uma mudança de cor desproporcional na imagem.

- Ataques estruturais: o formato do arquivo que recebe informação muda de acordo com a quantidade de vezes que se insere dados dentro dele. Em alguns casos, os programas esteganográficos utilizam versões diferentes de um mesmo arquivo, assim identificando alterações estruturais e denunciando a presença de dados ocultos.

- Ataques estatísticos: os padrões de pixels nos seus bits menos significativos, muitas vezes podem revelar a existência de uma mensagem oculta no perfil estatístico. Os novos dados não têm o mesmo perfil estatístico que se espera que os dados padrão tenham.

Em alguns casos, o esteganalista quer saber somente se existe mensagem oculta, sem necessariamente ler seu conteúdo, para que desse modo possa destruir o arquivo, ou seja, quem interceptou o arquivo não lerá o conteúdo, mas o receptor também não lerá. Podemos considerar esse exemplo como esteganálise passiva, entretanto para alguns esteganalistas isso não é o suficiente, pode-se querer também ler o conteúdo e alterar o conteúdo para que o receptor receba dados inconsistentes. Nessa situação, é esteganálise ativa (CHANDRAMOULI, 2002):

Quando o esteganalista confirma a existência de uma mensagem oculta, dependendo do seu objetivo, pode seguir alguns caminhos diferentes. Abaixo são listadas as principais ações (WAYNER, 2008):

- Destruir tudo: muito se discute sobre a esteganografia não ser útil, uma vez que a sua informação pode ser totalmente destruída em um ataque, assim como a criptografia também está sujeita a esse tipo de ação.

- Adicionar novas informações: é uma forma de sobrescrever as informações originais, considerado um ataque simples que utiliza um programa semelhante para codificar novas informações.
- Alterar o formato do arquivo: ao alterar o formato do arquivo, é possível que informações sejam perdidas na conversão, pois cada formato armazena seus dados de forma diferente.
- Comprimir o arquivo: consiste em comprimir o arquivo, pois geralmente os algoritmos de compressão causam alterações em áreas com informações extras, onde possivelmente possa haver dados ocultos.

4.4 TÉCNICAS DE ESTEGANOGRAFIA ESTUDADAS

Todas as técnicas de esteganografia em imagem possuem o mesmo objetivo, camuflar informação dentro do arquivo selecionado, o que muda é a escolha da localização dentro da imagem que será armazenado o dado. Por isso, deve-se levar em conta quais fins é o objetivo da aplicação da esteganografia.

O LSB, por exemplo, é uma técnica simples, porém não é recomendado sua utilização em arquivos do formato JPEG, pois o modo de compressão desse formato, causa alterações nos bits menos significativos do pixel, local onde o LSB esconde os dados, sendo assim, ocasionado perda da informação esteganografada.

Já as técnicas baseadas em mascaramento ou em transformações, como o DCT, por exemplo, possuem maior proteção contra os algoritmos de compressão de imagens, pois fazem a ocultação dos dados nos bits mais significativos de cada imagem, baseados em cálculos diferentes para tal. O ponto fraco é que podem gerar alterações visuais na imagem, sendo perceptíveis ao olho humano, então em alguns casos, orienta-se que se utilize tais técnicas em imagens com somente tons de cinza. O Quadro 4, apresenta a lista das técnicas estudadas.

Quadro 4 – Técnicas estudadas

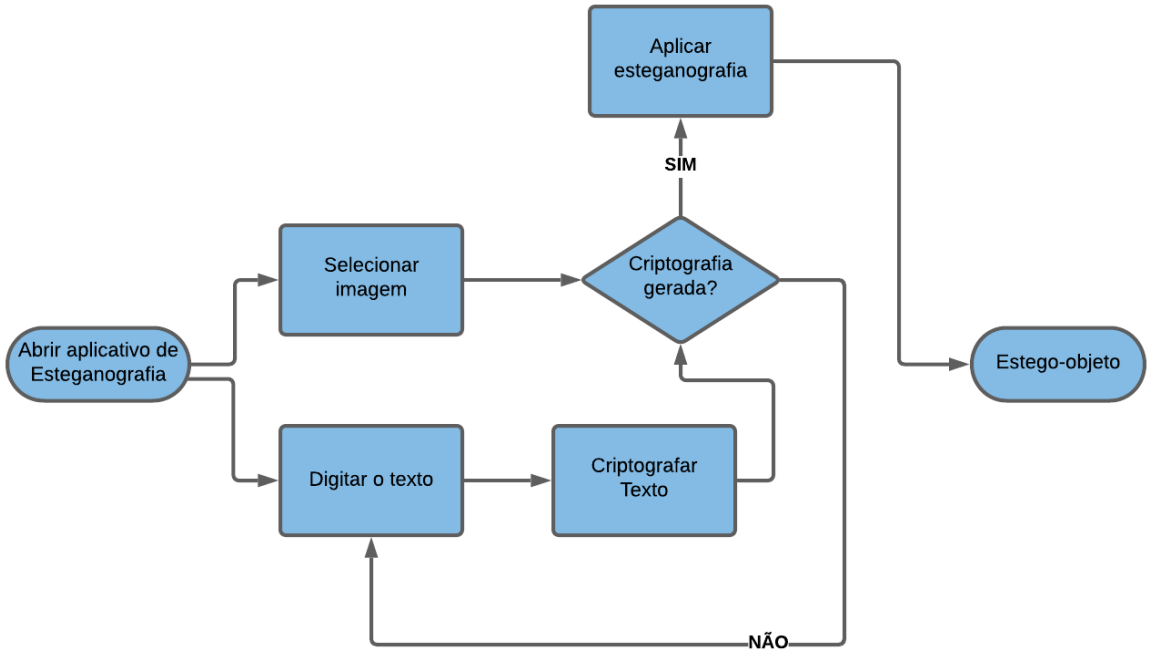
Técnicas	**Ideal para imagem**	**Imune à compressão?**
LSB	Colorida	Não
BPCS	Colorida	Não
Mascaramento	Tons de cinza	Sim
Transformação DCT	Colorida	Sim
Espalhamento de Espectro	Colorida	Sim

Fonte: O Autor (2021).

5 PROPOSTA DE DESENVOLVIMENTO

A esteganografia pode ser aliada da criptografia, levando em consideração qual tipo de aplicação se deseja desenvolver. Aplicações onde o importante é transferir informações sem que essa comunicação seja descoberta, ou simplesmente utilizada para manter informações junto aos arquivos de forma segura. É necessário também, que seja feita uma análise dos algoritmos utilizados, pois todos possuem seus pontos positivos e negativos, e geram diferentes modificações nos arquivos, assim como alguns algoritmos não são adequados a certos formatos de imagens, por gerar perda de informação, ou realizar alterações a ponto de comprometer a qualidade da imagem. Após concluir os estudos envolvendo criptografia e esteganografia, pode-se identificar que é possível criar uma aplicação unindo algumas das técnicas estudadas no presente trabalho. Para criar a aplicação, será selecionado um algoritmo de criptografia e um algoritmo de esteganografia para inserir informação criptografada em um arquivo de imagem. A Figura 21 mostra qual será o fluxo padrão da aplicação, até a geração do estego-objeto, que é o produto final, ou seja, a imagem com dados ocultos.

Figura 21 – Floxograma Aplicação (Ocultação x Criptografia)



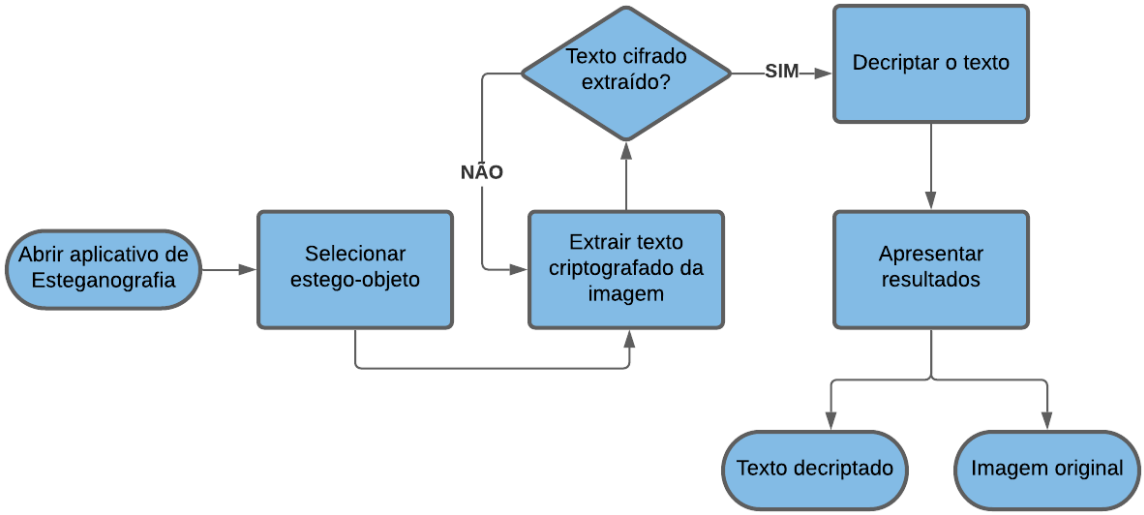
Fonte: O Autor (2021).

O fluxo da aplicação funcionará da seguinte maneira. Ao abrir a aplicação, é possível selecionar a imagem e também digitar o texto que será ocultado. A aplicação da técnica de esteganografia será aplicada somente se o processo de encriptação da mensagem for executado com sucesso, caso contrário, aguarda-se até que a criptografia seja aplicada, pois é necessário

que tenhamos os dois conjuntos prontos, a imagem e o texto cifrado, para a última etapa que gerará o objeto final.

A aplicação também fará o processo inverso. Ao receber um estego-objeto com dados criptografados, realizará o processo de extração da informação oculta presente na imagem e em seguida aplicado o processo de decriptação no texto. A Figura 22 mostra esse processo, desde selecionar o estego-objeto, até a geração separada do arquivo de imagem e do texto decifrado.

Figura 22 – Floxograma Aplicação (Extração x Decriptação)



Fonte: O Autor (2021).

É importante salientar que a utilização da esteganografia em conjunto da criptografia, adiciona uma camada a mais no quesito segurança. Como podemos perceber, primeiro é necessário descobrir que o objeto possui informação oculta, e depois, caso consiga extraí-la, ainda é necessário saber como decriptar a informação obtida.

5.1 ÁREAS DE APLICAÇÃO

Durante os estudos, identificou-se diversas áreas onde a esteganografia pode ser utilizada, desde aplicações militares até aplicações médicas voltadas para a proteção e segurança de exames. A aplicação anteriormente descrita, será utilizada com foco em imagens médicas.

Verificou-se a importância de deixar as informações do paciente junto com o arquivo do exame, para evitar equívocos no momento de relacionar exame e informações, pois pode ocorrer de trocar a informação de um exame para outro e até atribuir de forma errônea um paciente a um exame, gerando inconsistência de dados.

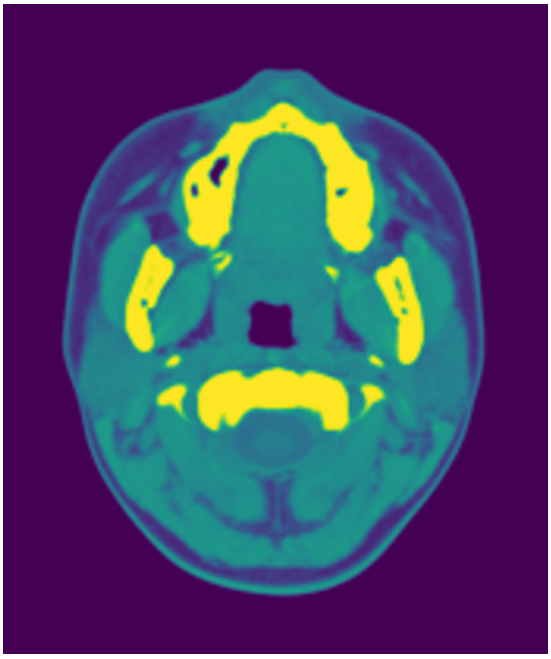
Desta forma, a aplicação realizará a inserção das informações criptografadas do paciente em seu respectivo exame, que podem ser tomografias, ressonâncias, radiografias, dentre outros.

Também será feito o processo inverso, recuperar as informações criptografadas do paciente dentro do exame.

Nesse contexto, é importante que as imagens não sofram distorções visuais, a ponto de comprometer o diagnóstico feito pelo médico, por esse motivo, é importante validar qual algoritmo de esteganografia é o ideal para aplicar neste contexto.

É necessário utilizar um *dataset* de imagens médicas para os experimentos, desta forma, foi realizada uma pesquisa na internet por bases de imagens médicas onde os arquivos correspondessem às necessidades de padrão estabelecidas para os estudos deste trabalho. Localizou-se no *site* da USP[1], uma seção aberta a visitantes para apoio aos alunos de disciplinas *online*, especificamente uma pasta com imagens radiológicas no formato DICOM. Nesse *dataset*, é possível localizar e realizar o *download* de diversos exames, tais como ressonâncias magnéticas do coração, abdômen, tórax, tomografias, radiografias, dentre outros.

Figura 23 – Tomografia computadorizada do Crânio



Fonte: USP e-disciplinas (2019).

Para escolher o algoritmo de esteganografia ideal para a aplicação proposta é necessário realizar análises. Similaridade é uma forma de medir o quanto as amostras estão relacionadas entre si. Ela é utilizada para classificação onde os objetos são rotulados com base na semelhança de seus dados (HARMOUCH, 2021).

Ao selecionar uma medida de similaridade para utilizar, é necessário levar em consideração qual objeto está sendo analisado. Dados mútuos e dados mútuos normalizados são as medidas mais utilizadas para verificar semelhança entre imagens multimodais. A correlação

[1] <https://edisciplinas.usp.br/mod/folder/view.php?id=2615128>

cruzada, soma das diferenças quadradas e proporção uniforme da imagem, são os métodos mais utilizados para imagens da mesma modalidade. Serão aplicadas duas abordagens para avaliar os resultados, qualitativa e quantitativa (ULYSSES; CONCI, 2010).

5.1.1 Imagens médicas DICOM

Com o avanço da tecnologia, inclusive na área médica, instituições como *American College of Radiology* e *National Electrical Manufacturers Association*, identificaram a necessidade da criação de um padrão para transferência de imagens e informações relacionadas com exames, isso facilitaria a integração destes arquivos com outras informações hospitalares. Em 1985 foi publicada a primeira versão do padrão que ficou conhecido como DICOM (SAMPAIO; JACKOWSKI, 2014).

DICOM não é apenas um formato de arquivo convencional, é um protocolo que abrange a transferência e exibição de dados. Todos os dados do mundo real como pacientes, estudos, equipamentos médicos entre outros, são vistos pelo DICOM como objetos com suas respectivas propriedades e atributos. Esses atributos são padronizados de acordo com o DICOM *Information Object Definitions* (IODs). Os IODs são como coleções de atributos, descrevendo cada objeto de dado específico. Eles podem ser descritos por nome, número de registro médico (ID), sexo, idade, peso, dentre outros (PIANYKH, 2008).

Esses arquivos possuem dois componentes: um cabeçalho, onde se encontram as IODs, e uma matriz em tons de cinza, que representa a intensidade da imagem. Vale ressaltar, que o padrão DICOM não oferece segurança na transmissão nem na proteção de dados. Desta forma, é possível interceptar uma imagem e alterá-la sem que se possa identificar o uso indevido (SAMPAIO; JACKOWSKI, 2014).

Pode referir-se às IODs como *tags*, assim como apresentado no Quadro 5, que exemplifica algumas informações do padrão.

Quadro 5 – *Tags* relacionadas a informações do paciente

Tags	**Descrição**	**Tipo**	**Valor**
0010-0010	PatientName	PN	Diego Alcoforado Aires
0010-0020	PatientID	LO	010101B4
0010-0030	PatientBirthDate	DA	20000318
0010-0040	PatientSex	CS	M
0010-1000	OtherPatientIDs	LO	1987365293747
0010-1030	PatientWeight	DS	70
0010-21C0	PregnancyStatus	US	4

Fonte: O Autor (2021).

5.1.2 Análise qualitativa

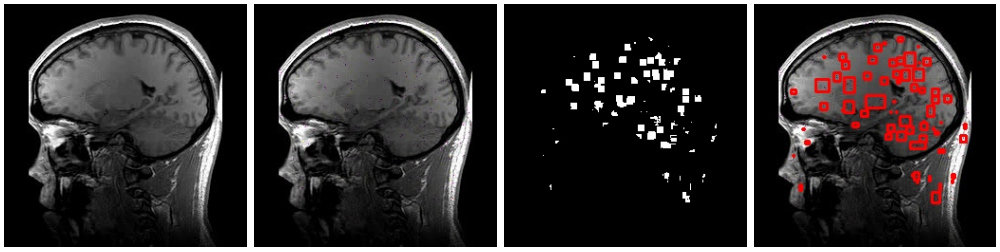
Será avaliado o quanto o método causou alterações que podem ser perceptíveis ao olho humano, pois como já foi mencionado, é muito importante que visualmente a imagem não sofra tais alterações a ponto de comprometer o diagnóstico médico, para isso, será utilizada uma função para calcular a diferença entre uma imagem e outra.

A função utilizada será a *Structural Similarity Index Measure* (SSIM), pois ela atua na percepção de mudanças estruturais da imagem. Ela compara a imagem original e a imagem que passou por alguma transformação, tem como retorno um *score* que está no intervalo entre 1 e 0, sendo 1 a similaridade perfeita, o *score* será utilizado na análise quantitativa, a SSIM será utilizada somente para destacar as diferenças visualmente. O valor 0 indica que a imagem não possui similaridade estrutural. Esse índice é implementado no domínio espacial para identificar a ideia de similaridade estrutural. Qualquer distorção no espaço da imagem pode ser interpretado como a adição de um vetor de distorção ao vetor central que representa a imagem de referência original (WANG *et al.*, 2004).

A função SSIM será utilizada a partir da biblioteca *skimage*, que disponibiliza tanto o cálculo do *score* quanto um retorno com a imagem diferença entre as imagens comparadas. Através desse retorno, pode-se trabalhar melhor a apresentação dos resultados, aplicando algumas transformações visuais para tornar a análise mais produtiva, como por exemplo detectar as diferenças e contornar a área, para que o olho busque diretamente esses lugares.

A Figura 24 representa esse processo, onde a imagem (a) representa uma MRI normal, para fins de simulação, foi inserido o ruído *Salt and Pepper*[2] na imagem (b) para ocasionar mudanças em relação à original. A imagem (c) é a diferença entre as imagens (a) e (b), identificando as alterações possíveis de serem observadas, e a imagem (d) faz um paralelo entre a imagem original e as alterações sofridas.

Figura 24 – Análise de diferença de uma imagem após receber ruído.



(a) Original. (b) Ruído. (c) Diferença. (d) Destaque.

Fonte: O Autor (2021).

[2] Causado por erros na transmissão de dados. Os pixels corrompidos são alterados para o valor máximo, ou causa uma diferença brusca entre de tons entre um pixel e seus vizinhos

5.1.3 Análise quantitativa

Quando fala-se sobre análise quantitativa, identifica-se a busca por comparar dois objetos com base em dados e informações sólidas. Cada grupo de objetos possuem métodos, técnicas e funções específicas para realizar o levantamento de dados necessários para proceder com algum tipo de comparação. Por isso é importante identificar os métodos adequados para relacionar dados relevantes e que façam sentido aos objetos estudados.

Avalia-se a similaridade entre a imagem original e a imagem final, que passou pelo processo esteganográfico. Cinco critérios serão utilizados para avaliar as imagens antes e após o processo de criptografia e esteganografia, que segundo Ulysses & Conci (2010) são considerados bons métodos e que geram resultados satisfatórios para comparação de imagens médicas que passaram por algum tipo de transformação digital. Para isso, considera-se A a matriz que corresponde à imagem original e B a matriz que representa a imagem após passar pelos processos citados.

- *Sum of Square Differences* (SSD): o valor ideal de similaridade para essa medida é zero. Correspondências ruins entre A e B resultam em valores grandes.

$$SSD = \sum_{x_A \in \Omega^T{}_{A,B}} |A(x_A) - B^T(x_A)|^2 \tag{5.1}$$

- *Sum of Absolute Differences* (SAD): esse método é muito sensível a um pequeno número de pixels apresentando diferenças de intensidade muito grandes entre imagens A e B. Menores valores para essa métrica, representam maior similaridade entre os objetos comparados.

$$SAD = \sum_{i,j} |A_{ij} - B_{ij}| \tag{5.2}$$

- *Max of Absolute Differences* (MAD): utiliza-se valores absolutos máximos das diferenças entre pixels na imagem original A e o correspondente pixel na imagem B.

$$MAD = max|A_{ij} - B_{ij}| \tag{5.3}$$

Para o olho humano, é fácil dizer o quão semelhantes são em qualidade duas imagens. Porém, se for necessário quantificar essa diferença, necessita-se de expressões matemáticas para auxiliar. Dentro da análise quantitativa, existem algumas métricas para cálculo de erro que são importantes indicadores dentro do processamento de imagem, dentre elas podemos citar as funções *Mean Squared Error* (MSE) e a *Root Mean Squared Error* (RMSE). Juntamente com a função SSIM, essas métricas avaliam a similaridade entre dois objetos, neste caso, duas imagens (RAVAL, 2021).

- MSE: utilizado para medir a distorção entre a imagem original e o estego-objeto. M e N são o número de linhas e coluna na imagem de entrada. Valores próximos a 0 são ideais.

$$MSE = \frac{1}{NM} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2 \tag{5.4}$$

- RMSE: é uma medida frequentemente usada para identificar diferenças entre os valores previstos por um modelo. Valores próximos a 0 são ideais.

$$RMSE = \sqrt{\frac{1}{NM} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2} \tag{5.5}$$

5.2 SELEÇÃO DAS TÉCNICAS E ALGORITMOS

Alguns dos algoritmos de esteganografia estudados serão testados de acordo com as métricas estabelecidas. Serão utilizados o LSB, inserção no bit menos significativo, o algoritmo de transformação por DCT, e o de identificação de áreas complexas BPCS. O LSB foi selecionado por não ser tão complexo de implementar, e também por ser bastante utilizado em aplicações de esteganografia. O algoritmo de transformação por DCT também é bastante utilizado, porém possui maior complexidade de implementação, mas também apresenta bons resultados, principalmente por não ser afetado pelos métodos de compressão de alguns formatos de imagens, como o JPEG por exemplo. Para criptografia, é utilizado o método RSA de chave pública, pois é o algoritmo mais popular de cifra assimétrica, dessa maneira evita-se que além do arquivo de imagem que será transmitido por algum meio, uma chave privada também tenha que ser transmitida, aumentando o nível de segurança da aplicação.

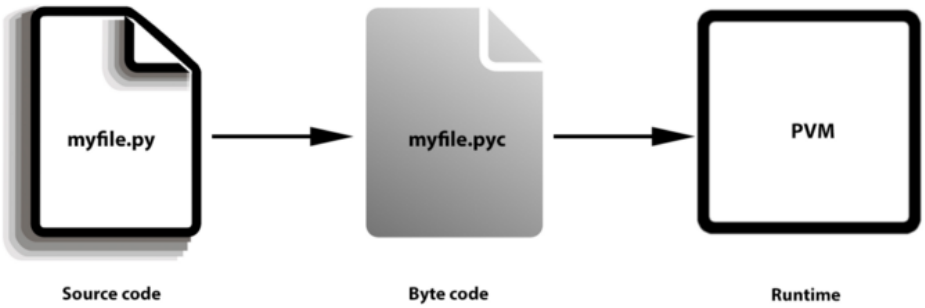
5.3 LINGUAGEM DE PROGRAMAÇÃO

Para implementar a aplicação de criptografia e esteganografia, e também para realizar as análises, será utilizada a linguagem de programação *Python*. É uma linguagem de alto nível de tipagem dinâmica e forte. Foi criada por Guido van Rossum em 1991, e atualmente possui seu desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation*.

Sempre que um programa *Python* é chamado, será checado se existe a sua versão compilada com extensão .pyc. Será carregado o *bytecode*, que vai acelerar o *script*. Se não existir a versão em *bytecode*, será criado esse arquivo antes de iniciar a execução do programa. Isso significa a execução de um código *bytecode* na *Python Virtual Machine*.

Essa linguagem possui um conjunto de bibliotecas que facilitam a manipulação de imagens e matrizes (formato como as imagens são representadas computacionalmente), como

Figura 25 – Linguagem Python



Fonte: Kaminskyi (2019).

numpy e *opencv*. A interface gráfica também será desenvolvida nessa linguagem, para isso, será utilizada a biblioteca *Kivy*. Essa interface será responsável por gerar o arquivo esteganografado e apresentar a imagem original e o estego-objeto para comparação, assim como receber o texto a ser criptografado.

- *Numpy*: é uma biblioteca utilizada principalmente para realizar cálculos em *Arrays* Multidimensionais. Fornece um grande conjunto de funções e operações que auxiliam a realizar cálculos numéricos. Grande parte do seu código foi escrita em linguagem C, o que torna seus métodos mais eficientes, isso em um módulo *Python* simples de utilizar.
- *OpenCV*: lançada em 1999 pela Intel, é uma biblioteca multiplataforma livre ao uso acadêmico, muito utilizada para a manipulação de imagens que possui compatibilidade com a linguagem *Python*.
- *Kivy*: é uma biblioteca *open source* escrita em *Python* para o desenvolvimento de aplicativos *mobile* ou *desktop*. As aplicações podem ser executadas em *Android*, *iOS*, *Linux*, *OS X* e *Windows*.
- *Pydicom*: Através desta biblioteca, é possível manipular todo o conjunto de informações de arquivo *.dcm*, as *tags* e a matriz de pixels que compõe a imagem.

6 DESENVOLVIMENTO

Este capítulo busca explicar o desenvolvimento da proposta criada no Capítulo 5, apresentando os passos seguidos, desde a seleção das imagens, as modificações nos arquivos *.dcm* para se adequarem ao estudo proposto, a geração das métricas de comparação e o cálculo de erro entre a imagem original e o estego-objeto. A proposta indica que os algoritmos de esteganografia seriam testados para identificar qual o melhor para a aplicação de imagens médicas, no capítulo será apresentado como as métricas foram calculadas, para justificar a escolha do algoritmo anexado ao protótipo de esteganografia e criptografia. Referente à parte de criptografia, é abordado o processo de como por meio da biblioteca *Crypto* desenvolvida em *python*, podemos utilizar o modelo de chave pública com o algoritmo criptográfico RSA.

6.1 ADEQUAÇÃO DO MODELO DE CORES

As imagens do *dataset* escolhido, inicialmente estavam representadas em escala de cinza, o que para alguns algoritmos de esteganografia, como o LSB, não ocorreria nenhum problema em relação a inserção de dados, pois a premissa é a alteração do bit menos significativo do pixel. Porém, o algoritmo BPCS utiliza diversas camadas de pixels do modelo RGB para esconder fragmentos do dado a ser embutido na imagem. Por este motivo surgiu a necessidade de alterar o modelo de escala de cinza para RGB.

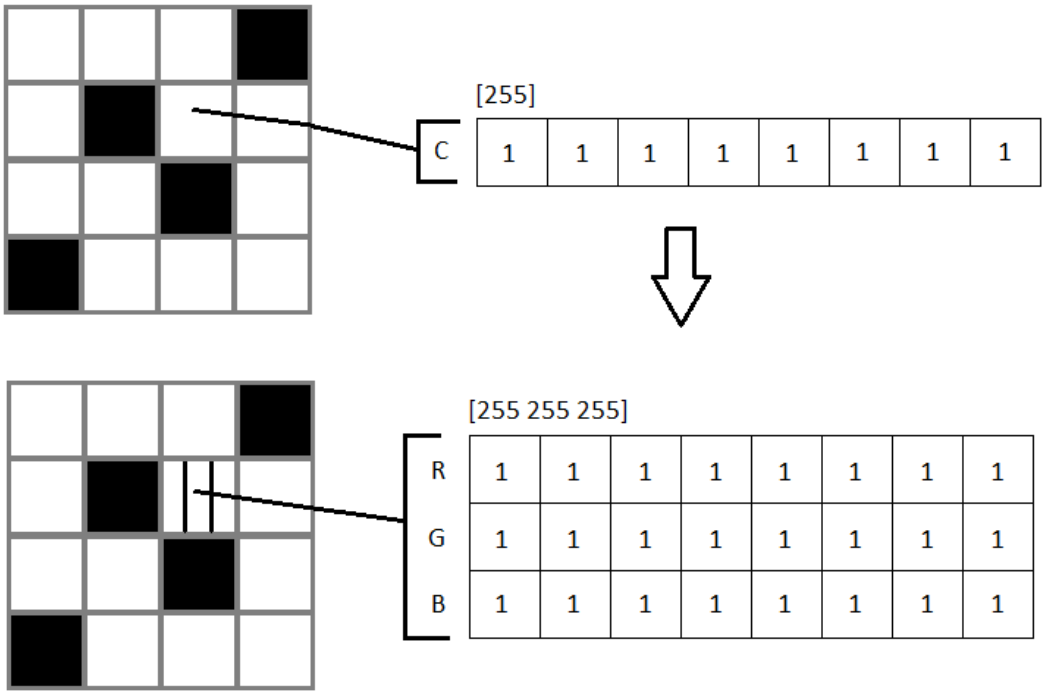
O formato de arquivo DICOM, como já foi explicado, separa o arquivo entre o *array* de *bytes* que forma a imagem e um cabeçalho que possui diversas informação do paciente, do exame, do hospital dentre outras. Além disso, no cabeçalho são armazenados algumas informações do próprio arquivo, como por exemplo o seu tamanho.

Para ler um arquivo de extensão *.dcm*, é utilizada a biblioteca escrita na linguagem *python*, a *pydicom*. Abrindo a imagem e separando cabeçalho do *array* de pixels, altera-se algumas *tags* para que o arquivo entenda que a escala de cores mudou. Existem duas *tags* principais que devem ser alteradas para que a conversão de escala de cinza para RGB tenha êxito e qualquer *software* possa interpretar o arquivo corretamente:

- *PhotometricInterpretation*: inicialmente possui o valor *MONOCHROME2* e deve ser alterado para *RGB*. Indica em qual modelo de cores a imagem está representada.
- *SamplesPerPixel*: inicialmente possui o valor *1* e deve ser alterado para *3*. Indica em quantas camadas o pixel é representado.

Após alterar as *tags* do arquivo, é necessário manipular o *array* de *bytes*. No modo escala de cinza, cada pixel é representado por apenas 8 bits, que indica a sua luminância, os valores podem variar de 0 a 255, 0 sendo preto absoluto e 255 o branco absoluto. Para alterar o modelo para RGB, em cada pixel foram adicionados mais dois *arrays* com o mesmo valor do original para preservar a mesma cor. Deste modo o pixel passa a ser representado por uma tripla, um *array* de 8 bits para cada componente de cor, a Figura 26 apresenta essa transformação.

Figura 26 – Escala de Cinza para RGB



Fonte: O Autor (2021).

Quando os dois passos descritos são finalizados, é necessário reunir as informações, *array* de *bytes* modificado e o cabeçalho atualizado, no objeto que está na memória e salvá-lo novamente em um arquivo de extensão *.dcm*.

Para automatizar o processo, foi escrito um *script* que percorre determinado diretório em busca de arquivos com extensão DICOM. Quando encontrado, é realizada a leitura através do biblioteca *pydicom*, extrai-se a matriz de pixels e a transforma em um *array*. Através dos *bytes* e o auxílio da biblioteca *numpy*, realizam-se as transformações para que cada pixel seja representado em RGB. Com os dados da matriz separados e transformados em uma outra variável, atualiza-se o cabeçalho do arquivo e o campo da matriz de pixels com os *bytes* alterados. As novas imagens geradas, são armazenadas em um diretório diferente, pois serão utilizadas nos algoritmos de esteganografia, para as avaliações de semelhança.

6.2 ALGORITMOS DE ESTEGANOGRAFIA

Como parâmetro de entrada de todos eles, é recebido uma imagem no formato *png* e um texto a ser escondido dentro da imagem. Foram necessárias algumas alterações nos algoritmos para que aceitassem como entrada imagens no formato *dcm*.

Quando uma imagem *dcm* é aberta, é necessário separar o cabeçalho da matriz de pixels, que é onde as transformações são efetivamente realizadas. Para isso, é utilizada uma biblioteca construída em *Python* para trabalhar com esses arquivos. O Algoritmo 1 exemplifica o uso da função de abertura no código, separando o *array* de pixels do restante do arquivo. A variável *ds* recebe todo o conteúdo do arquivo *dcm* e a partir dela é possível extrair o *pixel_array* para a variável *image_array*, que é utilizada nas transformações que a imagem receberá, de acordo com o algoritmo de esteganografia que estiver sendo executado.

Algoritmo 1 – Abertura de arquivo *.dcm*

```
1 import pydicom as dicom
2
3 ds = dicom.dcmread('diretorio/imagem.dcm')
4 image_array = ds.pixel_array
```

Fonte: O Autor (2021)

Outro processo comum aos algoritmos estudados, antes que a lógica de inserção seja efetivamente executada, é a transformação da mensagem em *array* de bits, que inicialmente é representada por uma *string*. A mensagem é transformada em um *array* de *bytes*, onde cada caractere é transformado no seu valor inteiro correspondente na tabela ASCII, então esse valor inteiro é convertido para um valor binário de 8 bits.

Para o algoritmo LSB, após os processos comuns, inicia-se de forma sequencial a inserção da informação no bit menos significativo de cada componente RGB, isso é feito transformando a matriz de pixels em um *array* unidimensional, percorrendo cada *byte* e alterando seu último bit.

É definido um conjunto de caracteres para delimitar o fim da mensagem, neste caso, foi definido o conjunto ";;;", isso é importante no processo de decodificação, pois durante esse processo, o algoritmo busca os bits menos significativos de todos *bytes*, ou seja, não interrompe a busca quando extrai toda a mensagem, desta forma, quando a mensagem é convertida de volta em *string* é retornado somente o texto encontrado até o delimitador.

Para finalizar, a matriz de pixels é atualizada no cabeçalho do arquivo, e salvo em outro diretório com nome diferente. O trecho de código responsável por isso, está representado no Algoritmo 2.

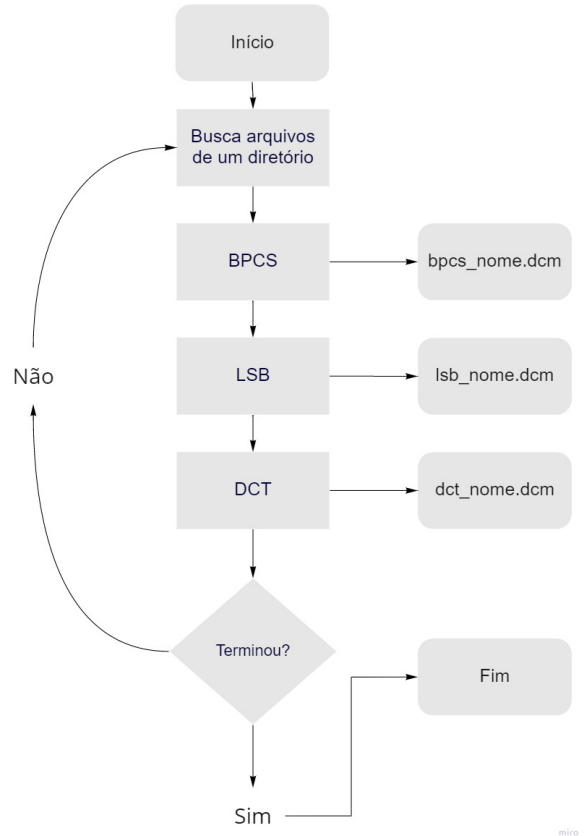
Algoritmo 2 – Salvar arquivo *.dcm*

```
1 import pydicom as dicom
2
3 image = encode(image_array, message)
4 ds.PixelData = image.tobytes()
5
6 ds.save_as('diretorio/imagem.dcm')
```

Fonte: O Autor (2021)

Para criar os arquivos que foram utilizados na análise de dados sobre os algoritmos estudados, foi desenvolvido um *script* que percorre todos os arquivos de um determinado diretório, seleciona aqueles com extensão *.dcm* e em seguida aplica cada um dos métodos de esteganografia sobre a imagem, embutindo a mesma mensagem[1] com 261 caracteres, dessa forma, a mesma imagem possui a sua correspondente alterada por LSB, DCT e BPCS, os arquivos diferenciam-se pelo nome, onde é concatenado ao título da imagem qual algoritmo a processou.

Figura 27 – Fluxograma Script



Fonte: O Autor (2021).

[1] #0010-0010#PatientName#PN#Diego Alcoforado Aires#;#0010-0020#PatientID#LO#010101B4#;#0010-0030#PatientBirthDate#DA#20000318#;#0010-0040#PatientSex#CS#M#;#0010-1000#OtherPatientIDs#LO#1987365293747#;#0010-1030#PatientWeight#DS#70#;#0010-21C0#PregnancyStatus#US#4#;

6.3 MÉTRICAS

Pode-se dividir a criação das métricas em duas situações específicas. Para programar as fórmulas SSD, SAD e MAD, foi construído um *script* com o auxílio da biblioteca *numpy* para percorrer as posições da imagem A e B, aplicando os cálculos aos pixels correspondentes. Nas funções do Algoritmo 3, cada uma delas recebe A como a imagem original e B como a imagem alterada. Após, é feito o cálculo de similaridade baseado na fórmula de cada métrica.

Algoritmo 3 – Funções de cálculo SSD, SAD e MAD

```
1  import numpy as np
2
3  def ssd(A, B):
4      s = np.sum(np.abs(A-B)**2)
5      return s
6
7  def sad(A, B):
8      s = np.sum(np.abs(A-B))
9      return s
10
11 def mad(A, B):
12     s = np.max(np.abs(np.array(A)- np.array(B)))
13     return s
```

Fonte: O Autor (2021)

Para as métricas SSIM, MSE e RMSE foram utilizadas duas bibliotecas que possuem funções para realizar os cálculos de comparação entre imagens de acordo com cada fórmula.

A SSIM calcula a similaridade estrutural entre duas imagens, tal cálculo é realizado através da função *ssim* da biblioteca *skimage*. No Algoritmo 4 é possível identificar como o código é utilizado. Essa função, além de trazer dados para a análise quantitativa, retorna a imagem diferença entre A e B que pode ser aproveitada na análise qualitativa.

Algoritmo 4 – Cálculo da métrica SSIM

```
1  from skimage.metrics import structural_similarity as ssim
2
3  def score_ssim(A, B):
4      (score, diff) = ssim(A, B, full=True)
5      return score, diff
```

Fonte: O Autor (2021)

Para MSE e RMSE foram utilizadas as funções respectivas através da biblioteca *sewar*. Elas funcionam de maneira simples, sendo apenas necessário utilizar como parâmetros de entrada as imagens A e B como exemplifica o Algoritmo 5. O retorno das funções é o resultado baseado em cada uma das fórmulas.

Algoritmo 5 – Cálculo das métricas MSE e RMSE

```
1 from sewar.full_ref import mse, rmse
2
3 def score_mse(A, B):
4     s = mse(A, B)
5     return s
6
7 def score_rmse(A, B):
8     s = rmse(A, B)
9     return s
```

Fonte: O Autor (2021)

Essas funções foram agrupadas em um mesmo *script*, que itera sobre o conjunto de imagens, comparando a original com aquela que foi alterada por cada um dos algoritmos de esteganografia. Os resultados de cada iteração são armazenados separadamente para cada imagem de acordo com a métrica. Uma variável por cada métrica é responsável por somar os resultados correspondentes para que ao final do processo, seja realizada uma média dos valores gerados pelo cálculo de cada métrica.

6.4 CRIPTOGRAFIA COM RSA

O algoritmo RSA foi escolhido sem a necessidade de testar outras opções, pois se trata de um algoritmo bastante utilizado atualmente em diversas aplicações, aliando uma arquitetura moderna ao utilizar o conceito de chave pública e um alto nível de segurança, ou seja, tornando difícil que mensagens criptografadas através deste algoritmo sejam decriptadas.

Esse algoritmo pode ser utilizado para criptografar mensagens e também utilizado em assinatura digital. O ponto negativo é que o processo de decriptar uma mensagem é consideravelmente mais lento que outros algoritmos de criptografia.

Seguindo a proposta, optou-se por utilizar uma biblioteca *Python* para iniciar os testes com o algoritmo. Essa biblioteca também foi escolhida pois é possível salvar arquivos com o conteúdo das chaves públicas e privadas geradas, isso é importante pois se encaixa no fluxo estabelecido ao protótipo ao encriptar e decriptar mensagens. Antes que o algoritmo fosse implementado propriamente ao projeto, foram realizados alguns testes, para validar o seu funcionamento.

É possível criar um par de chaves, a pública e a privada, através da função *RSA.generate*. O parâmetro de entrada da função é o comprimento da chave, que deve ser 1024, 2048 ou 3072. Gerando o par de chaves, é possível extrair a chave pública e em seguida, utilizando-a é realizado o processo de encriptação da mensagem escolhida. Para a criptografia, é escolhida uma cifra que represente o RSA, nesta situação foi utilizada a *PKCS1_OAEP*, que através da

função *new* passando como parâmetro a chave pública, é realizada a criptografia do texto. É possível visualizar o processo no Algoritmo 6.

Algoritmo 6 – RSA - Criptografa

```
1  from Crypto.PublicKey import RSA
2  from Crypto.Cipher import PKCS1_OAEP
3
4  # Gera as chaves
5  keyPair = RSA.generate(3072)
6
7  pubKey = keyPair.publickey()
8  privKeyPEM = keyPair.exportKey()
9
10 # Encripta
11 msg = 'mensagem secreta'
12
13 encryptor = PKCS1_OAEP.new(pubKey)
14 encrypted = encryptor.encrypt(msg)
```

Fonte: O Autor (2021)

Para realizar o processo inverso, de decriptação do texto é necessário obter a mensagem cifrada e a chave privada que foi gerada junto da chave pública. Em termos de código, o processo pode ser verificado no Algoritmo 7.

Algoritmo 7 – RSA - Descriptografa

```
1  from Crypto.PublicKey import RSA
2  from Crypto.Cipher import PKCS1_OAEP
3
4  # Decripta
5  decryptor = PKCS1_OAEP.new(keyPair)
6  decrypted = decryptor.decrypt(encrypted)
```

Fonte: O Autor (2021)

6.5 RELAÇÕES DE TAMANHO E CAPACIDADE DE ARMAZENAMENTO

Para entender as relações de tamanho da imagem processada e da quantidade de informação que é possível embutir dentro dela, pode-se utilizar como exemplo o arquivo *coracao_01.dcm*, que das imagens do *dataset* é uma das com as menores dimensões de altura e largura.

As dimensões da imagem utilizada para o exemplo são de $320 \times 320 \times 3$, isso significa que a matriz que a representa possui 320 pixels de altura e 320 pixels de largura, e ao final, cada posição da matriz é multiplicada por três, mostrando que cada pixel é composto por um

array de 3 *bytes*, representando o modelo de cores RGB. Isso significa que o tamanho total da imagem é de 307200 *bytes*.

Porém, para os cálculos de capacidade de armazenamento de conteúdo esteganografado, é necessário identificar a quantidade total de bits da imagem utilizada. Cada *byte* é representado por 8 bits, então o cálculo realizado é 307200 *bytes* $\times$ 8 bits $=$ 2457600 bits.

É importante obter essa relação pois cada caractere do texto que é embutido na imagem, é representado em 8 bits, por exemplo, utilizando o algoritmo LSB para ocultar o caractere "A" dentro da imagem, o que será embutido nos bits menos significativos será a informação 01000001, que é o valor binário para 65 representando o caractere em questão na tabela ASCII.

Identificando essas dimensões, é possível prever a quantidade máxima de caracteres que cada algoritmo pode embutir no objeto selecionado. Neste caso, pode-se dizer que para o exemplo apresentado, alterando 1 bit de cada componente RGB totalizando 3 por pixel, tem-se 12.5% de alteração em relação ao total de bits que formam a imagem, isso significa que é possível modificar 307200 bits, convertendo para *bytes* obtém-se 38400 caracteres. Esses valores são apresentados no Quadro 6.

Quadro 6 – Capacidade de armazenamento em LSB

LSB				
Arquivo	***Shape***	***Bytes***	**Bits**	**Caracteres**
coracao_01.dcm	320x320x3	307200	2457600	38400
cranio_01.dcm	512x512x3	786432	6291456	98304

Fonte: O Autor (2021).

Segundo Kawaguchi (2005), o algoritmo BPCS é capaz de utilizar 50% do tamanho em bits da imagem para embutir informação. Utilizando esse valor como referência, pode-se estimar a quantidade de caracteres que é possível ocultar com essa técnica, os valores são apresentados em Quadro 7.

Quadro 7 – Capacidade de armazenamento em BPCS

BPCS				
Arquivo	***Shape***	***Bytes***	**Bits**	**Caracteres**
coracao_01.dcm	320x320x3	307200	2457600	153600
cranio_01.dcm	512x512x3	786432	6291456	393216

Fonte: O Autor (2021).

Para o algoritmo DCT, a capacidade de armazenamento é a mesma que do algoritmo LSB, por essa razão não existe necessidade de uma nova tabela.

7 ANÁLISE DOS DADOS

Este capítulo destina-se a apresentar e explicar os dados coletados em função das métricas de similaridade selecionadas. A parte quantitativa apresenta os gráficos e tabelas, enquanto a qualitativa apresenta a comparação visual entre a imagem e o estego-objeto gerado por cada um dos algoritmos de esteganografia.

7.1 ANÁLISE QUANTITATIVA

Para analisar os algoritmos, baseados nas métricas de similaridade definidas (SSD, SAD, MAD, SSIM, MSE e RMSE), foi utilizado um conjunto de dez imagens diferentes, cada uma delas foi processada pelos três algoritmos de esteganografia e embutindo a mesma mensagem[1]. O Quadro 8 tem o objetivo de demonstrar a média dos resultados obtidos para cada uma das métricas. Vale ressaltar que o quadro não é comparativo entre as métricas, todas medem similaridade, mas a faixa de valores obtidos é diferente para cada uma delas, isso é de acordo com a fórmula utilizada para o cálculo.

Quadro 8 – Média das métricas para um conjunto de 10 imagens

Métrica	**LSB**	**DCT**	**BPCS**
SSD	313.0	34394437.4	1781104.9
SAD	59571.2	45445600.6	6431834.1
MAD	229.6	255	255
SSIM	0.999	0.537	0.984
MSE	0.000	374.267	3.261
RMSE	0.000	6.117	0.571

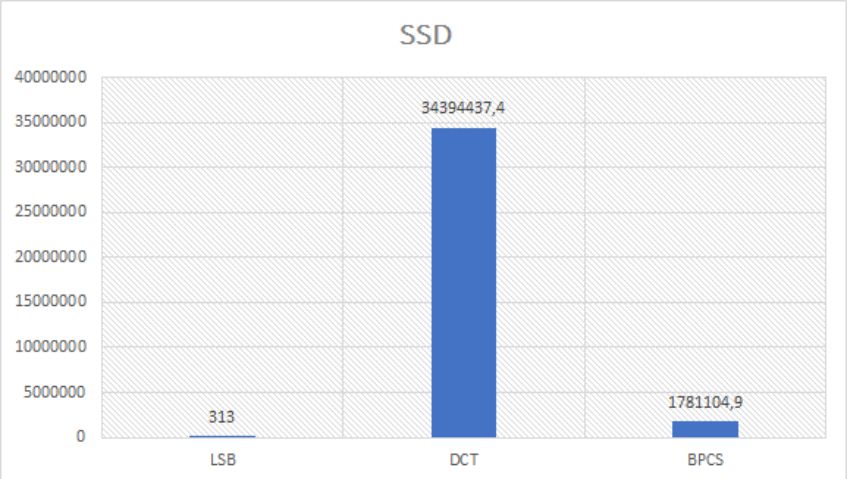
Fonte: O Autor (2021).

O *script* criado para calcular as métricas compara a imagem original com a imagem gerada pelos algoritmos somando seus resultados e calculando a média final.

É possível analisar que em média, o LSB apresentou resultados mais satisfatórios, levando em consideração o valor calculado para cada métrica. Para as métricas SSD, SAD e MAD, onde o valor para similaridade perfeita é zero, o LSB apresentou valores elevados, mas em comparação com os outros algoritmos, teve a melhor pontuação. O BPCS fica em segundo lugar no *ranking*, levando em consideração as três primeiras métricas, com valores elevados. O algoritmo que apresentou os piores resultados foi o DCT, onde é possível observar valores relativamente maiores do que os dos outros algoritmos de esteganografia. Essa grande diferença pode ser visualmente analisada nos gráficos representados na Figura 28, Figura 29 e Figura 30.

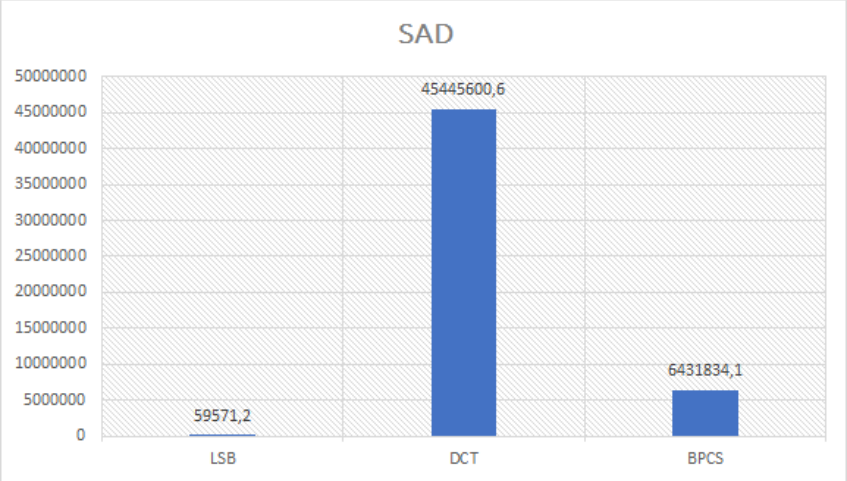
[1] #0010-0010#PatientName#PN#Diego Alcoforado Aires#;#0010-0020#PatientID#LO#010101B4#;#0010-0030#PatientBirthDate#DA#20000318#;#0010-0040#PatientSex#CS#M#;#0010-1000#OtherPatientIDs#LO#1987365293747#;#0010-1030#PatientWeight#DS#70#;#0010-21C0#PregnancyStatus#US#4#;

Figura 28 – Gráfico comparativo para SSD



Fonte: O Autor (2021).

Figura 29 – Gráfico comparativo para SAD

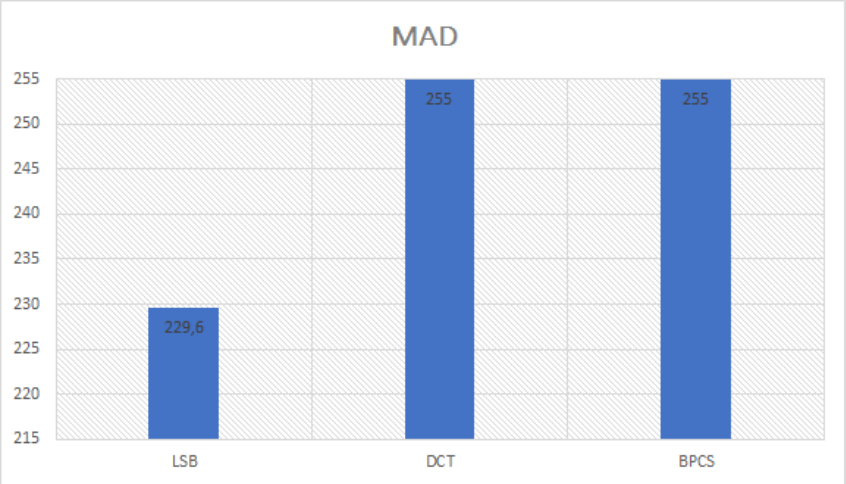


Fonte: O Autor (2021).

Após visualizar os gráficos, é possível perceber o quanto o método de esteganografia por DCT apresentou valores ruins para o conjunto de imagens utilizadas. Isso significa que não houveram boas correspondências entre a imagem original e seu estego-objeto, indicando claramente não se tratar de um bom método para a aplicação. Apenas analisando os dados, sem olhar a diferença entre as imagens, pode-se deduzir que ocorreram mudanças visuais significativas na imagem destino, o que vai contra o proposto de escolher um método que insira informação sem que sejam percebidas grandes alterações.

Especificamente na métrica MAD, que basicamente subtrai os valores dos pixels correspondentes e ao final retorna o maior valor, observa-se que tanto DCT quanto BPCS, tiveram pixels onde a alteração foi de 100%, devido o fato do maior valor possível ser 255, então em alguma correspondência, o mesmo pixel foi subtraído com 0.

Figura 30 – Gráfico comparativo para MAD

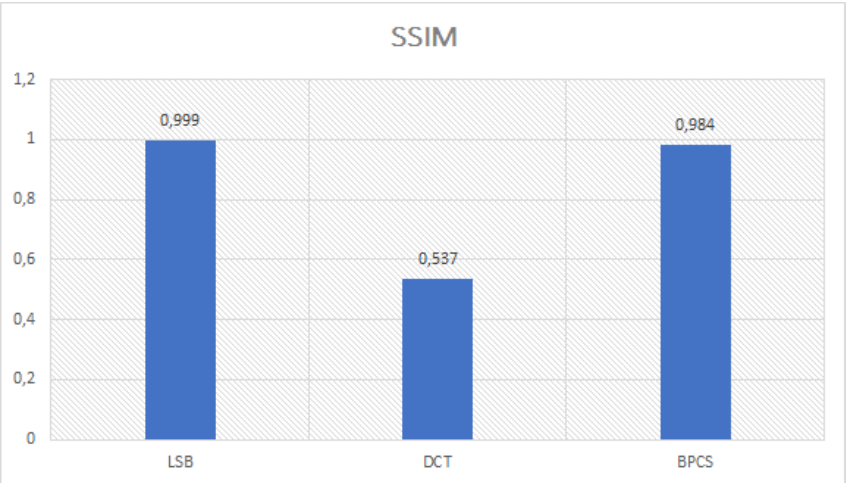


Fonte: O Autor (2021).

A métrica SSIM que calcula a similaridade estrutural entre dois objetos, utiliza valores entre 0 e 1, considerando 1 a similaridade perfeita e 0 totalmente diferente. Os valores para LSB e BPCS indicaram que a comparação entre as imagens originais e seus respectivos estego-objetos, indicam estar próximos à similaridade estrutural perfeita. O algoritmo DCT mais uma vez apresentou valores ruins e inferiores quando comparado com os outros algoritmos, apresentando uma alteração estrutural de aproximadamente 50%. É possível analisar isoladamente esses resultados verificando o gráfico da Figura 31.

Excluindo as métricas analisadas anteriormente, seria possível deduzir que LSB e BPCS seriam métodos equivalentes, já que apresentaram *scores* muito próximos. Mas levando em comparação as métricas anteriores, o LSB ainda apresenta uma vantagem, mostrando melhor desempenho segundo os dados.

Figura 31 – Gráfico comparativo para SSIM

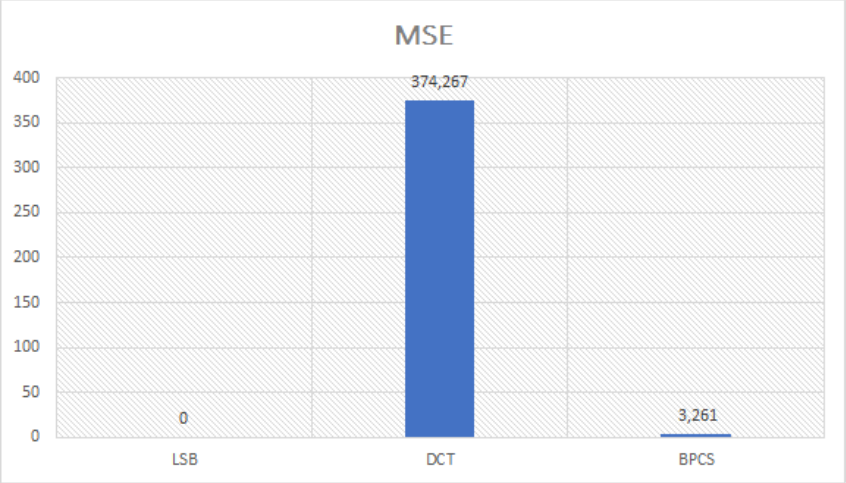


Fonte: O Autor (2021).

Analisando a tabela de valores médios e os gráficos da Figura 32 e Figura 33, mais uma vez é possível destacar que o método DCT foi o que apresentou menor similaridade segundo as métricas. Os outros dois métodos apresentaram valores aproximados, tanto para MSE quanto para RMSE. As duas métricas estão relacionadas, uma sendo baseada no resultado da primeira e o seus valores ideais são próximos a 0.

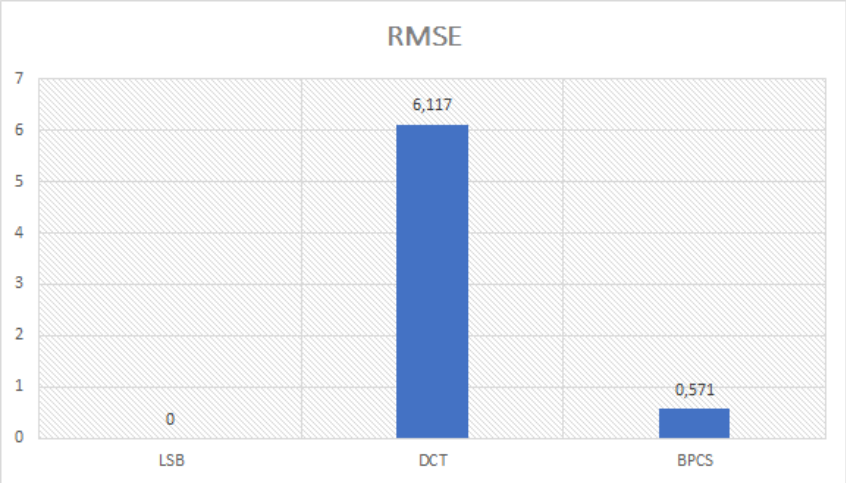
Após analisar todas a métricas com o auxílio visual dos gráficos, pode-se concluir que o algoritmo que performou melhor foi o LSB, nas métricas de SSIM e MSE, o algoritmo BPCS teve uma performance similar porém obteve resultado muito inferior nas métricas SSD, SAD e MAD. Em todas as métricas, o algoritmo DCT apresentou resultados insatisfatórios para ser aplicado na aplicação proposta, com base apenas nos dados, as diferenças para os outros algoritmos foi relativamente alta, desta forma ele pode ser considerado inviável para o protótipo.

Figura 32 – Gráfico comparativo para MSE



Fonte: O Autor (2021).

Figura 33 – Gráfico comparativo para RMSE



Fonte: O Autor (2021).

Até o momento, foram apresentados os resultados para valores médios de um conjunto de imagens. É interessante também verificar os resultados das métricas para cada uma das imagens, pois posteriormente pode-se analisar valores que tenham saído do padrão esperado, utilizando como auxílio a análise qualitativa, onde é possível identificar as diferenças visualmente.

O Quadro 9 apresenta os valores individuais para cada uma das imagens quando passaram pelo processo de esteganografia pelo algoritmo de LSB. É possível identificar que nas métricas utilizadas para calcular similaridade estrutural, todas as imagens obtiveram valores satisfatórios, próximos a 1 na SSIM e próximos a 0 na MSE.

Quadro 9 – Resultados individuais para LSB

LSB							
Arquivo	***Shape***	**SSD**	**SAD**	**MAD**	**SSIM**	**MSE**	**RMSE**
coracao_01.dcm	320x320x3	398	5732	255	0.999	0.003	0.062
coracao_02.dcm	320x320x3	420	420	1	0.999	0.004	0.063
cranio_01.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
cranio_02.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
cranio_03.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
torax_01.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
torax_02.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
torax_03.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
torax_04.dcm	512x512x3	289	73695	255	0.999	0.001	0.033
torax_05.dcm	512x512x3	289	73695	255	0.999	0.001	0.033

Fonte: O Autor (2021).

O Quadro 10 apresenta os valores individuais para cada uma das imagens quando passaram pelo processo de esteganografia pelo algoritmo de DCT. A maioria das métricas apresentaram valores insatisfatórios, estando muito longe do ideal. Nas duas primeiras imagens foi possível observar que a SSIM apresentou valores próximos a 1, porém levando em consideração todas as outras métricas e os valores individuais das outras imagens, todas estavam com valores de similaridade ruins.

Quadro 10 – Resultados individuais para DCT

DCT							
Arquivo	**Shape**	**SSD**	**SAD**	**MAD**	**SSIM**	**MSE**	**RMSE**
coracao_01.dcm	320x320x3	8807661	11271743	255	0.985	2657.332	51.549
coracao_02.dcm	320x320x3	8596435	12444407	255	0.985	2579.590	50.789
cranio_01.dcm	512x512x3	41460338	53566472	255	0.392	4004.943	63.284
cranio_02.dcm	512x512x3	35441543	56517029	255	0.508	3480.099	58.992
cranio_03.dcm	512x512x3	41435489	49997361	255	0.405	4031.008	63.490
torax_01.dcm	512x512x3	44933818	57334626	255	0.369	4374.557	66.140
torax_02.dcm	512x512x3	44742198	56768258	255	0.355	4322.875	65.748
torax_03.dcm	512x512x3	43022545	55769515	255	0.383	4172.866	64.597
torax_04.dcm	512x512x3	37772242	50442960	255	0.493	3747.887	61.219
torax_05.dcm	512x512x3	37732105	50343635	255	0.493	3742.672	61.177

Fonte: O Autor (2021).

O Quadro 11 que apresenta os valores individuais para as imagens após passar pelo algoritmo BPCS, mostra alguns resultados interessantes. Para as métricas SSD, SAD e MAD, apresentou valores muito distantes de 0, porém para SSIM apresentou valores quase ideais, próximos a 1, assim como na MSE, onde para a maioria das imagens do conjunto, apresentou valores próximos de 0.

Quadro 11 – Resultados individuais para BPCS

BPCS							
Arquivo	**Shape**	**SSD**	**SAD**	**MAD**	**SSIM**	**MSE**	**RMSE**
coracao_01.dcm	320x320x3	1196210	2360984	255	0.967	63.622	7.976
coracao_02.dcm	320x320x3	968457	2027461	255	0.976	49.490	7.034
cranio_01.dcm	512x512x3	462087	4972999	255	0.995	2.111	1.453
cranio_02.dcm	512x512x3	391856	5602876	255	0.996	1.704	1.305
cranio_03.dcm	512x512x3	623606	5647024	255	0.995	2.634	1.623
torax_01.dcm	512x512x3	1047041	6818853	255	0.991	4.370	2.090
torax_02.dcm	512x512x3	1038178	7032020	255	0.991	4.172	2.042
torax_03.dcm	512x512x3	1132530	7621536	255	0.990	4.700	2.168
torax_04.dcm	512x512x3	5500296	11125306	255	0.971	35.268	5.938
torax_05.dcm	512x512x3	5450788	11109282	255	0.971	32.614	5.710

Fonte: O Autor (2021).

De maneira geral. pode-se estabelecer que o tamanho da imagem não interfere de modo isolado na qualidade dos resultados. Utilizando o BPCS como exemplo, identificou-se que obteve resultados não satisfatório para as imagens de 320×320 e para apenas duas imagens de 512×512. Por outro lado, o algoritmo DCT obteve resultados melhores para os arquivos de 320×320 do que para os de tamanhos maiores, aproximando o índice SSIM de 1, sendo que nas imagens dos outros tamanhos ficaram com valores abaixo de 0.5. O algoritmo LSB apresentou valores melhores para imagens de tamanhos maiores, porém para o conjunto de dados analisado a diferença não foi muito significativa.

Um ponto importante para analisar é a quantidade de dados que cada algoritmo é capaz de esconder entre os *bytes* da imagem. Enquanto o BPCS consegue aproveitar até 50% do tamanho total de bits da imagem, os outros podem inserir somente 10% do tamanho total de bits da imagem.

Levando em consideração o conjunto de imagens utilizados para os testes, também foi realizada uma rodada de coleta de dados utilizando estego-imagens onde foram ocultados 38400 caracteres, que é o máximo suportado para as imagens de dimensão 320×320 quando utilizado o algoritmo LSB e DCT. Esse teste é necessário para validar o quanto a quantidade de bits alterados reflete nos valores das métricas.

No Quadro 12 é possível identificar que os valores dos resultados dos cálculos das métricas tiveram mais impacto no algoritmo LSB. Observa-se que as métricas SSD e SAD tiveram um aumento significativo, distanciando-se do valor ideal de similaridade. Porém, para SSIM e MSE os valores mantiveram-se relativamente próximos.

Quadro 12 – Média de 10 imagens ocultando 38400 caracteres

Métrica	**LSB**	**DCT**	**BPCS**
SSD	45503.6	34394437.4	1677597.4
SAD	7085291.4	45445600.6	6516443.2
MAD	255.0	255	255
SSIM	0.993	0.537	0.985
MSE	0.015	367.79	2.662
RMSE	0.0397	6.064	0.515

Fonte: O Autor (2021).

Outro ponto observado, é que para os algoritmos DCT e BPCS os valores das métricas tiveram pouca ou nenhuma alteração. Os valores de todas as métricas do Quadro 12 permaneceram inalterados para o DCT, enquanto para o BPCS a alteração foi pequena.

O Quadro 13 tem o objetivo de mostrar os resultados individuais para o arquivo *coracao_01.dcm* do conjunto de imagens, comparando os valores obtidos nas métricas calculadas em todos os algoritmos de esteganografia. Essa imagem suporta o total de 38400 caracteres embutidos por LSB, ou seja, ocupando a capacidade máxima.

É possível identificar que mesmo alterando o último bit de todos os pixels da imagem, a técnica de inserção no bit menos significativo apresentou valores mais próximos de similaridade em todas as métricas, levando em consideração a escala de cada uma, do que os demais algoritmos. A tendência é que quanto maior a imagem for, maior os resultados obtidos nas métricas SSD e SAD para o LSB, enquanto aparentemente, BPCS e DCT mantiveram valores próximos. Para as demais métricas em todos os algoritmos, o tamanho da imagem e a quantidade de caracteres inseridos, não causa alteração significativa nos resultados.

Quadro 13 – Resultados individuais

coracao_01.dcm			
Métrica	**LSB**	**DCT**	**BPCS**
SSD	59364	8807661	1151326
SAD	1785294	11271743	2469568
MAD	255	255	255
SSIM	0.997	0.985	0.968
MSE	0.345	2166.154	50.041
RMSE	0.588	46.541	7.074

Fonte: O Autor (2021).

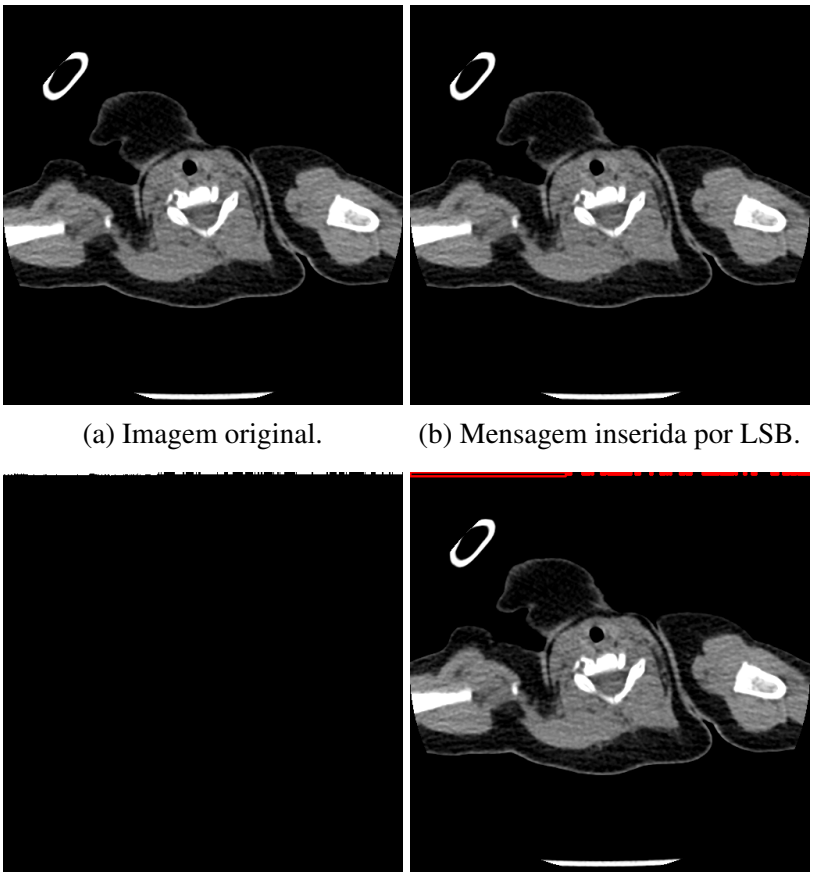
Mesmo utilizando a capacidade total de armazenamento de uma imagem por LSB, as métricas ainda são mais positivas do que as dos outros algoritmos. Para um protótipo, o LSB, segundo os dados, atenderia melhor aos requisitos estipulados. A restrição seria em relação ao tamanho do texto a ser inserido, que neste caso, para mensagens com maior quantidade de caracteres deve ser utilizado o algoritmo BPCS, porém a restrição são imagens de menor qualidade e pouca variabilidade de cores ou tons de cinza. O DCT estaria descartado em qualquer situação.

7.2 ANÁLISE QUALITATIVA

Após calcular as métricas, com o auxílio da função SSIM no *Python*, é possível destacar as diferenças estruturais entre as imagens comparadas. Tem-se como auxílio visual a imagem original, o estego-objeto, a imagem diferença gerada pela combinação da original com a final e uma imagem destacando em vermelho, sobre a imagem do estego-objeto, as diferenças.

A imagem diferença não reflete necessariamente uma alteração visual na imagem, embora sempre que exista esse tipo de alteração ela será destacada. Através dessa diferença, podemos rastrear em quais áreas da imagem o texto fora inserido. São comparadas imagens para um mesmo método e também uma mesma imagem, mostrando três resultados diferentes, uma para cada algoritmo de esteganografia.

Figura 34 – LSB



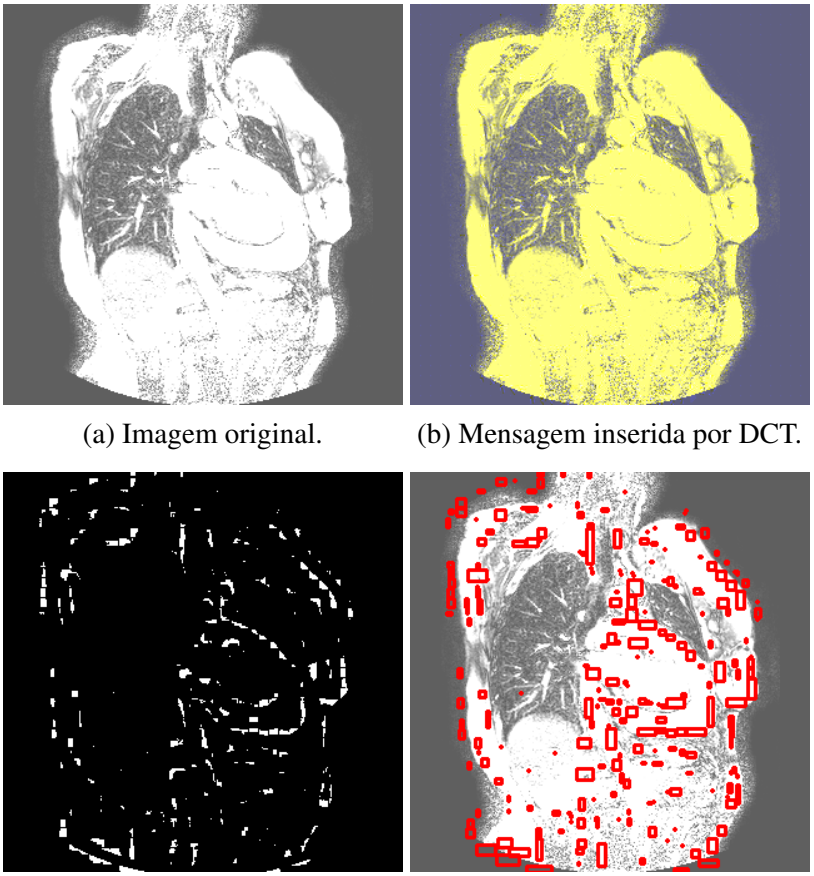
(a) Imagem original. (b) Mensagem inserida por LSB.

(c) Imagem diferença. (d) Diferenças selecionadas.

Fonte: O Autor (2021).

Na Figura 34 pode-ser perceber que o algoritmo iniciou o processo de inserção, pensando de forma matricial, por $(x, y) = (0, 0)$, inserindo no bit menos significativo do pixel e continuando a inserção na mesma linha. Não é possível identificar diferença visual entre (a) e (b), porém na imagem diferença (c) é possível identificar o rastro de pixels alterados. Em (d), apenas destacam-se em vermelho os locais alterados.

Figura 35 – DCT



(a) Imagem original. (b) Mensagem inserida por DCT.

(c) Imagem diferença. (d) Diferenças selecionadas.

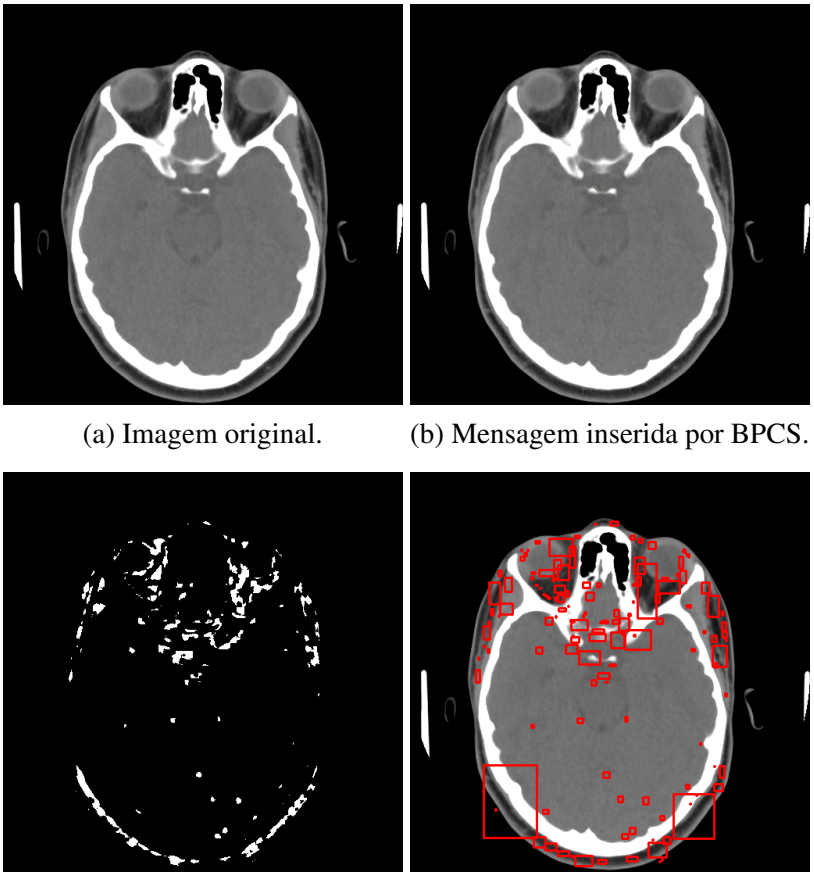
Fonte: O Autor (2021).

No algoritmo DCT, representado na Figura 35, é possível perceber mudanças visuais significativas. Apenas comparando (a) e (b), percebe-se que a cor original da imagem foi drasticamente alterada, o que era branco passou a ser amarelo, e o que era cinza, recebeu certo tom de azul.

Um coeficiente DCT transforma um sinal do domínio espacial para o domínio da frequência, desta forma, a imagem pode ser dividida entre alta, média e baixa componentes de frequência. Na sub-banda de baixa frequência ficam grande parte do conteúdo visual importante da imagem. Na sub-banda de alta frequência, os componentes geralmente são removidos por algoritmos de compressão. Por isso a mensagem secreta é incorporada modificando os coeficientes de frequência média. No caso do algoritmo de DCT, pode ser aplicado a um canal específico RGB, no caso do algoritmo apresentado, é no canal do azul B (JIANSHENG; SUKANG; XIAOMEI, 2009).

Analisando a imagem diferença (c) e (d), percebe-se que a mensagem é inserida principalmente nos contornos da imagem, pode-se visualizar que em (c) é visível o contorno do coração da imagem, assim como o formato do corpo do paciente.

Figura 36 – BPCS



(a) Imagem original. (b) Mensagem inserida por BPCS.

(c) Imagem diferença. (d) Diferenças selecionadas.

Fonte: O Autor (2021).

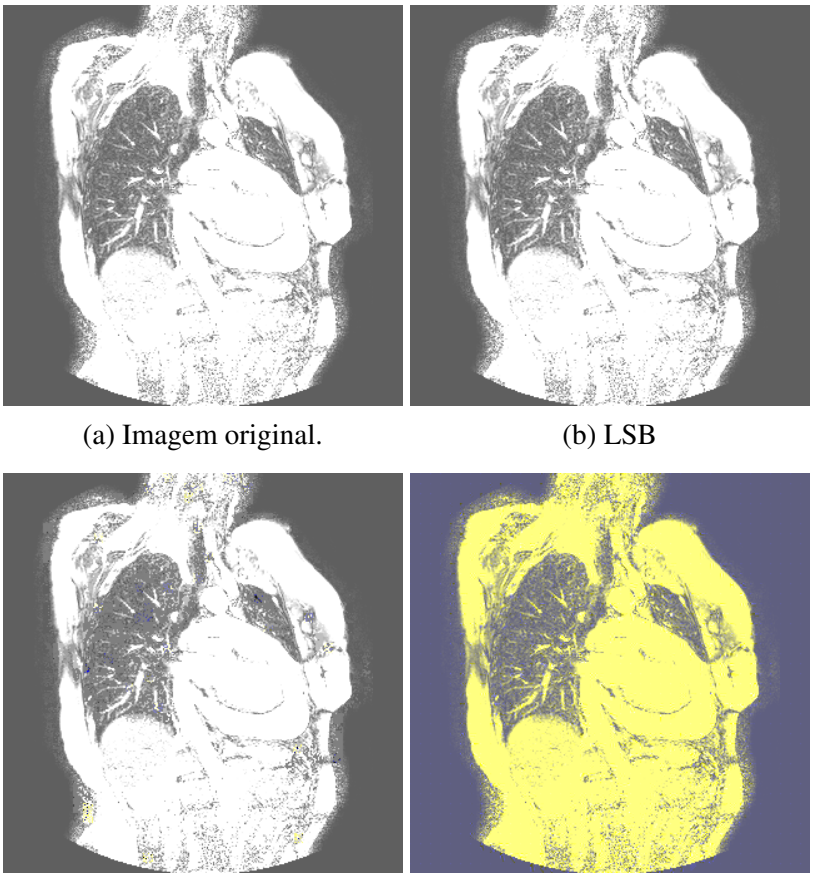
Analisando os resultados do algoritmo BPCS na Figura 36, quando comparado (a) e (b), não identificamos nenhuma alteração visual, aparentemente as imagens não possuem nenhuma diferença. Vale ressaltar que para imagens com melhores resoluções como a apresentada, o algoritmo não gera mudanças visuais, porém, foi notado que em imagens de menor qualidade e tamanho, o algoritmo gerou alterações visuais muito perceptíveis.

No que diz respeito a inserção da mensagem na imagem, o BPCS tem como princípio encontrar regiões de ruído, ou seja, regiões com uma variedade maior de informação diferente, é possível notar que a informação foi inserida nas fronteiras dos contornos dos componentes da imagem, por exemplo na região do olho, onde existe uma variação maior de cores, nota-se que não foi inserido nada em nenhuma região onde somente a cor preta é dominante.

Esse algoritmo age de maneira mais "inteligente"do que o LSB, buscando áreas que podem ser alteradas gerando pouca ou nenhuma alteração visual na imagem. Porém, ainda assim, segundo as métricas da análise quantitativa, para todas as imagens, o LSB se saiu melhor. Um outro ponto positivo do BPCS contra o LSB, é que ele pode inserir uma quantidade de informação maior dentro da imagem, aproximadamente 37% a mais.

Para analisar a mesma imagem comparando o resultado para cada algoritmo de esteganografia, foi utilizado um exemplo onde o BPCS gerou algumas alterações visuais indesejadas. A Figura 37 representa em (a) a imagem original, sem alterações, ao comparar (a) e (c), percebe-se alguma alteração de cor em alguns pontos da imagem, no pulmão esquerdo e no direito, alguns pixel tiveram suas cores alteradas para azul, enquanto no contorno do corpo do paciente, é possível observar alguns pixel amarelos, que não são vistos em (a). Comparando (a) e (b), nenhuma alteração visual é percebida, e comparando com (d), percebe-se muitas mudanças.

Figura 37 – Mesma imagem, todos os métodos



(a) Imagem original. (b) LSB

(c) BPCS (d) DCT

Fonte: O Autor (2021).

8 PROTÓTIPO

Esse capítulo visa reunir todo o trabalho desenvolvido em um protótipo, que através de uma interface gráfica, permite ao usuário a visualização dos conceitos aplicados de criptografia através do algoritmo RSA e de esteganografia através do algoritmo LSB. O protótipo possui diversas telas e funções para a manipulação de imagens *.dcm* utilizando os algoritmos selecionados. Para o desenvolvimento da interface do protótipo, assim como dos demais algoritmos, foi utilizada linguagem *Python*.

8.1 ALGORITMOS

De acordo com todas as análises desenvolvidas e as conclusões dissertadas nas seções 7.1 e 7.2, optou-se pela utilização do algoritmo LSB para incorporar ao protótipo, juntamente com o algoritmo de criptografia RSA. O algoritmo apresentou melhores resultados em todas as métricas calculadas e também demonstrou ter uma execução rápida. Por este motivo, algumas funções principais serão explicadas.

O Algoritmo 8 apresenta o processo pelo qual o texto é convertido em um *array* de bits. Essa função já recebe a mensagem criptografada em formato hexadecimal, então a conversão é feita sobre cada caractere, transformando-o para 8 bits. O seu retorno é um *array* binário com a mensagem completa.

Algoritmo 8 – Conversão da mensagem em bits

```
1  def convert_text_to_uint8(message):
2      msgarr = np.frombuffer(message.encode(), dtype=np.uint8)
3      msglen = msgarr.size
4      maskarr = np.zeros(msglen * 8, dtype=np.uint8)
5
6      arrind = 0
7      for char in msgarr:
8          for shift in range(8):
9              if(arrind == maskarr.size):
10                 break
11             if(char & (0x1 << (7 - shift))):
12                 maskarr[arrind] = 1
13             else:
14                 maskarr[arrind] = 0
15             arrind += 1
16
17     return maskarr
```

Fonte: O Autor (2021)

Para transformar os caracteres para binário, foram utilizados operadores *bitwise* para operações bit a bit, realizando um *shift left* e depois uma operação *AND*, caso o resultado dessa operação seja zero, é adicionado zero ao *array*, caso o resultado seja um valor maior que 0, é adicionado 1 ao *array*.

A função apresentada no Algoritmo 9, é a responsável por alterar o ultimo bit de cada *byte*, com o propósito de ocultar a mensagem criptografada. Recebe como entrada os *bytes* da imagem dispostos em forma de um *array* unidimensional, a matriz é previamente transformada, e após a execução da função, ela é redimensionada novamente às dimensões originais. O outro parâmetro de entrada é a mensagem representada em um *array* de bits. Os bits são embutidos na imagem e a função retorna o resultado das operações de inserção.

Algoritmo 9 – Inserção no último bit

```
1 def apply_LSB(image_bytes, array_Tbits):
2     for i in range(0, array_Tbits.size):
3         if(array_Tbits[i] == 0):
4             image_bytes[i] &= (~array_Tbits[i] - 1)
5         else:
6             image_bytes[i] |= array_Tbits[i]
7
8     return image_bytes
```

Fonte: O Autor (2021)

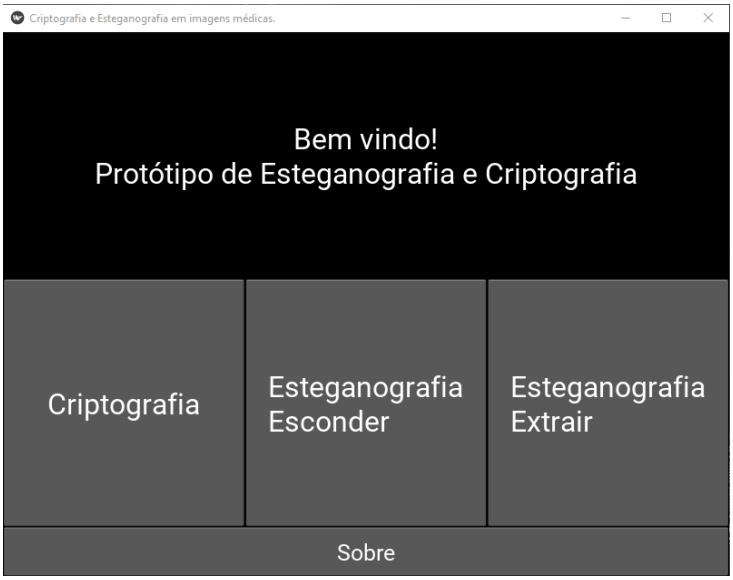
Para a manipulação dos bits nessa função, também foram utilizados os operadores *bitwise*, quando o bit da mensagem é 0, é realizada uma operação de *NOT*, uma negação, invertendo o valor do bit e em seguida, realiza-se a operação de *AND* com o *byte* da imagem. Caso o valor do bit da mensagem seja 1, é realizada uma operação *OR* do *array* da mensagem com o *array* da imagem.

8.2 INTERFACE

A interface de um projeto é importante para que alguém que não tenha conhecimento prévio de um programa possa utilizá-lo. A interface foi desenvolvida para três situações distintas, de acordo com os fluxo apresentados na Seção 5.1, a geração das chaves de criptografia, o processo de ocultação da mensagem dentro da imagem por esteganografia e o processo de extração da mensagem por esteganografia. A Figura 38 mostra a tela de boas vindas, onde são oferecidas três opções de diferentes ações, além de uma tela com um conteúdo adicional no rodapé *Sobre*, que apresenta algumas informações do desenvolvedor, das tecnologias utilizadas no trabalho e nos algoritmos relacionados, no caso o LSB e o RSA.

A biblioteca *kivy* permite separar o código da interface da lógica das funcionalidades. Para isso, escreve-se um arquivo *.kv* que é o responsável por montar os componentes da tela.

Figura 38 – Tela inicial



Fonte: O Autor (2021).

O primeiro passo é gerar as chaves pública e privada na tela de *Criptografia*, que é apresentada na Figura 39, para que posteriormente possam ser utilizadas nos processos estenográficos. As chaves são criadas e salvas em uma pasta dentro dos diretórios do projeto, desta forma, elas podem ser importadas e utilizadas.

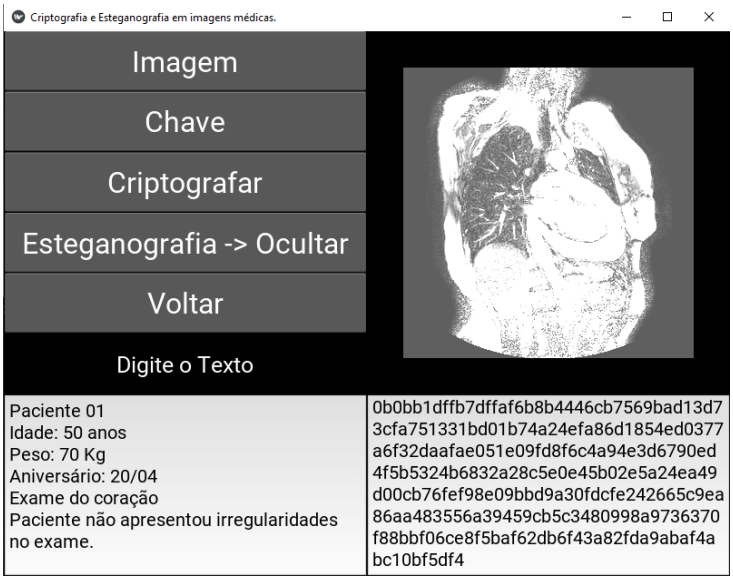
Figura 39 – Gerar Chaves

Fonte: O Autor (2021).

Na tela de *Esteganografia - Esconder*, representada na Figura 40, é possível gerar todo o processo de encriptação de uma mensagem e de ocultar dentro da imagem. Quando o processo é concluído é exibida uma mensagem informado que a operação foi concluída com sucesso. Para

que se obtenha êxito, é necessário escrever uma mensagem e criptografá-la e somente depois selecionar o botão para aplicar esteganografia. Quando a mensagem é criptografada, é exibido logo ao lado em outra caixa de texto o resultado gerado após ter passado pelo algoritmo RSA, para essa exibição é utilizado o formato hexadecimal.

Figura 40 – Esteganografia - Ocultar



Fonte: O Autor (2021).

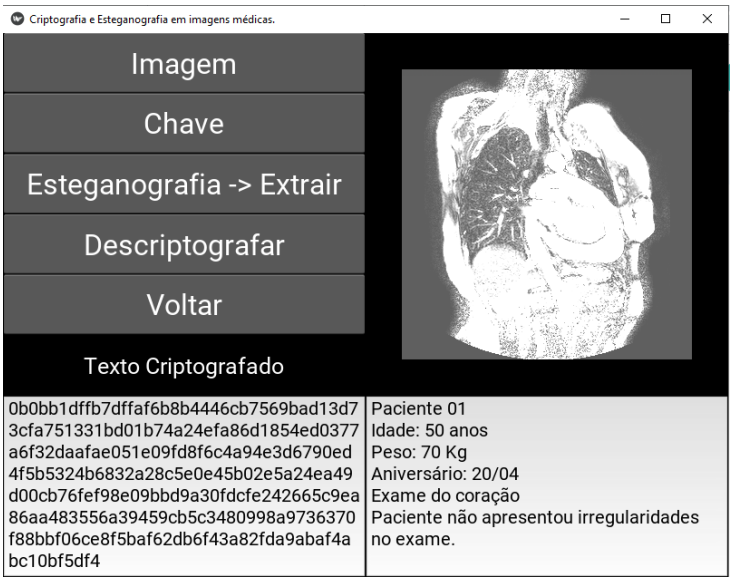
Cada uma das ações é representada por um botão ou caixa de texto, que de forma específica possuem as seguintes ações:

- Imagem: Abre uma tela de seleção para que o arquivo *.dcm* seja selecionado. Ao ser selecionado, é exibido na parte direita da tela.
- Chave: Abre uma janela de seleção para importar o arquivo *.pem* de chave pública, previamente criado.
- Criptografar: Com o texto previamente escrito, chama a função de criptografia RSA e exibe o resultado.
- Esteganografia Ocultar: Chama a função de esteganografia por LSB passando o texto criptografado. Cria um novo arquivo na pasta de imagens dentro do projeto.
- Voltar: Volta para a tela inicial.
- Digite o Texto: Insere o texto que será oculto dentro da imagem.

A terceira e última tela, representada na Figura 41, faz o processo inverso do apresentado na tela da Figura 40. Através dela é possível pegar uma imagem que passou pelo processo

de esteganografia e extrair o seu conteúdo. O conteúdo extraído será apresentado de forma criptografada, e poderá somente ser descriptografado se a chave privada, que foi gerada juntamente com a chave pública, for importada. Com a chave privada sendo anexada e sendo válida, o texto pode ser descriptografado e exibido na caixa de texto ao lado daquela que apresenta o resultado da extração da mensagem criptografada de dentro da estego-imagem.

Figura 41 – Esteganografia - Extrair



Fonte: O Autor (2021).

Cada um dos botões desempenha uma função, sendo elas:

- Imagem: Abre uma tela de seleção para que o arquivo *.dcm* seja selecionado. Esse arquivo deve ter passado pelo processo de esteganografia
- Chave: Abre uma janela de seleção para importar o arquivo *.pem* de chave privada, utilizado para descriptografar o texto.
- Esteganografia Extrair: Chama a função de esteganografia por LSB para extrair os dados previamente ocultados dentro da imagem. Os dados extraídos estarão criptografados.
- Descriptografar: Possuindo uma chave privada válida, é possível descriptografar o texto extraído da imagem, que é exibido na caixa de texto ao lado.
- Voltar: Volta para a tela inicial.

A estrutura do projeto está organizada de maneira a separar código (interface e funções), imagens *.dcm* comuns, estego-imagens, chaves de criptografia e os algoritmos RSA e o LSB em pastas separadas, como apresentado na Figura 42 a estrutura do projeto. Desta maneira, para futuras alterações no protótipo, o processo torna-se mais organizado.

Figura 42 – Estrutura do projeto

Fonte: O Autor (2021).

O protótipo, assim como os códigos dos algoritmos e as imagens utilizadas nos testes, podem ser localizados e baixados diretamente do *Github*[1] e estão livres para contribuições e alterações.

[1] https://github.com/MatheusSche/algos-tcc

9 CONCLUSÃO

Ao longo do desenvolvimento do trabalho, buscou-se entender e testar combinações de criptografia e esteganografia. É possível combiná-las, mas é necessário considerar alguns fatores, como o tipo de aplicação que está sendo desenvolvida e o tamanho das mensagens que serão embutidas. Ao longo do trabalho, identificou-se que algumas técnicas de criptografia e esteganografia são mais utilizados que outras, dependendo da aplicação onde estão sendo utilizados e do nível de segurança requerido.

Ao estudar-se criptografia, identifica-se diversas técnicas com diferentes níveis de criptografia. A Cifra de César, por exemplo, um exemplo clássico de criptografia, é considerado de baixa segurança pois pode ser facilmente quebrada. Ao longo do tempo foram surgindo técnicas mais avançadas para criptografar informação, o RSA é um exemplo disso. A criptografia moderna pode ser dividida em técnicas de chave simétrica e assimétrica, em ambos esses campos, existem algoritmos avançados. Algoritmos de chave assimétrica, utilizam o conceito de chave pública, onde a chave de encriptação é diferente da chave de decrepitação, sendo a chave pública podendo ser conhecida por qualquer um, enquanto na chave simétrica, a mesma chave secreta é utilizada nos dois processos criptográficos. Por esse motivo o RSA foi selecionado para o protótipo, pois utiliza conceitos mais modernos e um ótimo nível de segurança.

Em esteganografia, identificou-se que pode ser usada em diversas áreas e em diversos tipos de arquivo. Como o trabalho foi voltado para arquivo de imagem, as técnicas estudadas também se focaram para interagir com esses formatos de arquivos. Algumas técnicas agiam no domínio da frequência enquanto outras agiam no domínio espacial das imagens. Técnicas baseadas em DCT que alteram o domínio da frequência, geraram mais distorções na imagem, indicando não ser ideal utilizá-la em aplicações onde alterações visuais não devam ser geradas. Técnicas que alteram o domínio espacial como LSB e BPCS, geram menos distorções, LSB ainda menos que o BPCS. O BPCS consegue armazenar mais informação do que LSB, conseguindo ocupar até 50% do tamanho total em *bytes* da imagem, e o LSB apenas 10%. Porém o BPCS causa mais alterações estruturais na imagem e em imagens de menor qualidade, gera alterações visíveis ao olho.

Por esses motivos, foi identificado que o ideal para o protótipo, seria utilizar o algoritmo baseado em LSB, pois não gerou alterações visuais nem em imagens de menor qualidade, além de que de acordo com as métricas utilizadas para medir similaridade esse algoritmo apresentou melhores resultados. Mesmo que consiga ocultar uma quantidade menor de informação, ainda assim, para o conjunto de informações utilizados para inserção, demonstrou-se ideal.

Verificou-se que algumas métricas, por mais que medissem a similaridade entre os objetos comparados, não retornaram dados muito legíveis para tomada de decisão, é o caso do SSD, SAD e MAD, que foram úteis para a análise de diferença dos algoritmos, mas ao mesmo tempo retornaram valores em uma escala muito ampla. Essas métricas, inicialmente foram identificadas para alguma operações específicas, como rotação, translação de imagens, e não para esteganografia. As métricas SSIM e MSE demonstraram resultados mais legíveis, confirmando a relação de diferença que fora apontada nas outras métricas, e por possuírem escalas menores, porém altamente confiáveis, tornam-se melhores opções para analisar similaridade sobre imagens que receberam algum tipo de transformação por esteganografia.

Como trabalhos futuros, são sugeridos alguns pontos, envolvendo os algoritmos de esteganografia e também referente a melhoria do protótipo:

- Testar diferentes proporções entre tamanho da imagem e a quantidade de caracteres da mensagem a ser inseridos, para validar o comportamento das métricas de similaridade.
- Utilizar outros algoritmos de esteganografia, outras métricas de similaridade ou otimizar os utilizados neste trabalho.
- Melhorar a interface gráfica.
- Permitir que sejam utilizados diversos algoritmos de esteganografia, criando um menu para escolha.

REFERÊNCIAS

BARBOSA, L. D. A.; CORNELISSEN, M. G. Cifra de hill: uma aplicação ao estudo de matrizes. **Revista Ciencias Exatas e Naturais**, Salinas, v. 19, n. 2, p. 152–167, 2017.

BRAGA, A.; DAHAB, R. Introdução à criptografia para programadores: evitando maus usos da criptografia em sistemas de software. In: SIMPOSIO BRASILEIRO EM SEGURANCA DA INFORMACAO E DE SISTEMAS COMPUTACIOONAIS, 15., 2015, Florianópolis. **Livro-Texto de Minicursos**. Florianópolis: FAPERJ, 2015. p. 1–50.

CHANDRAMOULI, R. Mathematical approach to steganalysis. In: III, E. J. D.; WONG, P. W. (Ed.). **Security and Watermarking of Multimedia Contents IV**. SPIE, 2002. v. 4675, p. 14 – 25. Disponível em: <https://doi.org/10.1117/12.465273>.

COSTA, E. M.; CAETANO, N. G. Criptografia com utilização de cifra de hill e cifra afim. **Matemática e Estatística em Foco**, Uberlândia, v. 5, n. 1, p. 14–21, 2017.

COUTINHO, S. C. **Números Inteiros e Criptografia RSA**. Rio de Janeiro: IMPA, 2005. 226 p.

DOBBERTIN, H.; KNUDSEN, L.; ROBSHAW, M. The cryptanalysis of the aes. In: CONFERENCE ON ADVANCED ENCRYPTION STANDARD, 4., 2004, Bonn. **Advanced Encryption Standard – AES**. Bonn: Springer, 2004. p. 1–10.

ESPERANCA, L. **Cifra de César em linguagem C**. 2016. Disponível em: <https://capivararex.wordpress.com/2016/01/30/cifra-de-cesar-em-linguagem-c-2/>.

FIERRASGA, V. M. C. **Criptografia e Matemática**. 144 f. Dissertação (Mestrado em Matemática) — Universidade de Lisboa, Lisboa, Lisboa, 2010.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3. ed. São Paulo: Pearson Education, Inc., 2010. 624 p.

GUTTMAN, B.; ROBACK, E. **An Introduction to Computer Security: the NIST Handbook**. [S.l.]: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 1995.

HARMOUCH, M. **17 types of similarity and dissimilarity measures used in data science**. [s.n.], 2021. Disponível em: <https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681>. Acesso em: 02 jun. 2021.

HOSSEM, A. F. *et al.* Tics na sociedade do conhecimento: método de hill para ensino de criptografia. **Vozes dos Vales: Publicações Acadêmicas**, Mucuri, v. 1, n. 13, p. 1–21, 2018.

JAIN, D. **LSB Image Steganography Using Python**. The Startup, 2021. Disponível em: <https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2>.

JIANSHENG, M.; SUKANG, L.; XIAOMEI, T. A digital watermarking algorithm based on dct and dwt. In: **Proceedings of the International Symposium on Web Information Systems and Applications, (WISA'09**. [S.l.]: Academy Publisher, 2009. p. 104–107. ISBN 978-952-5726-00-8.

JULIO, E. P.; BRAZIL, W. G.; ALBUQUERQUE, C. V. N. Esteganografia e suas aplicações. In: SIMPOSIO BRASILEIRO EM SEGURANCA DA INFORMACAO E DE SISTEMAS COMPUTACIOONAIS, 7., 2007, Rio de Janeiro. **Livro de Minicursos**. Rio de Janeiro: FAPERJ, 2007. p. 54–102.

KAMATA, S.; EASON, R.; KAWAGUCHI, E. Depth-first coding for multivalued pictures using bit-plane decomposition. **IEEE Transactions on Communications**, v. 43, n. 5, p. 1961–1969, 1995.

KAMINSKYI, B. **Python Interview Questions. Part 0: How Python works**. Medium, 2019. Disponível em: <https://medium.com/@b.kaminskyi99/python-interview-questions-part-0-how-python-works-2fd0a4275865>.

KATZENBEISSER, S.; PETITCOLAS, F. **Information Hiding Techniques for Steganography and Digital Watermaking**. [S.l.: s.n.], 1999. v. 28. 1-2 p. ISBN 978-1-58053-035-4.

KAWAGUCHI, E. Bpcs-steganography – principle and applications. In: KHOSLA, R.; HOWLETT, R. J.; JAIN, L. C. (Ed.). **Knowledge-Based Intelligent Information and Engineering Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 289–299. ISBN 978-3-540-31997-9.

____. **Invitation to BPCS-Steganography**. [s.n.], 2015. Apresenta o funcionamento do algoritmo de esteganografia BPCS. Disponível em: <http://datahide.org/BPCSe/principle-e.html>. Acesso em: 11 jun. 2021.

KAWAGUCHI, E.; EASON, R. O. Principles and applications of BPCS steganography. In: TESCHER, A. G. *et al.* (Ed.). **Multimedia Systems and Applications**. SPIE, 1999. v. 3528, p. 464 – 473. Disponível em: <https://doi.org/10.1117/12.337436>.

LADEIRA, R. de L. R.; RAUGUST, A. S. Uma análise da complexidade do algoritmo rsa implementado com o teste probabilístico de miller-rabin. **Revista de Empreendedorismo, Inovação e Tecnologia**, Passo Fundo, v. 4, n. 1, p. 24–33, 2017.

PAAR, C.; PELZL, J. **Understanding Cryptography**: A textbook for students and practitioners. 1. ed. Berlin: Springer, 2010. 372 p.

PERIN, G. **Arquiteturas de criptografia de chave pública**: análise de desempenho e robustez. 85 f. Dissertação (Mestrado em Informática) — Universidade Federal de Santa Maria, Santa Maria, 2011.

PETITCOLAS, F.; ANDERSON, R.; KUHN, M. Information hiding-a survey. **Proceedings of the IEEE**, v. 87, n. 7, p. 1062–1078, 1999.

PIANYKH, O. S. **Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide**. 1. ed. [S.l.]: Springer Berlin Heidelberg, 2008. ISBN 9783540745709; 354074570X; 9783540745716; 3540745718.

POPA, R. **An analysis of steganography techniques**. Dissertação (Mestrado em Computer Science) — The "Polytechnic" University of Timisoara, Timisoara, Romênia, 1998.

PROVOS, N.; HONEYMAN, P. Detecting steganographic content on the internet. **CITI Technical Report**, Michigan, p. 01–11, 12 2001.

RAVAL, P. **Measuring similarity in two images using Python**. Towards Data Science, 2021. Disponível em: <https://towardsdatascience.com/measuring-similarity-in-two-images-using-python-b72233eb53c6>.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, v. 21, p. 120–126, 1978.

ROCHA, A. *et al.* Segurança e privacidade na internet por esteganografia em imagens. Lavras, 01 2004. Disponível em: <https://www.researchgate.net/publication/229004358_Seguranca_e_privacidade_na_internet_por_esteganografia_em_imagens>. Acesso em: 31 mai. 2021.

SALOMON, D. **Data Compression: The Complete Reference**. Berlin, Heidelberg: Springer-Verlag, 2006. 899 p. ISBN 1846286026.

SAMPAIO, A. R.; JACKOWSKI, P. M. Avaliação de métodos esteganográficos em imagens médicas. **Revista Eletrônica De Iniciação Científica Em Computação**, São Paulo, v. 13, n. 4, 2014. Disponível em: <https://sol.sbc.org.br/journals/index.php/reic/article/view/1044>. Acesso em: 17 mai. 2021.

SEMOLA, M. **Gestão da Segurança da Informação**: uma visão executiva. 1. ed. Rio de Janeiro: Elsevier, 2002. 184 p.

STALLINGS, W. **Criptografia e segurança de redes**: princípios e práticas. 6. ed. São Paulo: Pearson Education do Brasil, 2015. 558 p.

SULLIVAN, K. *et al.* Steganalysis of quantization index modulation data hiding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING. **[s.n.]**. 2004. Disponível em: <http://vision.ece.ucsb.edu/publications/04ICIPKen.pdf>.

ULYSSES, J. N.; CONCI, A. Mathematical approach to steganalysis. In: LETA, F. R.; CONCI, A. (Ed.). **IWSSIP Proceedings**. EdUFF, 2010. Disponível em: <https://www.iwssip.org/archive/2010/Proceedings/nav/paper.htm>.

WANG, Z. *et al.* Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, 2004.

WAYNER, P. **Disappearing Cryptography**: Information hiding: Steganography and watermarking. 3. ed. Burlington: Morgan Kaufmann Publishers, 2008. 439 p.