Centro de Ensino Universitário de Brasília - UniCEUB
Faculdade de Ciências Exatas e Tecnologia - FAET
Departamento de Engenharia da Computação
Projeto Final

Esteganografia – Inserção de Texto ou de Arquivo em Vídeo MPEG-1

Henrique Lanna Passos
RA 2021826-4

Orientador: Prof. Aderlon M. Queiroz
Brasília/DF – Dezembro/2006

Agradecimentos

Agradeço a Deus, que me deu forças para concluir este objetivo.

À minha família, principalmente aos meus pais, pelo carinho, apoio e confiança, estando presentes em todos os momentos na caminhada para a conclusão do curso de Engenharia da Computação.

Ao meu orientador, Aderlon M. Queiroz, por toda sabedoria, apoio, incentivo e disponibilidade para a conclusão deste trabalho.

A todos os professores, colegas e amigos que de alguma forma participaram e colaboraram para a conclusão desta importante etapa de minha vida.

Resumo

Este trabalho apresenta a implementação de um protótipo, escrito na linguagem C, que utiliza uma técnica de esteganografia para esconder uma mensagem de texto ou um arquivo, em qualquer formato, dentro de um vídeo do formato MPEG-1, sem que as alterações feitas no vídeo sejam perceptíveis ao olho humano. Além desta técnica, foi utilizado um algoritmo de criptografia DES (*Data Encryption Stardard*) a fim de aumentar a confidencialidade da informação a ser transmitida.

Palavras chaves: esteganografia, algoritmo de criptografia DES, formato de vídeo MPEG-1.

Abstract

This work presents the implementation of a prototype, written on the C language, that uses a steganography technique to hide a text message or a file of any format into a MPEG-1 video format, without any noticeable difference from the original video to the human eye. Aside from this technique it was used a cryptography algorithm (Data Encryption Standard) to increase the confidentiality of the information that will be transmitted.

Keywords: steganography, DES cryptography algorithm, MPEG-1 video format.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 – Introdução

1.1- Motivação

A Internet é um meio de tráfego de informações que está em notável expansão. Esse fenômeno gera alguns problemas no quesito segurança do conteúdo trafegado. Com isso, a busca por mecanismos eficientes de proteção digital tornou-se uma necessidade para o atendimento dos requisitos fundamentais da segurança (integridade, confidencialidade e disponibilidade).

O uso da rede mundial de computadores deixou de ser apenas uma ferramenta de pesquisa e lazer para se tornar um dos mais promissores centros de negócio. Considerando que a História nos ensina que onde há riquezas e concentração de poder existe também a cobiça, esse sentimento é despertado em pessoas que se especializam em roubar e interceptar informações de outras pessoas e empresas.

A perda ou exposição indevida de uma informação valiosa acarreta prejuízos e perda de credibilidade. Uma solução encontrada para driblar a falta de privacidade é a utilização de criptografia. Apesar dessa técnica ser bastante eficiente e possuir algoritmos robustos, ela é perceptível, pois a mensagem interceptada pelo especialista, mesmo que ele não decifre seu conteúdo, revela a existência de uma informação cifrada.

Então, é da necessidade de camuflar a informação dos interceptores que surgiu a esteganografia moderna, baseada em técnicas digitais de ocultamento. Essa técnica utiliza arquivos de multimídias (som, vídeo, imagem e textos) para serem portadores de mensagem encobertas. Dessa forma, ao ser interceptado, o arquivo seria considerado indiferente no quesito de valor da informação explícita, passando despercebidamente a mensagem de real valor.

Apesar dos benefícios que essa técnica pode prover - principalmente sigilo e privacidade - ela também vem sendo aplicada para fins ilegais, haja vista sua utilização para o terrorismo, a pedofilia e a pirataria, dentre outros males do mundo contemporâneo.

Para combater o mau uso dessa técnica surgiu a esteganálise, que busca maneiras de detectar e decifrar o conteúdo das informações escondidas. Apesar da recuperação da informação ser complexa, só o fato de identificar sua presença já é suficiente para melhor investigação das práticas do emissor e do receptor.

Portanto, ao analisar a evolução das técnicas de segurança adotadas para o sigilo de informações pode-se observar que as técnicas são utilizadas tanto para o "bem" como para o "mau". Novos mecanismos de proteção e de mascaramento são desenvolvidos em paralelo a mecanismos de identificação e quebra de paradigmas. Estabelece-se, pois, um verdadeiro círculo vicioso, que consiste na descoberta de novas técnicas, em seguida seu mau uso, a descoberta de mecanismos de proteção, novamente seu mau uso, novas formas de proteção, conseqüentemente resultando na evolução da segurança da informação digital.

1.2- Metodologia

Neste trabalho, para a implementação do protótipo, foi utilizada a linguagem de programação C. O algoritmo de ocultamento foi baseado na técnica de substituição de dados, onde os *bytes* de algumas figuras do vídeo foram substituídos por fragmentos da mensagem que se deseja esconder. Além disso, um algoritmo de criptografia DES foi utilizado no código do protótipo.

A ferramenta de esteganálise escolhida para provar a existência de esteganografia no vídeo foi *Hex Workshop* 4.2 da empresa *BreakPoint Software*. Essa ferramenta faz a comparação entre dois arquivos de vídeo, o primeiro com conteúdo original e o outro com conteúdo esteganografado.

1.3- Objetivo

O objetivo do trabalho foi o desenvolvimento de um *software* utilizando a linguagem C, com a finalidade de esconder uma mensagem ou um arquivo dentro de um arquivo de vídeo comum no padrão MPEG-1. Assim, a mensagem será transmitida de forma mais segura, dentro desse arquivo. Visando ampliar a

confidencialidade da informação ocultada foi utilizado um algoritmo de criptografia DES (*Data Encryption Standard*).

1.4- Estrutura do trabalho

Após o capítulo introdutório, o Capítulo 2 expõe os aspectos históricos, as principais componentes da estrutura e as principais técnicas de compressão presentes no formato de vídeo MPEG-1.

No Capítulo 3 são apresentados os principais conceitos, os aspectos históricos, os principais métodos de ocultamento da informação e as características gerais de alguns tipos de arquivos portadores utilizados no processo de esteganografia.

No Capítulo 4 são comentados alguns dos principais conceitos, terminologias, aspectos históricos, algoritmos de criptografia e características do algoritmo DES. Além disso, é feita a diferenciação entre os termos criptografia e esteganografia.

O Capítulo 5 é referente ao protótipo desenvolvido, incluindo informações sobre o equipamento, os *softwares*, a linguagem e algoritmo utilizados ao longo da implementação.

No Capítulo 6 são apresentados os testes, simulações e resultados obtidos.

No Capítulo 7 são expostas as considerações finais e conclusões do trabalho, limitações, dificuldades e, também, as recomendações para trabalhos futuros.

Capítulo 2 – Padrão MPEG-1

2.1- Aspectos Históricos

O padrão MPEG-1 surgiu da necessidade de se introduzir vídeos para a Era Digital. O grande problema inicial era que a digitalização de vídeos sem compressão ocupava um enorme volume de dados. Por exemplo, a capacidade de gravação de vídeo em um CD limitava-se a aproximadamente 5 minutos de imagem, sem som.

Então, em 1988, foi criado um comitê de pesquisa, constituído por 300 especialistas em sistemas de vídeo, denominado MPEG (*Moving Picture Expert Group*). A proposta do MPEG era reduzir a quantidade de dados necessária para representar um vídeo digital e ainda manter elevada a qualidade da imagem, ou seja, realizar a compressão digital.

O primeiro padrão de vídeo lançando por esse comitê foi o MPEG-1 (ISO & IEC, 93a; ISO & IEC, 93b), formalmente denominado ISO/IEC 11172. Esse padrão fornece suporte à codificação de vídeo associado a áudio em taxa de dados entre 1 e 1,5 Mbit/s. Seu projeto de desenvolvimento tem como objetivo atender aplicações como vídeo em CD-ROM (*Video* CD - VCD), *Digital Audio Tapes* (*DATs*), codificadores de câmera digitais (*CamCoders)* e aplicações de transmissão de vídeo por meio de redes de telecomunicações ou pela *Internet.* [Vasconcelos, 2005]

Apresenta-se, a seguir, um resumo das versões de padronização do MPEG que foram aprovadas ao longo dos últimos anos:

- MPEG-1, padrão para armazenamento e carregamento de filmes e áudio em dispositivos de armazenamento (aprovado em Novembro de 1992).[Fernandes, 1999]
- MPEG-2, padrão para televisão digital (aprovado em Novembro de 1994).[Fernandes, 1999]

- MPEG-4 Versão 1, padrão para aplicações multimídias (aprovado em Outubro de 1998) ainda em desenvolvimento.[Fernandes, 1999]
- MPEG-4 Versão 2 (aprovado em Dezembro de 1999).[Fernandes, 1999]
- MPEG-4 Versão 3, 4 e 5 estão nas etapas finais de desenvolvimento. [Chiariglione, 2006]
- MPEG-7, padrão para representação de conteúdo para procura, filtragem, gerenciamento e processamento de informações multimídia, que está nas etapas finais de desenvolvimento (aprovado em 2002) [Fernandes, 1999]
- MPEG-21, padrão para multimídia *framework*, o qual é baseado em dois conceitos essenciais: a definição de uma unidade fundamental de distribuição e negociação (o Item Digital) e o conceito de usuários interagindo com os Itens Digitais. Esses Itens Digitais são os elementos que compõem o multimídia *framework* como, por exemplo, uma coleção de vídeos ou um álbum de música, e o usuário é quem utiliza o multimídia *framework*.[Chiariglione, 2006]

2.2- ISO/IEC 11172

De acordo com a espeficicação ISO/IEC 11172 (“Tecnologia de Informação –Codificação de imagens em movimento associadas a áudio para armazenamento em mídias digital até 1,5 Mbit/s”, Copyright 1993), o padrão MPEG-1 é composto de cinco partes. As três primeiras partes alcançaram o status de padrão internacional (ISO) no inicio de 1993, a quarta parte atingiu esse status ISO em 1994 e a quinta parte alcançou esse status em meados de 1995. Essas partes podem ser visualizadas a seguir:

2.2.1 – ISO/IEC 11172-1

Essa primeira parte é referente ao sistema MPEG-1. Descreve a sincronização e a multiplexação do vídeo e do áudio, como pode ser visto na figura a seguir:

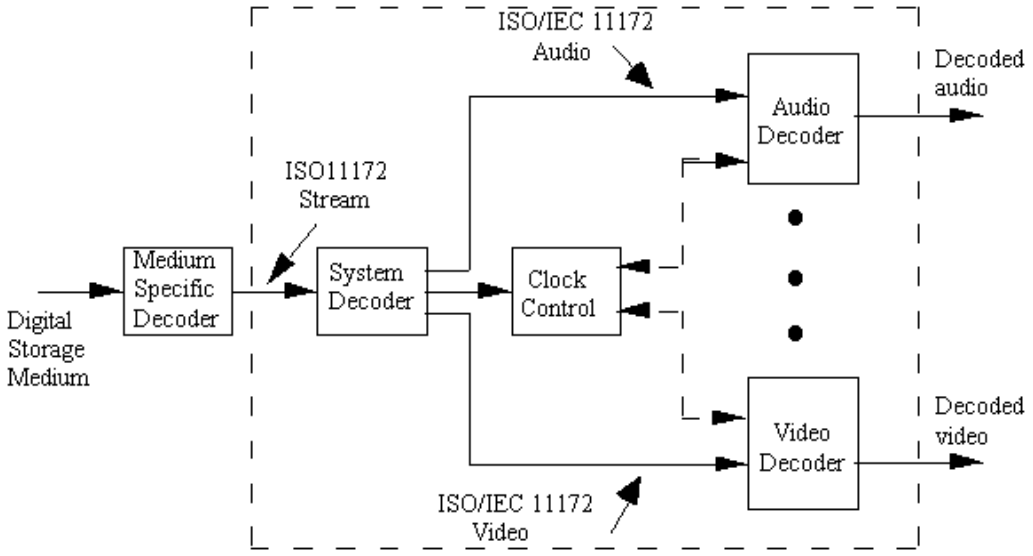

Figura 2.1: Protótipo do decodificador

Fonte: [Chiariglione, 2006]

Na figura 2.1, é mostrado o processo de demultiplexação dos dados armazenados do vídeo digital (*Digital Storage*). Esse processo é feito por meio de um decodificador do sistema (*System Decoder*) que divide os dados em duas *streams* (fluxo de dados), uma de vídeo e uma de áudio. O padrão MPEG trata separadamente vídeo e áudio, especificando como estes sinais são associados e sincronizados, possuindo assim três partes: a camada de sistema, a camada de vídeo e a camada de áudio. Estas camadas serão discutidas nos itens posteriores.

2.2.2 – ISO/IEC 11172-2

A segunda parte referente ao sistema MPEG-1 descreve a compressão de sinais de vídeo não entrelaçados, ou seja, traz informação sobre a sintaxe (cabeçalho e elementos da cadeia de *bits* - *bitstream*). A estrutura de vídeo desse padrão pode ser vista na figura seguinte:

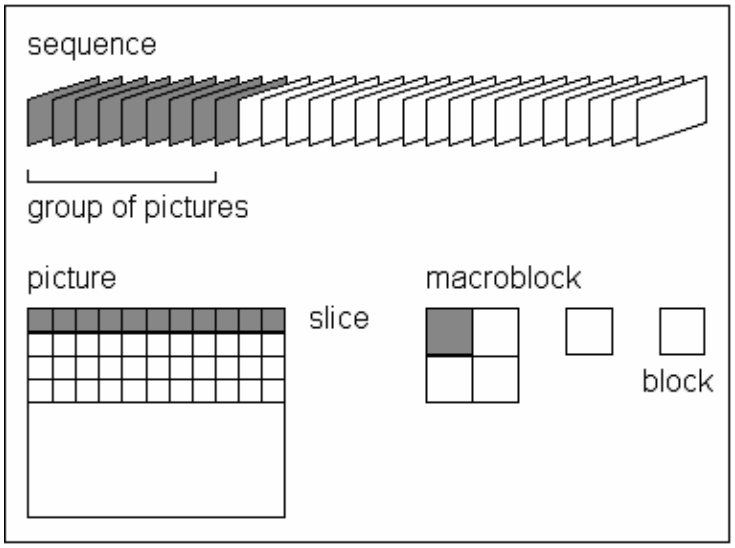

Figura 2.2: Estrutura básica de vídeo no formato MPEG-1

Fonte: [Chiariglione, 2006]

2.2.3 – ISO/IEC 11172-3

Essa terceira parte referente ao sistema MPEG-1 descreve a compressão de sinais de áudio, isto é, traz informações referentes à sintaxe e semântica para três classes de métodos de compressão de áudio, conhecidas como Camadas I, II e III, como pode ser visto na figura a seguir:

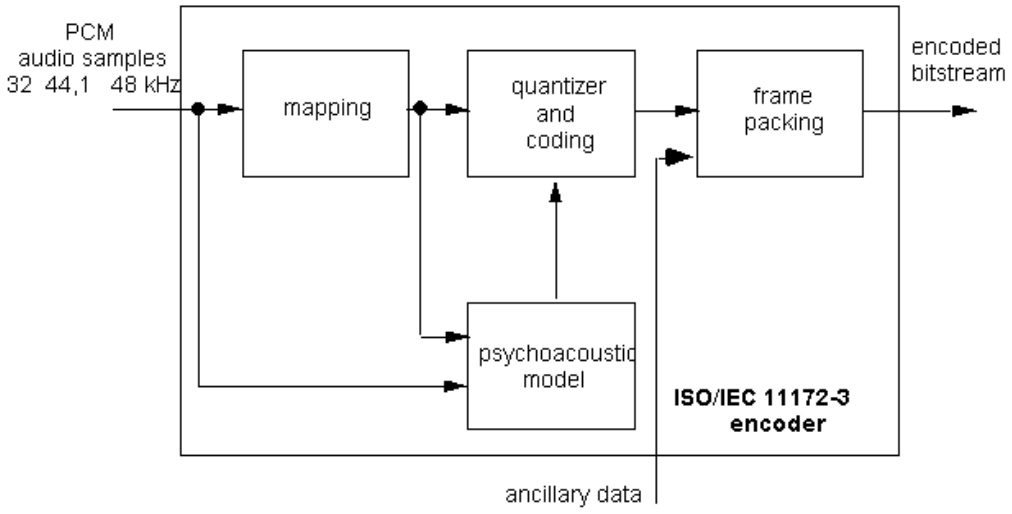

Figura 2.3: Estrutura básica de um codificador de áudio

Fonte: [Chiariglione, 2006]

2.2.4 – ISO/IEC 11172-4

A quarta parte referente ao sistema MPEG-1 é sobre Testes de Conformidade. Ela descreve os procedimentos usados para determinar as características das cadeias de *bits* (*bitstream*) codificados e o processo de decodificação, adequados às determinações das partes 1 (Sistema), 2 (Vídeo) e 3 (Áudio) do padrão MPEG-1.

2.2.5 – ISO/IEC 11172-5

A quinta parte referente ao padrão MPEG-1 é a simulação do *software*. Nela é apresentado um exemplo, escrito na linguagem ANSI C de codificador (*encoder*), e um decodificador (*decoder*), ambos para vídeo e áudio. Além disso, é fornecido um exemplo de sistema *codec* que pode multiplexar e demultiplexar, separando cadeias elementares distintas de áudio e vídeo contidas em arquivos de computador.

2.3- Codificação da Estrutura

Os vídeos do formato MPEG-1 possuem uma estrutura organizada em camadas ordenadas a partir da camada mais global, chamada de *Video Sequence*, para a camada mais interna, denominada *Block*. Essas camadas são: *Video Sequence, GOP (Group of Pictures), Picture, Slice, Macroblock, Block*. Elas podem ser visualizadas na figura detalhada a seguir:

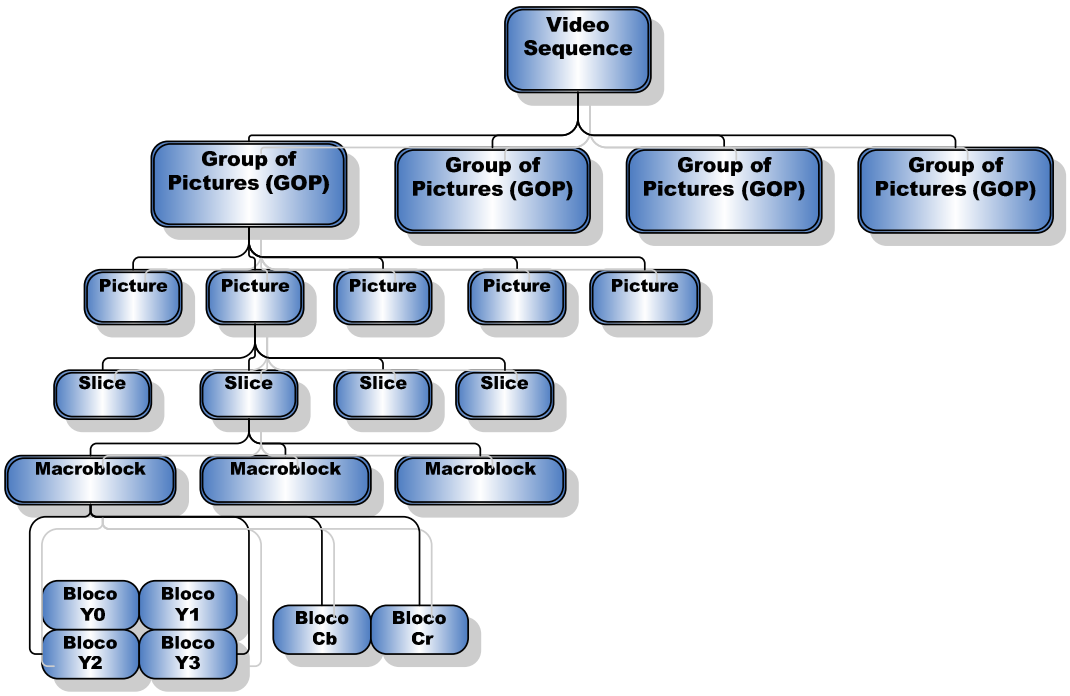

Figura 2.4: Estrutura hierárquica de vídeo do MPEG-1

Fonte: Autor

2.3.1- *Video Sequence*

É a camada principal da estrutura do MPEG-1. Essa camada engloba um ou mais grupos de figuras (GOP - *Group of Pictures*). A seqüência de vídeo (*Vídeo Sequence*) contém algumas informações como:

- *sequence_header* - cabeçalho de inicialização do vídeo;
- *horizontal_size* e *vertical_size* - informa o comprimento e a largura da visualização do vídeo na tela;
- *picture_rate* - informa o número de quadros visualizados por segundo;
- *sequence_end_code* - cabeçalho de finalização do vídeo.

Essa camada pode ser visualizada na figura 2.5:

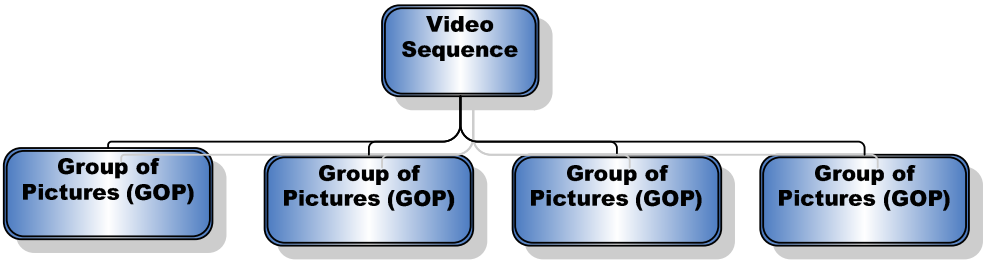

Figura 2.5: Camada *Vídeo Sequence*

Fonte: Autor

2.3.2 – *Group of Pictures (GOP)*

É uma camada formada pela seqüência de uma ou mais figuras. Embora o padrão MPEG-1 permita que o *GOP* contenha quantos quadros se deseja, a quantidade de quadros mais utilizada varia entre 9 a 18 figuras (quadros) por grupo. Algumas das principais informações encontradas nessa camada são:

- *group_start_code* - cabeçalho de inicialização de um grupo de figuras;
- *time_code* - apresenta informações temporais e de edição de vídeo;
- *close_gop* - indica se um grupo de figuras está aberto ou fechado.

Essa camada pode ser visualizada na figura 2.6:

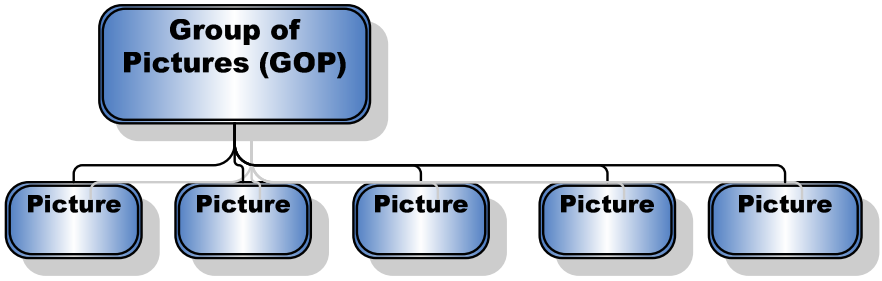

Figura 2.6: Camada GOP (*Group of Pictures*)

Fonte: Autor

2.3.3 – *Picture*

A camada de figura possui todas as informações codificadas da representação de uma imagem presente no vídeo. Essa camada engloba um ou mais *slices*, que são pedaços de uma imagem. A principal informação armazenada nessa camada é o *picture_coding_type*, que identifica se a figura é dos seguintes tipos:

- intracodificado (*intra-coded*), representado por “I” - este tipo não referencia outras figuras.
- predição causal (*predictive*), representado por “P” - a figura atual está baseada em dados da figura anterior.
- predição interpolativa (*bidirectionally predictive)* representado por “B” - a figura atual apresenta alto grau de compressão, pois é baseada em dados de figuras passadas e futuras, para compensação de movimento.
- *dc-coded*, representado por “D” - a figura armazena apenas a informação dos coeficientes DC da imagem.

Essa camada pode ser visualizada na figura 2.7:

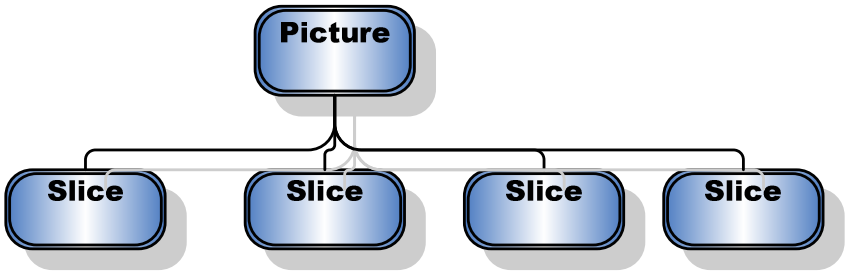

Figura 2.7: Camada Picture
Fonte: Autor

2.3.4- Slice

A camada *Slice* é formada pelo agrupamento de um ou mais macroblocos (*macroblock*) percorridos na direção horizontal, da esquerda para a direita e de

cima para baixo. Um *Slice* pode conter no mínimo um Macrobloco (*Macroblock*), ou até várias linhas de Macrobloco. Essa camada é responsável pela recuperação de erros nas figuras. Essa característica permite que o decodificador possa avançar para o próximo *Slice* ao encontrar um erro. [Vasconcelos, 2005]

O *Slice* não deve sobrepor outros *slices* e não devem existir lacunas vazias entre *slices*. Obrigatoriamente, o primeiro *slice* deve ser inicializado com o primeiro Macrobloco da figura e o último *slice* deve possuir o último Macrobloco da figura.

Essa camada pode ser visualizada na figura 2.8:

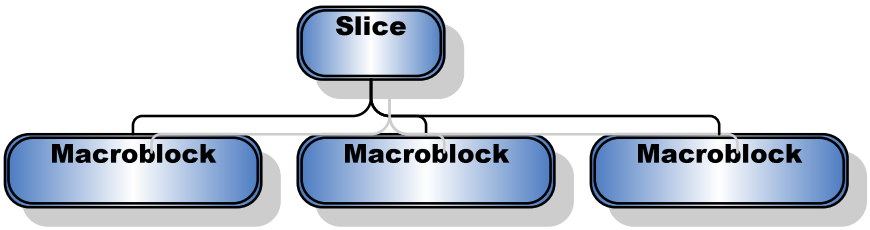

Figura 2.8: Camada *Slice*

Fonte: Autor

2.3.5- *Macroblock*

Um Macrobloco (*Macroblock*) é uma região codificada a partir de 16 por 16 *pixels* da imagem original. O número total de Macroblocos em uma figura depende do tamanho em *pixels* da própria figura, não sendo admitido um número fracionário de Macroblocos. Desta forma, as dimensões horizontal e vertical da imagem devem ser obrigatoriamente múltiplas de 16. [Vasconcelos, 2005]

Essa camada é a unidade básica usada para o cálculo da compensação de movimento e para alterações na escala de quantização. [Vasconcelos, 2005] Os canais de cromância (cor) são sub-amostrados nas direções vertical e horizontal, na razão de 2 por 1. Então, cada Macrobloco é constituído por quatro blocos de luminância (Y), com informações sobre o brilho, e dois blocos de cromância (Cb e Cr), com informações sobre a cor. O limite de Macroblocos é 396 por figura.

O Macrobloco é considerado a menor unidade codificada da imagem, pois essa camada é a mais interna da estrutura MPEG a selecionar camada de um método de compressão.

Essa camada pode ser visualizada na figura 2.9:

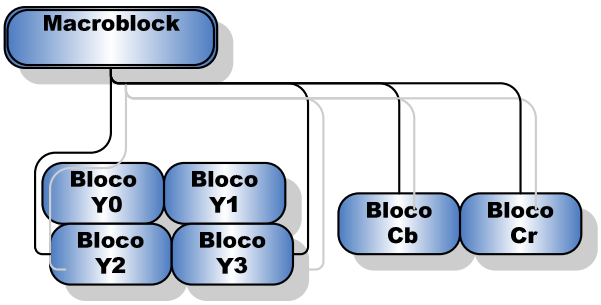

Figura 2.9: Camada Macrobloco

Fonte: Autor

2.3.6-*Block*

A camada mais interna é denominada Bloco (*Block*), que é a unidade básica para aplicação da Transformada Discreta de Cosseno (DTC). Cada Bloco é formado por uma matriz de 8 por 8 coeficientes. Desta forma, os 64 valores armazenados nos blocos representam a informação dos *pixels*, transformados para o domínio da freqüência. [Vasconcelos, 2005]

Essa camada pode ser visualizada na figura 2.10:

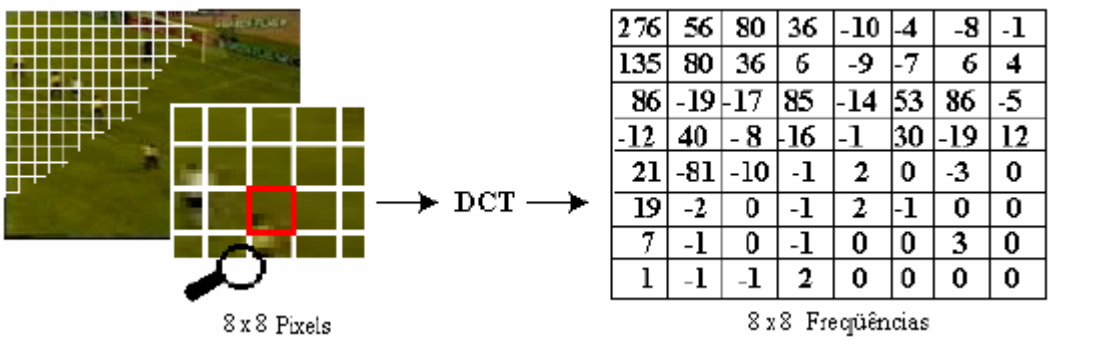

Figura 2.10: Camada Bloco

Fonte: [Vasconcelos, 2005]

2.4- Técnicas de Compressão

O principal objetivo das técnicas de compressão é reduzir o tamanho em *bytes* do armazenamento do vídeo sem perda de qualidade perceptível ao olho humano.

Para atingir elevadas taxas de compressão, o padrão MPEG-1 utiliza técnicas híbridas para eliminar informações redundantes, presentes nas imagens da seqüência de um vídeo.

Os principais métodos de compressão usados no padrão MPEG-1 são: Sub-amostragem das informações de cromância; Quantização; Compensação de Movimento; DTC (Transformada Discreta do Cosseno); VLC (*Variable Length Coding*), Interpolação de Figuras.

2.4.1- Sub-amostragem das informações de cromância

No geral, cada *pixel* (menor unidade de representação de uma imagem computadorizada) presente em uma figura é composto de três cores fundamentais: Vermelho (R - *Red*), Verde (*G - Green*) e Azul (*B - Blue*). Essas cores devem ser convertidas para valores luminância (Y) e cromância (Cb e Cr) para serem codificadas no padrão MPEG-1.

O olho humano é mais sensível a componentes de luminância (brilho). Então, os valores de Y dos *pixels* são codificados em resolução completa. Em contrapartida, componentes de cromância (cor) são menos sensíveis à visão humana. Portanto, os valores de cores são sub-amostrados nos valores de Cb e Cr, sem perda perceptível de qualidade.

Esse método reduz a quantidade de dados armazenados para representação dos *pixels* sem afetar a da visão do expectador.

2.4.2- Quantização

A Quantização reduz uma escala de valores para um único valor. Por exemplo, o fato de se converter um número real para o número inteiro mais próximo é uma forma de quantização. A diferença entre o valor atual e o valor

quantizado é denominada ruído da quantização. Em algumas circunstâncias, a visão humana é menos sensível ao ruído da quantização. Assim, esse ruído pode ser ampliado, aumentando-se a eficiência da codificação.[ISO 111772-2, 1993]

2.4.3- Compensação de Movimento (MC)

A compensação de movimento (*Motion Compensation*) é a predição de valores dos *pixels* na próxima figura usando um Bloco da figura atual. O deslocamento da posição do bloco da figura previamente codificada em relação à posição do bloco da figura atual é chamado de Vetor de Movimento (*Motion Vector* - MV). A diferença entre esses blocos é chamada de predição de erro (*prediction error*). No MPEG-1, o codificador (*encoder*) calcula o MV e a predição de erro. Quando o decodificador obtém as informações desse vetor e da figura atual, elas podem ser utilizadas para a construção da próxima figura.

Para exemplificar de maneira bastante simples o uso dessa técnica, tome-se uma cena de vídeo em que a câmera está fora de movimentação e nenhum objeto da cena está se movendo. Os valores dos *pixels* localizados em cada imagem permanecem na mesma posição e o vetor de movimento de cada bloco é “zero”, ou seja, não houve nenhuma alteração entre as figuras. Em geral, o codificador deve transmitir um vetor de movimento para cada bloco. Assim, o bloco traduzido a partir do bloco conhecido (atual) se torna a predição para a codificação do bloco da próxima figura a ser codificada. Essa técnica utiliza apenas um pequeno trecho de uma seqüência de figuras presentes pertencentes a uma mesma cena, quando a maioria dos objetos permanece estática, enquanto alguns outros fazem pequenos deslocamentos.

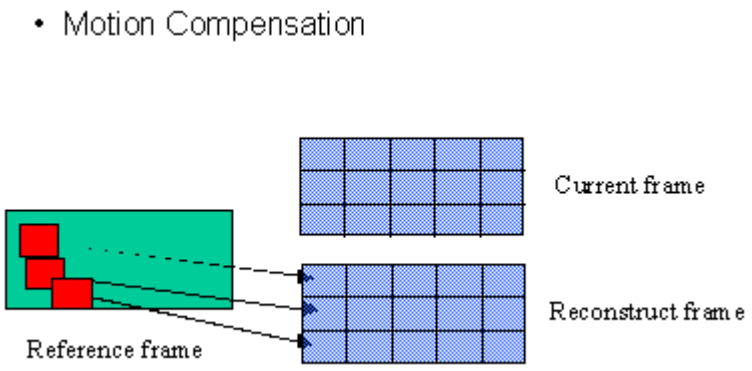

Figura: 2.11- Técnica de Compensação de Movimento

Autor: Cmlab[1]

2.4.4 – DTC

A Transformada Discreta de Cosseno converte um bloco de 8 por 8 *pixels,* pertencente ao domínio espacial, para o domínio de freqüências. Nesse estágio, os valores armazenados no bloco representam os coeficientes das freqüências horizontais e verticais da região representada pelo bloco. Os valores dos *pixels* podem ser reconstruídos pela aplicação da Inversa da Transformada Discreta do Cosseno (IDTC) nos coeficientes de freqüência. [Vasconcelos, 2005]

No domínio da freqüência ocorre uma concentração da energia do sinal da imagem equivalente às informações mais perceptíveis ao olho humano, em torno das baixas freqüências. Essa informação de maior relevância visual é armazenada na parte superior esquerda dos blocos. As informações menos perceptíveis, representadas pelas altas freqüências, são armazenadas no extremo oposto do bloco, na parte inferior direita. [Vasconcelos, 2005]

A compressão é alcançada pelo uso da técnica de quantização, onde os intervalos de quantização são identificados por um índice. A partir do princípio que o codificador identifica o intervalo e não o valor exato dentro do intervalo, os valores dos *pixels* do bloco reconstruído pela IDTC apresentam uma precisão reduzida.

[1] http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/mpeg1/

2.4.5 – VLC

Variable-Lenght Coding (VLC) é uma técnica de codificação estatística que determina *codewords* (palavras-código) para os valores que serão codificados. Essa técnica utiliza pequenas palavras-código (*short codewords*) para representar os valores que ocorrem mais freqüentemente e utiliza grandes palavras-código (*long codewords*) para representar valores que ocorrem com menor freqüência. A partir dessa técnica é gerado um código final com menor tamanho do que a representação dos dados originais.

O processo de codificação (*encoding*) e decodificação (*decoding*) utilizando o método VLC, no padrão MPEG-1, deve utilizar uma tabela-código VLC como referência. Essa tabela apresenta duas entradas, a primeira com a representação dos dados originais e a segunda com as *codewords* correpondentes. Nesse padrão de vídeo podem ser definidas diferentes tabelas-código VLC. Portanto, a utilização de múltiplas tabelas é uma forma de melhorar as taxas de compressão da técnica VLC.

2.4.6- Interpolação de Figuras

Nesse método, o decodificador reconstrói uma figura a partir de uma figura passada ou de uma figura futura. Então as figuras intermediárias podem ser reconstruídas utilizando-se a interpolação, ou seja, predição bidirecional. Os Macroblocos presentes nas figuras intermediárias podem possuir predição *forward* (adiantamento) e predição *backward* (retardo) que apresentam dois vetores de movimento (MV). Já as figuras com apenas uma das duas predições apresentam somente um vetor de movimento (MV). Dessa forma, o decodificador pode reconstruir os valores dos *pixels* pertencentes a um dado Macrobloco com uma média de valores de figuras passadas e futuras.

O esquema da técnica de Interpolação de Figuras pode ser visto na figura 2.12.

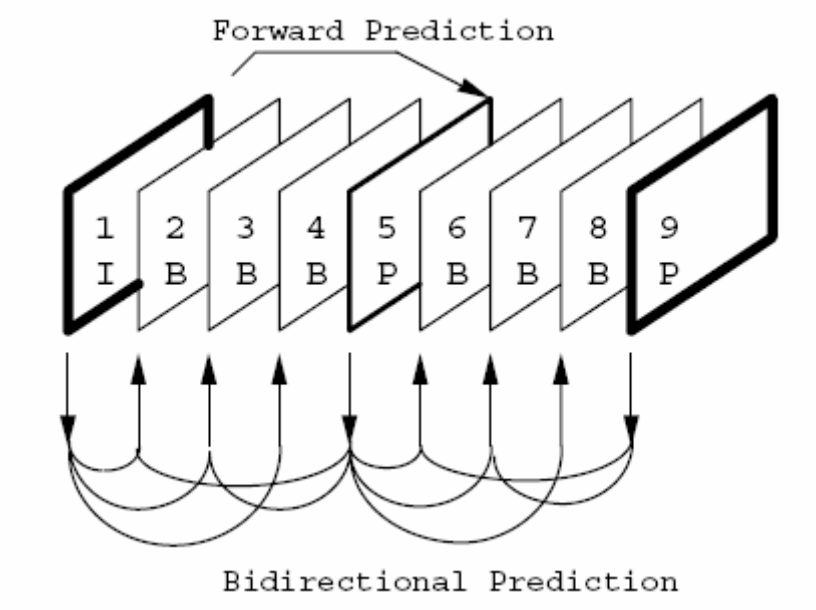

Figura: 2.12- Técnica de Interpolação de Figuras
Autor: [ISO 111772-2, 1993]

A figura 2.12 mostra a técnica de interpolação de Figuras (quadros). Esses quadros podem ser de três tipos básicos: I (intracodificado), B (predição interpolativa) e P (predicação bidirecional). Como pode ser visto na referida figura, o segundo quadro da seqüência é do tipo B e possui relacionamento com o primeiro quadro que é do tipo I e com o terceiro quadro que é do tipo B, pois todos os quadros do tipo B dependem de informações dos quadros anterior e posterior no processo de sua construção. Os quadros do tipo P só dependem do quadro anterior para sua formação. Como exemplo, o nono quadro da figura 2.12, que depende apenas do oitavo. Já os quadros do tipo I são independentes dos demais quadros para sua construção. Por exemplo, o primeiro quadro da seqüência apresentada na figura 2.12.

Após essa análise sobre a história, a estrutura e as técnicas de compressão do formato MPEG-1, pode-se entender que este formato de vídeo apresenta características que possibilitam o uso de esteganografia em sua estrutura, sem que essa prática afete a qualidade visual do vídeo para o espectador.

No capítulo 3 serão comentados os principais conceitos e as técnicas de esteganografia para melhor entendimento de como são feitas as modificações da

estrutura do vídeo, qual a técnica que foi adotada para tal feito e o porquê da escolha.

Capítulo 3 – Esteganografia

3.1- Definição

A palavra esteganografia é uma derivação das palavras gregas *steganos,* que significa protegido ou secreto, e *graphien,* que significa escrita ou desenho. Portanto, o termo esteganografia pode ser entendido como a arte de esconder informações sigilosas em meios de transmissão multimídia sem percepção de sua presença.

A esteganografia é um dos ramos da área de pesquisa denominada ocultamento da informação *(information hiding*). Outros ramos da hierarquia dessa área são: canais secretos, anonimato e marcação de *copyright*. Essa estrutura está demonstrada na figura 3.1.

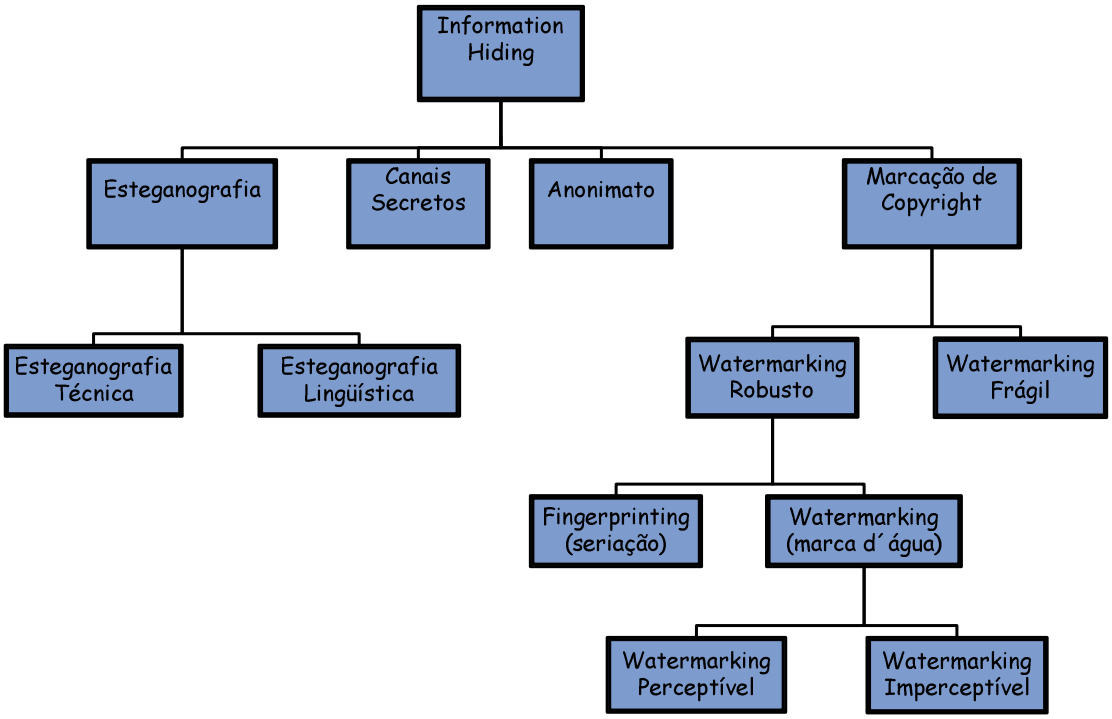

Figura 3.1: A hierarquia do *information hiding*

Fonte: Autor

Entende-se por canais secretos a criação de uma comunicação entre duas partes em que o meio é secreto e seguro. Um exemplo seriam as conversações militares em faixas de freqüência moduladas. [Pfitzmann, 1996 apud Rocha, 2003, p.7]

A arte da esteganografia constitui a segunda ramificação da hierarquia. Pode ser dividida em lingüística e técnica. Quando a mensagem é fisicamente escondida, tal como a maioria dos exemplos que serão apresentados no item 3.2, configura-se a chamada esteganografia técnica. Por outro lado, quando a mensagem é trabalhada e o seu ocultamento depende de propriedades lingüísticas, tal como a esteganografia digital, configura-se a chamada esteganografia lingüística. [Pfitzmann, 1996 apud Rocha, 2003, p.8]

Anonimato é um conjunto de técnicas para se tentar navegar na *Internet* sem que o agente seja localizado. Isto poderia ser feito utilizando-se *sites* de desvio, como o *Anonymizer*[2] e/ou os *remailers* — *sites* capazes de enviar mensagens secretas não revelando seu remetente —. [Pfitzmann, 1996 apud Rocha, 2003, p.8] O tunelamento *http* é outra técnica que estabelece uma conexão criptografada em um meio de comunicação (*proxy*) entre um computador da uma rede interna e outro computador de uma rede externa, impedindo a identificação do conteúdo da informação trafegada. Logo, o mecanismo de bloqueio ou filtro desse conteúdo é burlado. *Softwares* como *Your Freedom* ou *Proxy Tunnel* são ferramentas que usam o tunelamento *http* e são freqüentemente usadas para acessos indevidos a *sites* de relacionamento e a serviços de mensagem instantânea em ambientes de trabalho.

Marcação de *copyright* é a tentativa de manter ou provar a propriedade intelectual sobre algum tipo de mídia, seja esta eletrônica ou impressa. Neste sentido, sistemas de marcação robustos (*Watermarking* robustos) são aqueles que, mesmo após tentativas de remoção, permanecem intactos. Por outro lado, sistemas de marcação frágeis (*Watermarking* frágil) são aqueles em que qualquer modificação na mídia acarreta perda na marcação. Estes sistemas são úteis para impedir a cópia ilegal. Ao se copiar um material original, o resultado é um material

[2] www.anonymizer.com

não marcado e, por conseguinte, pirata. Sistemas de marcação imperceptível (*Watermarking* imperceptível) são aqueles em que as logomarcas dos autores encontram-se no material, mas não são diretamente visíveis. Em contrapartida, marcação visível (*Watermarking* visível) é aquela em que o autor deseja mostrar sua autoria a todos que observarem sua criação. Um exemplo desta última forma é formado pelas imagens disponibilizadas na biblioteca do Vaticano[3]. [Pfitzmann, 1996 apud Rocha, 2003, p.9] Segundo [Mintzer et al., 1996], naquela biblioteca as imagens possuem um sistema de marcação digital visível, como pode ser observado na figura 3.2.

Figura 3.2: Exemplo de *marcação visível*.

Fonte: Vatican Library[4]

[3] http://bav.vatican.va
[4] http://www.vaticanlibrary.vatlib.it/BAVT/

3.2 – Um pouco de história

Esteganografia é uma prática muito antiga que foi aplicada ao longo da História para comunicações secretas entre as pessoas. Cabe lembrar que, à medida que se foi desenvolvendo a comunicação humana, foram sendo criados novos mecanismos de ocultamento e cifragem das informações que se queria transmitir secretamente sem interceptação de terceiros. Sempre que um mecanismo caía no conhecimento geral, novas técnicas cada vez mais sofisticadas foram sendo desenvolvidas. Algumas delas serão apresentadas a seguir, sendo importante observar seu aperfeiçoamento ao longo da História:

Em 480 a.C. na Grécia Antiga, um mensageiro foi escolhido para enviar uma informação secreta. Para tanto, seu cabelo foi raspado e uma informação foi tatuada em sua cabeça. Após o cabelo crescer, estando a mensagem encoberta, ele pôde dirigir-se ao destinatário da mensagem, e mesmo sendo revistado a informação que levava não foi descoberta. Ao chegar no destino, seguindo a recomendação do emissor, raspou sua cabeça, novamente, para que a mensagem secreta fosse visualizada.

Também, na Grécia Antiga, um homem desejoso de enviar uma mensagem secreta matou uma lebre e escondeu a mensagem em suas entranhas. Entregou o animal para um mensageiro que se disfarçou de caçador para conduzi-la até seu destino sem que fosse descoberta a mensagem.

Na China Antiga, as mensagens eram escritas sobre uma seda fina. Em seguida, a seda era amassada e coberta com cera na forma de uma pequena bola e o mensageiro a engolia antes de se dirigir ao destinatário.

Outro método, surgido no século I d.C., consistia em se utilizar tinta invisível - leite titímalo - para escrever mensagens. Quando o leite secava, a mensagem ficava “invisível”. Para visualizar esse tipo de mensagem, o receptor aquecia suavemente o papel, queimando a tinta, e assim aparecendo a mensagem escrita na cor marrom.

O historiador da Grécia Antiga, Enéias, o Tático, teve a idéia de enviar uma mensagem secreta fazendo minúsculos furos em certas letras de um texto

qualquer. A sucessão destas letras marcadas fornecia o texto secreto. Dois mil anos mais tarde, remetentes ingleses empregaram o mesmo método, não para garantir o segredo de suas cartas, mas para evitar o pagamento de taxas muito caras. Na realidade, antes da reforma do serviço postal ao redor de 1850, enviar uma carta custava cerca de um *shilling* para cada cem milhas de distância. Os jornais, no entanto, eram isentos de taxas. Graças a furinhos de agulha, os ingleses espertos enviavam suas mensagens gratuitamente. Este procedimento foi também utilizado pelos alemães durante a Primeira Guerra Mundial. Na Segunda Guerra, eles aperfeiçoaram o método, marcando letras de jornais com tintas "invisíveis". [Muller,2006]

Ainda durante a Segunda Guerra Mundial, a técnica de micropontos foi utilizada por espiões alemães. Consistia em colocar uma fotografia, uma espécie de microfilme do tamanho de um ponto (.) em uma letra de um texto. Quando o texto chegava ao receptor, a fotografia era ampliada e a mensagem aparecia claramente.

As cifras nulas (mensagens sem criptografia) foram também um método aproveitado pelos alemães na Segunda Guerra Mundial. Aparentemente, eram mensagens inocentes sobre acontecimentos comuns, ou seja, fora de suspeita e sem valor para fins militares. Um exemplo do uso de cifras nulas é visto na seguinte mensagem:

***O** **S**enhor **E**duardo **q**uer **u**sar **e**ste **s**alão **t**emporariamente. **R**elembre **o** **f**ato **o**corrido, **i**sto **p**oderia **e**stragar **r**elíquias, **f**lorais **e** **i**magens **t**alhadas. **O**brigado.*

Figura 3.3: Mensagem Inicial
Fonte: Autor

A mensagem codificada pode ser extraída pela captura de toda primeira letra de cada palavra. O resultado desse captura é a mensagem:

Figura 3.4: Resultado das cifra nula
Fonte: Autor

A esteganografia da atualidade é freqüentemente associada com a alta variedade de tecnologia onde as informações podem ser escondidas dentre de um arquivo eletrônico. Por exemplo, uma imagem no formato *jpg* pode esconder um arquivo de texto. Essa nova prática, geralmente, é feita pela substituição dos *bits* menos significativos ou mais redundantes pertencentes ao arquivo original por *bits* da mensagem ou do arquivo a ser escondido, ou seja, um tipo de informação que dificilmente os olhos e os ouvidos humanos perceberiam. Ironicamente, a esteganografia eletrônica é uma das poucas alternativas de privacidade na era da informação.

3.3 – Terminologia

O procedimento básico de ocultamento de uma informação dentro de outra informação é descrito a seguir. [Petitcolas et al., 1999]

Dado agregado (*embedded data*) - é a informação escolhida para enviar secretamente.

Portadora ou mensagem de cobertura (*cover-message*) - é uma informação ordinária que será usada para transportar o dado agregado. Essa mensagem pode ser um áudio de cobertura (*cover-audio*), um vídeo de cobertura (*cover-video*), ou uma imagem de cobertura (*cover-image*).

Estego-objeto (*stego-object*) ou estego-recipiente (*stego-carrier*) - é o estágio em que a mensagem de cobertura já apresenta o dado agregado

escondido em seu interior, com novas características imperceptíveis ao ser humano.

Estego-chave (*stego-key*) - é a chave que irá mascarar os dados agregados em uma mensagem de cobertura, transformando-a em um estego-objeto.

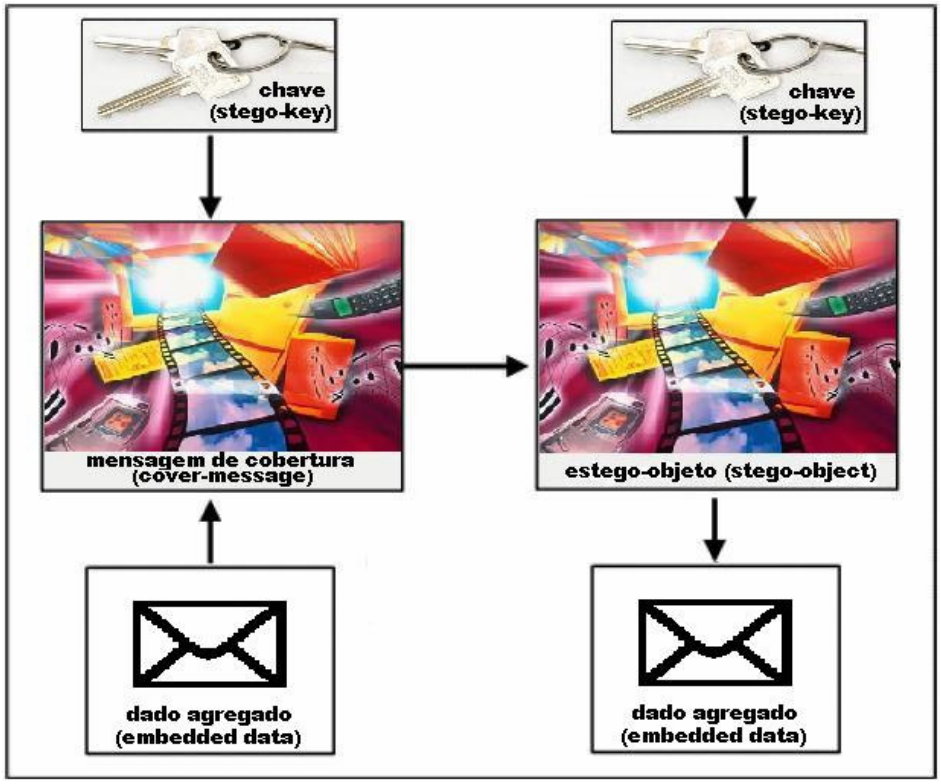

Figura 3.5: Exemplo de ocultamento da mensagem

Fonte: Autor

3.4 – Métodos de ocultamento da informação

Existem três métodos básicos de esconder os dados - por inserção, por substituição e por geração.

3.4.1- Inserção de dados

Neste método, os blocos contendo informações são inseridos dentro do arquivo escolhido para encobrir a mensagem. Para realizar essa técnica, é preciso

descobrir as áreas que o arquivo irá ignorar, para depois colocar a mensagem pretendida dentro dessas áreas. Uma vez identificadas as áreas, a mensagem pode ser partida e inserida. Dependendo do formato do arquivo, as informações podem ser escondidas entre cabeçalhos, dentro das tabelas de cores, dentro da camada de dados da imagem e em muitos outros campos. A inserção de dados aumenta o tamanho do arquivo, mas não tem nenhum efeito na representação (visual ou audível) do arquivo modificado em relação ao original.

Por exemplo, a maioria dos arquivos contém um cabeçalho que identifica o final do arquivo. Então, quando um arquivo de som está tocando, o *software* está executando esse arquivo, o que somente será interrompido quando for identificado o cabeçalho de final do arquivo, porque ele "pensa" que este é o fim do arquivo. Portanto, qualquer dado inserido após esse cabeçalho não irá afetar o som do arquivo.

Em suma, esse método adiciona dados a um arquivo, com a vantagem de não modificar ou substituir o conteúdo existente, ou seja, sem causar degradação. Entretanto, esse método apresenta a desvantagem de aumentar o tamanho do arquivo fazendo com que, em alguns casos, esse aumento torne-se perceptível.

3.4.2- Substituição ou sobrescrita de dados

Neste caso, procura-se uma informação insignificante no arquivo, a qual será usada para encobrir a mensagem, por meio de sua substituição pelo dado agregado. É preciso ter cautela na aplicação dessa técnica, pois a sobrescrita de dados mal implementada pode inutilizar ou deixar permanentemente defeituoso o estego-objeto. Para que se obtenha sucesso com a aplicação dessa técnica é necessário descobrir-se dados insignificantes do arquivo de cobertura, isto é, dados que possam ser substituídos sem causar nenhum impacto.

Por exemplo, em um arquivo de áudio cada unidade de som escutada é formada por muitos *bytes*. Se for modificado o *bit* menos significativo (*LSB- Least Significant Bit*), ele vai se modificar ligeiramente, não sendo perceptível ao ouvido humano tal mudança.

Portanto, a principal diferença entre o método de substituição e o método de inserção é que, no primeiro, o tamanho do arquivo não é alterado. Porém, existe a limitação de uma certa quantidade de dados que são considerados insignificantes dentro de cada arquivo.

3.4.3- Geração de um arquivo portador

Nas técnicas anteriormente comentadas a informação que se deseja esconder precisa de um arquivo para encobrir. Na técnica de geração, a mensagem que se deseja esconder é usada para criação de um arquivo de cobertura. A vantagem desse método em relação aos anteriores é que não existe arquivo original para encobrir a mensagem. Com isso, não existe meio de comparação binária entre o arquivo original, que pode encobrir a mensagem, e o estego-objeto, que é o arquivo já esteganografado.

Um exemplo simples da técnica de geração de arquivo é a criação de um arquivo de imagem que parece uma pintura abstrata a partir da mensagem que se deseja esconder. Isso é feito pela substituição de cada “0” por tonalidades verdes e de cada “1” por tonalidades amarelas.

Um dos problemas dessa técnica é que a criação de formas realistas, um carro, por exemplo, não podem ser facilmente geradas. Por isso geralmente são criadas formas abstratas.

3.5-Tipos de arquivo portador

Os tipos de arquivo portador (mensagem de cobertura) que serão utilizados para o ocultamento das informações apresentam diferentes formatos e cada um possui diferentes propriedades e maneiras de esconder uma mensagem, a seguir descritos.

3.5.1- Arquivo de Imagem

Para o computador, um arquivo de imagem é um arquivo comum que apresenta diferentes tipos de cor e de intensidade de luz em cada *pixel* dessa

imagem. A técnica mais aplicada para esteganografia em imagens é o LSB (*Least Significant Bit* – substituição do *bit* menos significativo). Esta técnica utiliza o último *bit* de cada *byte* da imagem, isto é, o *bit* menos significativo, para ocultar a mensagem desejada. No caso de imagens coloridas, utilizam-se 3 *bits* em cada *pixel*, sendo cada um deles pertencente a cada uma das componentes RGB (*Red, Green, Blue*). [KOBUSZEWSKI, 2004]

O melhor formato de imagem para esconder informação é o *Bitmap* (BMP) de 24 *bits*, pois as imagens deste formato possuem tamanho maior e melhor qualidade. Quando uma imagem apresenta maior qualidade e resolução, o arquivo permite que a mensagem maior possa ser encoberta.

É importante lembrar que se a estego-imagem for convertida para um formato diferente do formato em que ela foi esteganografada a informação escondida será perdida.

3.5.2- Arquivo de Som

Quando se deseja esteganografar uma mensagem dentro de um arquivo de áudio, a técnica mais utilizada é a codificação do menor *bit* (*low bit enconding*) que funciona de maneira similar à técnica LSB, usada para esteganografia em imagens. O problema desta técnica é que geralmente fica perceptível ao ouvido humano, sendo, assim, uma técnica arriscada para ocultar informações.

Spread Spectrum é outro método que adiciona ruídos aleatórios ao sinal. A mensagem secreta é escondida dentro do arquivo áudio e dispersa através do espectro da freqüência.

Além dos métodos anteriores, existe a ocultação de dados no eco (*Echo data hiding*) que utiliza os ecos existes nos arquivos de áudio para ocultar informações sigilosas. São adicionados “sons extras” para um eco dentro do arquivo de áudio. Comparado a outros, este método é considerado o melhor, pois pode melhorar a qualidade sonora do arquivo de áudio.

3.5.3- Arquivo de Vídeo

Um dos métodos mais utilizados para ocultar mensagens em arquivos de vídeo é o método DTC (*Discrete Cosine Transform* – Transformada Discreta do Cosseno), que modifica ligeiramente determinados valores de cada imagem dentro do vídeo. Com isso, apenas uma pequena quantidade de informação pode ser inserida sem que seja perceptível ao olho humano.

Outro método que foi empregado neste trabalho é a substituição de dados de algumas figuras do vídeo por partes da mensagem que se deseja esconder. Para aplicar essa técnica, deve-se considerar o número de *frames* (figuras) por segundo, pois se for feita a substituição de mais de uma figura em menos de um segundo a deformação no vídeo fica perceptível ao olho humano. Geralmente os arquivos de vídeo do formato MPEG-1, detalhados no capítulo 2, apresentam uma taxa de amostragem entre 24 e 30 *frames* por segundo. Então, nesse caso, é recomendável que o intervalo entre as figuras que serão substituídas seja maior ou igual à taxa de amostragem, ou seja, 30 ou mais *frames*. Portanto, a limitação do tamanho da mensagem a ser encoberta é baseado no tamanho em segundos do vídeo escolhido, ou seja, quanto maior o tempo de exibição do vídeo maior a tamanho da mensagem que pode ser encoberta, dando a essa técnica maior flexibilidade.

3.6- Esteganálise

Ao campo das pesquisas relacionado às tentativas de descobrir mensagens secretas dá-se o nome de esteganálise [Rocha, 2003], ou seja, detectar se existe esteganografia dentro de um arquivo.

As pesquisas de esteganálise não tentam extrair a mensagem encoberta em um estego-objeto, apenas buscam descobrir sua existência. Essa limitação ocorre em virtude da grande variedade de técnicas de ocultamento e, em alguns casos, a adição do uso de criptografia aumenta a segurança da informação secreta e dificulta sua decifração.

Um método mais simples de detecção é a comparação entre dois arquivos: um suspeito e o outro uma cópia similar do arquivo suspeito encontrada na *Internet*. Por exemplo, temos dois arquivos de imagem no formato *GIF*, que visualmente são idênticos e apresentam o mesmo comprimento e largura. Entretanto, se os dois apresentarem tamanhos diferentes, existe a possibilidade de que o arquivo de maior tamanho tenha uma mensagem encoberta.

Outra maneira de detecção é a utilização de um programa editor de hexadecimal. Por meio dele, uma comparação é feita entre dois arquivos (um suspeito e uma cópia similar do arquivo). Essa comparação é feita usando-se *checksum*. Isto quer dizer que o programa analisará a estrutura em hexadecimal dos dois arquivos e, caso as estruturas apresentem divergências em algumas seqüências de *bits*, ficará clara a existência de esteganografia.

Além dessas técnicas, existem *softwares* que detectam e destroem informações ocultadas. Dentre as ferramentas mais conhecidas de esteganálise pode-se citar:

Stegspy – que atualmente consegue identificar a presença de esteganografia dos seguintes programas: *Hiderman, JPHideandSeek, Masker, JPegX, and Invisible Secrets.*

Stego Suíte – desenvolvida pela *WetStone,* essa ferramenta é a mais completa e moderna ferramenta para investigação, detecção, análise e recuperação de mensagem secreta por meio de decifração.

Stegdetect – ferramenta para detecção de mensagem encoberta pelos seguintes *softwares: Jsteg, JP Hide and Seek, Invisible Secrets,* versões antigas do *Outguess, F5, appendX,* e *Camouflage.*

Capítulo 4 – Criptografia

4.1- Definições

A palavra criptografia é uma derivação das palavras gregas *kryptós,* que significa escondido ou oculto, e *graphien,* que significa escrita ou desenho. Portanto, o termo criptografia pode ser entendido como o estudo dos princípios e das técnicas que possibilitam que uma informação sigilosa torne-se incompreensível, permitindo que somente pessoas autorizadas consigam decifrá-la e compreendê-la.

A Criptoanálise, do grego: "*kryptós*" (oculto) e "*analyein*" (desfazer), é a ciência (e/ou arte) que estuda os princípios, processos e métodos para desvendar os segredos dos sistemas criptográficos existentes (no jargão técnico da área, "quebrar" os sistemas), objetivando ganhar acesso ou mesmo capacidade de alterar as informações ou dados pretensamente protegidos. [Almeida, 2006]

A criptografia computacional objetiva garantir que uma informação sigilosa apresente as seguintes características:

- Confidencialidade – a informação não estará disponível e não será divulgada a pessoas, entidades ou processos sem autorização;
- Integridade – os dados não serão modificados ou deformados, seja por ação intencional ou acidental;
- Autenticidade – a origem dos dados é legítima, ou seja, eles são válidos e precisos;
- Disponibilidade – a informação estará sempre à disposição de usuários com permissão, negado o acesso àqueles que não possuem autorização e tentam burlar a sua segurança.

4.2- Terminologia

Alguns termos básicos (primitivas) serão definidos a seguir para o melhor entendimento de como funciona a criptografia.

- Texto Claro (*Plaintext*) - é qualquer tipo de informação em sua forma original e compreensível ao ser humano. Exemplificando: um documento escrito, uma imagem ou um arquivo executável;
- Texto Cifrado ou Criptograma (*Chipertext*) - é uma mensagem em sua forma cifrada, isto é, com significado incompreensível ao ser humano;
- Cifrar (*Encryption*) - é o processo de transformar uma mensagem em texto claro para um texto cifrado. Com isso, ela fica ilegível, garantindo que somente pessoas autorizadas a compreendam;
- Decifrar (*Decryption*) - é o processo inverso da cifragem; transforma um texto cifrado (criptografado) para sua forma original (texto claro), inteligível;
- Chave (*Key*) - é a proteção dos dados. Ela é necessária para o processo de decifragem de um texto criptografado. Para obter melhor desempenho na cifragem é recomendável que a chave seja bem guardada e não seja fraca, de fácil suposição;
- Criptoanalista (*Cryptanalyst*) - é a pessoa que procura fraquezas em métodos de cifragem e que busca sempre maneiras de quebrar o processo de criptografia;
- Criptossistema – é um conjunto de primitivas usadas para se construir um sistema de segurança da informação.

Na Figura 4.1 pode ser visto o funcionamento genérico de um criptossistema.

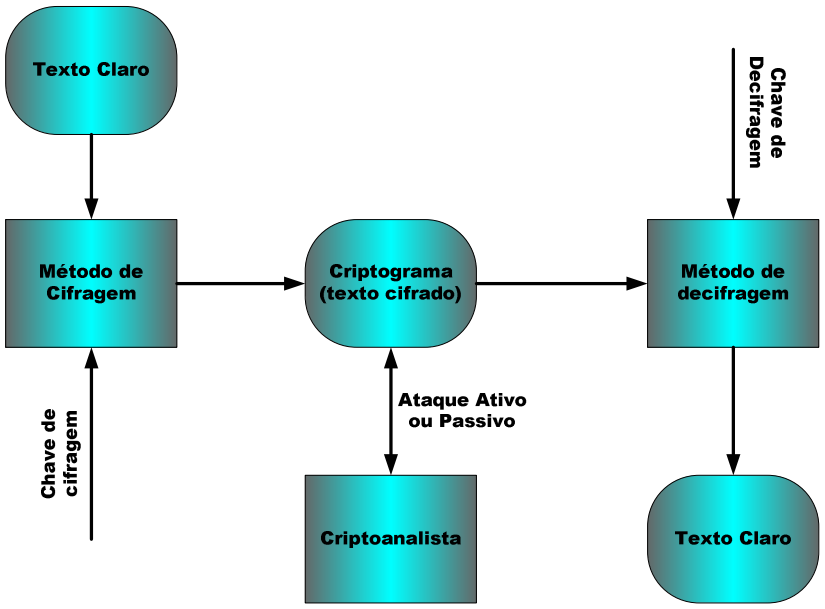

Figura 4.1: Modelo de um Criptossistema

Fonte: Autor

4.3- Aspectos Históricos

Ao longo da História, a criptografia foi sendo informalmente utilizada por pessoas que queriam se comunicar em sigilo. Os grupos militares foram os que mais adotaram o uso dessa técnica, visando à segurança das informações, principalmente em períodos de combate.

Segundo Kahn[5], o primeiro exemplo documentado da escrita cifrada aconteceu em uma vila egípcia perto do rio Nilo chamada Menet Khufu. Khnumhotep II era um arquiteto do faraó Amenemhet II. Ele construiu alguns monumentos para o faraó, os quais precisavam ser documentados. Nem é preciso dizer que aquelas informações, escritas em tabletes de argila, não deveriam cair no domínio público. [TKOTZ, 2006]. O escriba de Khnumhotep II teve a idéia de substituir algumas palavras ou trechos de texto destes tabletes. Caso o

[5] David Kahn, "*The Codebreakers*", Macmillan, 1967

documento fosse roubado, o ladrão não encontraria o caminho que o levaria ao tesouro - morreria de fome, perdido nas catacumbas da pirâmide [TKOTZ, 2006].

Em 50 a.C., um dos primeiros casos de uso de técnicas de criptografia foi empregado na Roma Antiga. O Imperador Júlio César utilizou um modelo básico de cifragem para comunicar seus segredos políticos e militares, o qual ficou conhecido como Algoritmo de César (Substituição cíclica do alfabeto). No caso em questão, a técnica de cifragem ocorreu com o deslocamento das letras do alfabeto em três posições em relação à sua inicial.

Para exemplificar a técnica empregada, será utilizado o alfabeto da língua portuguesa, fazendo-se um deslocamento de posições igual a 4 e a palavra “teste”. Cada letra será deslocada, portanto, em 4 posições no alfabeto, obtendo como resultado a palavra “zixzi”, isto é, o “t” foi substituído pelo ”z”, o “e” pelo “i” e o “s” pelo “x”. Esse processo pode ser visto na Figura 2.2.

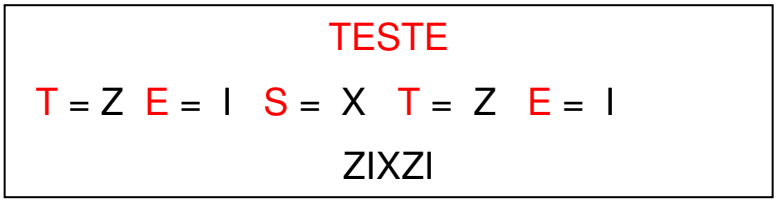

Figura 4.2: Exemplo do Algoritmo de César

Fonte: Autor

Esse algoritmo, apesar de apresentar um processo de simples execução, ilustra a um nível fundamental o funcionamento e o objetivo da criptografia. Foi a partir desse algoritmo que, durante muito tempo, Júlio César enganou seus inimigos, até que fosse descoberto.

Os deslocamentos de posição feitos no algoritmo de César podem ser a base para um melhor entendimento do conceito atual de chave criptográfica. O valor de deslocamento é mais essencial do que o próprio algoritmo, pois sem o conhecimento desse valor jamais se consegue decifrar a mensagem.

Durante a Segunda Guerra Mundial os alemães usaram uma máquina denominada Enigma, que oferecia mais de 712 milhões de possibilidades de

chave, o que parecia inquebrável para os padrões de segurança da época. Entretanto, esse mecanismo foi quebrado pelo matemático polonês Marian Rejewski, que se baseou apenas em textos cifrados interceptados e em uma lista de três meses de chaves diárias obtidas por meio de um espião. Alan Turing, Gordon Welchman e outros, em Bletchley Park, Inglaterra, deram continuidade à criptoanálise do sistema Enigma. [TKOTZ, 2006].

Na década de 1960, o Dr. Horst Feistel, liderando um projeto de pesquisa no *IBM Watson Research Lab,* desenvolveu a cifra *Lucifer.* Alguns anos depois, esta cifra serviu de base para o DES e outros produtos cifrantes, criando uma família conhecida como "cifras Feistel". [TKOTZ, 2006]

Em 1974, a IBM apresentou a cifra *Lucifer* ao NBS (*National Bureau of Standards*) o qual, após avaliar o algoritmo com a ajuda da NSA (*National Security Agency),* introduziu algumas modificações (como as Caixas S e uma chave menor) e adotou a cifra como padrão de encriptação de dados para os EUA - o FIPS PUB-46 - conhecido hoje como DES (*Data Encryption Standard*). [TKOTZ, 2006]

Os algoritmos mais modernos usam os mesmo princípios do algoritmo de César, onde o algoritmo é conhecido, mas a chave não. É importante salientar que o tamanho da chave é, também, um fator importantíssimo na segurança do algoritmo. Uma chave de dois dígitos possui 100 possibilidades de combinações, uma chave de 6 dígitos possui 1 milhões de possibilidades. Conseqüentemente, quanto mais possibilidades de combinações, mais tempo um criptoanalista levará para decifrar um texto. [Hinz, 2000]

4.4 – Algoritmos de Criptografia

Os algoritmos de criptografia estão divididos, basicamente, em duas categorias: algoritmos de Chave Privada ou Simétrica e algoritmos de Chave Pública ou Assimétrica.

4.4.1 – Chave Privada ou Simétrica

Chave simétrica é uma chave-única (*single-key*) no processo de criptografia. Em outras palavras, a chave usada para cifragem de um texto claro é a mesma chave usada para decifragem de um criptgrama. Esse método tem como finalidade a conversão de um bloco de tamanho fixo de texto claro para um bloco de texto cifrado, de mesmo tamanho, usando uma chave de tamanho fixo.

Dentre alguns exemplos de algoritmos de chave privada, pode-se citar:

- *DES* (*Data Encryption Standard*), que será detalhado no item 4.6;
- *Triple* DES, uma variação do DES;
- *IDEA* (*International Data Encryption Algorithm*), algoritmo forte e resistente a várias formas de criptoanálise;
- RC2, duas vezes mais rápido que o algoritmo DES;
- RC4, dez vezes mais rápido que o algoritmo DES.

A seguir serão apresentados os três processos básicos aplicados em todos os algoritmos simétricos: a substituição, a permutação e o XOR (ou exclusivo).

4.4.1.1 – Substituição

A substituição consiste em tomar uma letra e trocá-la por outra. Para que esse procedimento ocorra, deve haver um mapeamento *um a um* entre as letras. Por exemplo, se a letra “A” for substituída por “W”, para o método funcionar nenhuma outra letra poderá ser substituída por “W”. Caso essa condição seja ignorada, tanto a letra “A” como uma outra poderão ser substituídas por “W”. Mesmo assim, o estágio de cifragem poderá ser concluído com sucesso. Entretanto, no processo inverso, de decifragem, não será possível determinar precisamente qual letra será substituída por “W”.

Este método é considerado fraco porque cada letra sempre será substituída pela mesma referência. Na língua portuguesa, por exemplo, determinadas letras são empregadas com maior freqüência do que outras, o que pode ser usado para descobrir o mapeamento daquelas com maior freqüência.

4.4.1.2 – Permutação

Com o uso da substituição, anteriormente comentado, uma letra pode ser trocada por outra, baseando-se o método em um mapeamento ditado por uma chave. Já com a permutação, todas as letras de uma mensagem presente em um texto claro se mantêm, apenas sendo movidas para diferentes posições, ou seja, elas são embaralhadas, como se fosse um anagrama. Em um exemplo bastante simplificado, a palavra “segurança” pode ser cifrada “aleatóriamente”, por permutação para “çnagasure”. Porém, para uma decodificação precisa de um texto cifrado, a permutação não pode ser aleatória, sendo necessário o uso de uma chave para ditar a movimentação das letras.

Na permutação, a chave mostra especificamente onde cada letra da mensagem do texto claro deve se posicionar no texto criptografado. Retomando o exemplo da palavra “segurança”, suponha-se que a chave dite as seguintes posições 8, 7, 9, 3, 6, 1, 4, 5 e 2. A chave vai indicar as posições em que as letras devem ser reorganizadas. No processo de cifragem, a letra da posição 8 vai ser reposicionada na primeira posição e as letras seguintes se reposicionam conforme a seqüência ditada pela chave. Com isso, o resultado da cifragem do texto claro “segurança” é o texto criptografado “çnagasure”.

No processo inverso - a decifragem - após a entrada da chave correta, a letra “ç” seria reorganizada para a oitava posição e assim por diante.

4.4.1.3 – XOR

O Ou Exclusivo (*XOR* - *Exclusive Or*) é um método mais robusto comparado à permutação e à substituição. Esse método é executado a nível binário (0 e 1), logo, a mensagem a ser criptografada deve ser convertida para uma equivalente em binário.

O XOR é uma operação matemática que toma dois números binários como entrada e resulta em uma saída. Caso os dois números possuam o mesmo valor, o resultado da operação é zero e, se eles forem diferentes, o resultado é 1. Então, ao utilizar a técnica do XOR para cifragem, a mensagem do texto claro irá realizar essa operação com a chave criptográfica, tendo como resultado o criptograma. Na

decifragem, ao realizar a mesma operação entre criptograma e chave, irá resultar na reconstrução da mensagem do texto claro.

Por exemplo, será feita a operação da palavra “teste” com a chave “11010011”. Primeiro passo deve-se converter “teste” para o valor equivalente em binário utilizando a tabela ASCII[6].

Tabela 4.1: Versão simplificada da tabela ASCII

Fonte: Autor

Letra	Binário Equivalente
T	01110100
E	11010011
S	01110011

Após a conversão para o binário equivalente, é feita a operação XOR entre ele e a chave “11010011”, como pode ser visto a seguir:

	T	E	S	T	E
	01110100	01100101	01110011	01110100	01100101
Chave:	11010011	11010011	11010011	11010011	11010011
Resutado:	10100111	10110110	10100000	10100111	10110110

Figura 4.3: Exemplo da Operação XOR

Fonte: Autor

[6] http://www.lookuptables.com/

Se a operação XOR for novamente feita, agora utilizando o resultado e a chave, a mensagem será decifrada, necessitando apenas consultar na tabela ASCII os valores binários referentes para identificar as letras.

Em suma, esse método para desenvolvimento da cifragem no algoritmo de chave simétrica é bastante eficaz. A técnica XOR, como os demais procedimentos básicos desse algoritmo, garante que a chave é o principal fator da segurança.

4.4.2 – Chave Pública ou Assimétrica

A cifragem assimétrica utiliza duas chaves, uma pública e uma privada. O que for criptografado com uma chave somente será decriptografado com a outra chave. Então, como o próprio nome diz, chave pública é uma chave para a qual não há necessidade de sigilo pleno, pois ela é utilizada para a cifragem da mensagem. Já a chave privada, é uma chave que deve ser guardada em segredo, pois ela será o único meio de decifragem do criptograma.

Em comparação com o uso da chave simétrica, que utiliza a mesma chave nos processos de criptografia e decriptografia e que requer um canal seguro para o envio da chave ao destinatário, a chave assimétrica não precisa desse canal, pois o destinatário possui uma chave privada, para decifragem. Entretanto, o emissor da mensagem deve ser confiável, pois, como a chave de cifragem é pública, qualquer pessoa que tiver posse dessa chave poderá cifrar uma mensagem e enviar para o destinatário. Com isso, é necessário que haja confiabilidade na fonte emissora para o não recebimento de mensagens falsas e desnecessárias.

Surge, então, uma dúvida: Como provar que uma fonte é confiável? O procedimento é simples. Basta que a fonte emissora cifre uma mensagem com a chave privada e o receptor a decifre com a chave pública. Assim, mesmo que uma pessoa que venha a interceptar a mensagem e que tenha conhecimento da chave pública, possa decifrá-la, o destinatário identificará a confiabilidade do emissor, podendo este enviar outras mensagens cifradas a partir da chave pública, para a real transmissão segura da informação sigilosa.

Dentre alguns exemplos de algoritmos de chave pública pode-se citar:

- *RSA* (*Rivest, Shamir, Adleman*) – algoritmo de chave assimétrica mais popular, de fácil entendimento e desenvolvimento;
- *DSA* (*Digital Signature Algorithm*) – algoritmo de assinatura digital, que utiliza função *hash*[7].

4.4.3 – Assinatura Digital

As assinaturas digitais desempenham as mesmas funções para documentos digitais assim como as assinaturas comuns funcionam para documentos impressos. A assinatura é uma mensagem não falsificável, garantindo que uma certa pessoa ou entidade escreveu ou está de acordo com o documento no qual a assinatura está colocada. [Fahn, 1992]

O algoritmo responsável pela assinatura digital transforma uma informação de qualquer tamanho, usando uma chave privada em uma assinatura. Sendo esta, no âmbito computacional, é praticamente impossível que coincidia com outra.

Esse tipo de assinatura garante dois aspectos - a autenticidade da origem do documento e a integridade de seu conteúdo.

Um sistema de autenticação por assinatura digital apresenta dois módulos básicos. O primeiro, que define o método de assinatura impossível de ser fraudado e o segundo, que define o método de certificação.

4.5 - Diferenças entre os termos esteganografia e criptografia

Os termos "esteganografia" e "criptografia" estão relacionados a estudos voltados para a segurança da informação e têm como principal objetivo garantir o sigilo de uma informação escolhida.

Apesar do objetivo em comum, são técnicas bem distintas. A criptografia garante a segurança de uma mensagem de texto claro, a partir do uso de técnicas de cifragem que o transformam em um texto ilegível. Já a esteganografia, conforme foi detalhado no capítulo 3, utiliza técnicas de ocultação para garantir a

[7] *Hash* é um algoritmo de transformação de um texto claro que se caracteriza por irreversível. Ele produz um criptograma de tamanho fixo menor a mensagem original. Dessa forma, o texto criptografado não pode ser decifrado. O hash é muito útil para armazenamento de senhas e para assinaturas digitais porque não armazena a chave, logo não a decifragem.

confidencialidade da informação, quer dizer, ela esconde a mensagem dentro de um arquivo portador, sem que as modificações nele feitas sejam perceptíveis à visão humana. Portanto, ao analisar os procedimentos de cada uma delas, fica evidente que a criptografia não oculta a informação, apenas a torna ilegível e, também, que a esteganografia objetiva o mascaramento da informação, sem afetar a qualidade do arquivo utilizado como cobertura, ou seja, torna a mensagem invisível.

Outra forma de identificar as diferenças entre os dois termos são os objetivos propostos pelas técnicas de esteganálise e de criptoanálise. A primeira tem como objetivo detectar a presença de esteganografia em um arquivo aparentemente normal. Já a segunda técnica tem como meta decifrar um texto inteligível, isto é, recuperar a mensagem original.

4.6 - Criptografia DES

DES, abreviatura de *DATA ENCRYPTION STANDARD*, é o nome do Padrão de Processamento de Informação Federal (*Federal Information Processing Standard* – FIPS), número 46-1, ou FIPS 46-1, (FIP461), que descreve o Algoritmo de Criptografia de Dados (*Data Encryption Algorithm* – DEA), originalmente desenvolvido pela IBM, a partir do seu algoritmo conhecido como *Lucifer*, e adotado como padrão nos Estados Unidos, em 1977, pelo antigo NBS (*National Bureau of Standards*), hoje denominado NIST (*National Institute of Standards and Technology*), com a substancial ajuda da NSA (*National Security Agency*).[Almeida, 2006]

4.6.1 – Descrição do algoritmo

O DES é um algoritmo simétrico (o processo usado na cifragem é o mesmo na decifragem) que adota uma técnica conhecida como cifra de bloco. Essa técnica consiste na separação da mensagem original em fragmentos de tamanho fixo (64 bits equivalentes a 8 caracteres) denominados blocos. Na cifragem de cada bloco é utilizada uma chave composta por 56 bits utilizados pelo algoritmo, e

por 8 bits para verificação de erro de paridade[8]. Cada criptograma resultante desse processo tem exatamente o mesmo tamanho do bloco original. Cada bloco passa pelo método de codificação, o que pode ser visto na Figura 4.3.

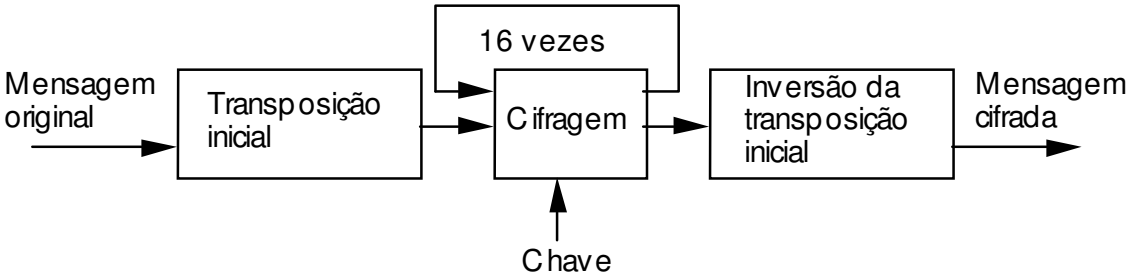

Figura 4.4: Processo de cifragem do DES

Fonte: [WEBER, 2006]

O algoritmo começa realizando uma permutação inicial (IP) sobre o bloco de 64 bits, em seguida realiza 16 vezes o passo da cifragem e conclui o processo realizando uma permutação final (IP-1), que é inversa a IP. Durante as transposições IP e IP-1 a chave criptográfica, de 64 bits, não é utilizada. Os subitens a seguir serão descritos de acordo com HINZ (2000, p20-23).

4.6.1.1 – Permutação Inicial (IP)

A permutação inicial objetiva fazer o deslocamento dos *bits* do bloco para novas posições de acordo com a tabela 4.2.

Tabela 4.2: Permutação Inicial (IP)

Fonte: Autor

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

[8] A verificação de erros por paridade é uma prática comum em circuitos computacionais para detectar erros na transmissão de *bits*. No caso do DES, cada *bit* dos 8 restantes é definido como 0 ou 1 para manter a paridade de cada *byte* da chave ímpar, ou seja, com um número ímpar de "1"s em cada *byte*. Assim, ao receber uma nova chave, o computador poderá verificar se cada *byte* desta possui um número ímpar de "1"s . Caso contrário, houve algum tipo de erro na transmissão e o processo é abortado.

Nessa tabela, o *bit* da posição 58 será deslocado para a primeira posição, o *bit* da posição 50 para a segunda e assim por diante, até a última posição receber o *bit* da posição 7.

4.6.1.2 – Etapas do processo de cifragem

São realizadas 16 etapas idênticas no processo de cifragem de um bloco. A cada etapa do algoritmo é executada uma função *F,* função cifra. Será adotada, a título de exemplo, a primeira etapa. Antes de entrar nessa função, o bloco de 64 *bits* é dividido em dois blocos menores de 32 *bits* denominados bloco L_0, o da esquerda, e R_0, o da direita. Na entrada da função, são recebidos os blocos L, R e uma subchave.

A função *F* efetua as seguintes operações a cada etapa:

1. Transformação da chave criptográfica para 48 *bits*, que será detalhada no subitem 4.6.1.4;
2. Expansão do bloco R_0 de 32 para 48 *bits* por meio de uma permutação denominada *Expansion Permutation, o* que será detalhado no subitem 4.6.1.2.1;
3. XOR ("ou exclusivo") entre bloco R resultante da operação de expansão e chave criptográfica transformada para 48 *bits*;
4. O bloco R_0 resultante do XOR terá seus dados enviados para oito *S-Boxes* produzindo no total 32 *bits* subitem 4.6.1.2.2;
5. O bloco R_0 resultante é permutado, agora, em uma *P-Box.*

Ao final da primeira etapa, na saída da função *F*, é aplicado XOR entre o bloco L_0, ainda intacto, e o bloco R_0, resultante da permutação P-Box. Após essa operação, o bloco L_0 resultante XOR será deslocado para a entrada direita ($R_1 = L_0$ resultante XOR) da função *F* e o bloco R_0 para a entrada esquerda dessa função, ou seja, na etapa seguinte o bloco L_1 apresentará os valores do bloco R_0 anterior e os blocos R_1 apresentarão os valor do bloco L_0 resultante XOR.

Na Figura 4.4 pode ser visualizado o fluxograma do algoritmo *DES*.

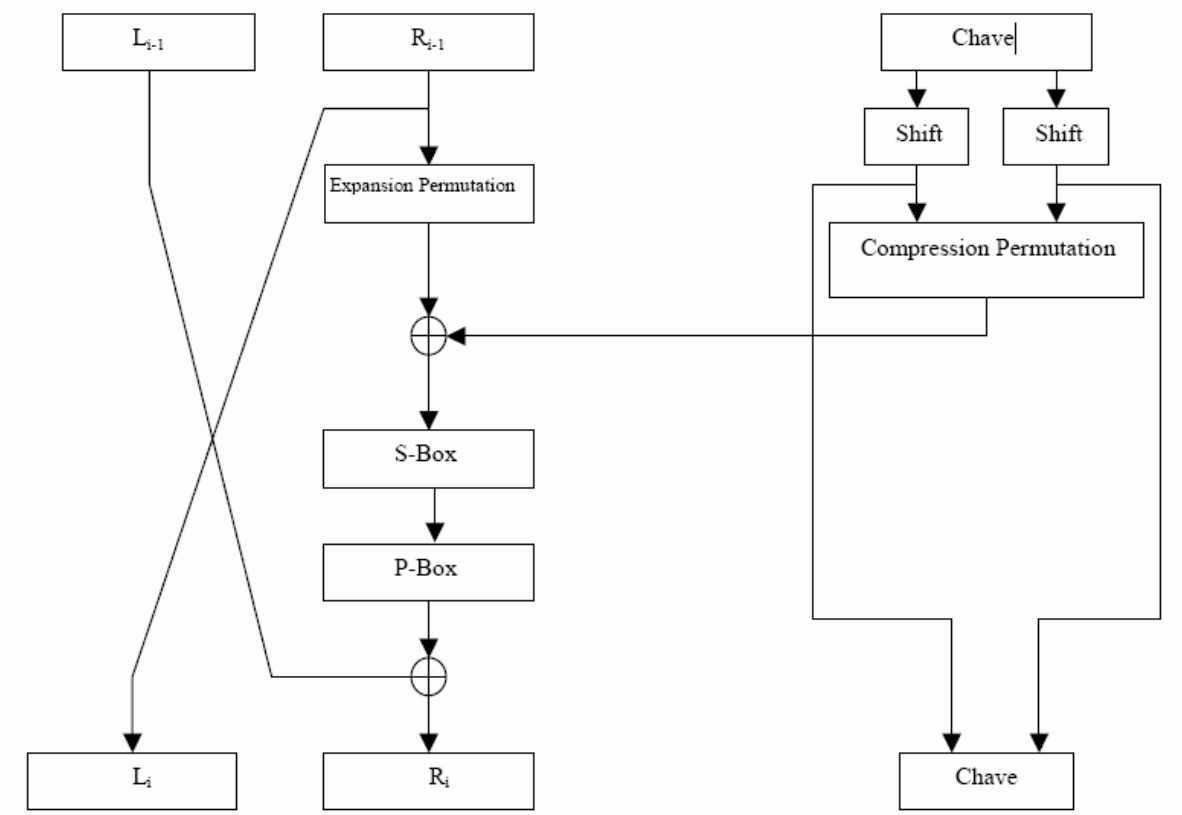

Figura 4.5: Fluxograma do DES

Fonte: [Hinz, 2000]

4.6.1.2.1 – *Expansion Permutation*

Utiliza uma função E, que expande o tamanho do bloco R de 32 para 48 *bits,* o que equivale ao tamanho da chave criptográfica após a transformação que será comentada no item 4.6.1.2.4.

Para cada seqüência de 4 *bits* presentes no bloco de entrada, o primeiro e o quarto *bits* passarão a representar, cada qual, dois *bits* do bloco de saída, e o segundo e terceiro *bits* da entrada permanecerão representando somente um *bit* cada. Esse procedimento pode ser mais bem entendido na Figura 4.6.

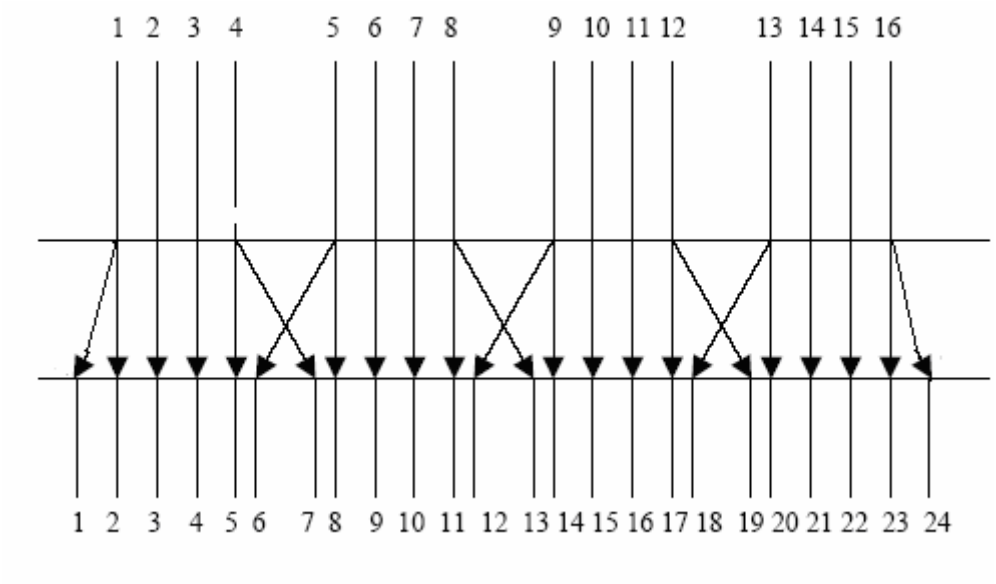

Figura 4.6: Expansion Permutation
Fonte: [Hinz, 2000]

No bloco R inicial há 32 *bits* que foram divididos em 8 seqüências de 4 *bits.* Em cada seqüência foram adicionados dois *bits* totalizando 6 *bits.* Dessa forma, o bloco final apresentará 48 *bits,* sendo 12 resultados da expansão.

4.6.1.2.2 – *S-Boxes*

S-Boxes são caixas de substituição não-lineares que visam emaranhar o texto cifrado para que se torne mais difícil a sua decifragem. O funcionamento de uma *S-Box* se baseia na simples troca de posição dos *bits* de entrada. [Hinz, 2000]

No DES são utilizados 8 *S-Boxes* distintos, que recebem como entrada 6 *bits* e devolvem na saída 4 *bits.*

Cada *S-Box* é uma matriz de 4 linhas por 16 colunas. O resultado final desta operação são oito blocos de 4 *bits* que são agrupados em um único bloco de 32 *bits*.

As 8 *S-Boxes* do *DES* podem ser visualizadas a seguir:

Tabela 4.3: S-Box - Função S1

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabela 4.4: S-Box - Função S2

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabela 4.5: S-Box - Função S3

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabela 4.6: S-Box - Função S4

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabela 4.7: S-Box - Função S5

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tabela 4.8: S-Box - Função S6

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabela 4.9: S-Box - Função S7

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabela 4.10: S-Box - Função S8

Fonte: Autor

Linhas	Colunas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

4.6.1.2.3 – *P-Boxes*

Dentro das *P-Boxes,* os *bits* são somente permutados para uma outra posição, com a característica de que nenhum *bit* é permutado mais de uma vez e todos os *bits* são permutados pelo menos uma vez. A tabela 4.3 mostra a *P-Box* adotada no DES. [Hinz, 2000]

Tabela 4.11: Caixa de Permutação (P-Box) (sobre 32 bits)

Fonte: Autor

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

4.6.1.3 – Permutação Final (IP^{-1})

A permutação final é o inverso da permutação inicial, os dois blocos da saída da décima-sexta etapa da cifragem são concatenados de forma que possam ser usados na entrada da permutação, visto na tabela 4.4. [Hinz, 2000]

Tabela 4.12: Permutação Final (IP-1)

Fonte: Autor

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

4.6.1.4 – Tratamento da chave

A chave de 64 *bits* do *DES* tem como primeiro procedimento a redução do seu tamanho para 56 *bits*, o que é feito por meio de descarte do oitavo *bit* de cada um dos oito *bytes* presentes nessa chave.

A chave é gerada da seguinte maneira:

1. chave de 56 *bits* é dividida em duas de 28 *bits;*
2. cada chave de 28 *bits* é circularmente rotacionada para a esquerda em um ou dois *bits*, dependendo da etapa, que pode ser visualizada na Tabela 4.5;
3. 48 dos 56 *bits* são selecionados de uma tabela de permutação. Essa operação é chamada de permutação de compressão, o que pode ser visualizado na Tabela 4.6.

Tabela 4.13: Deslocamentos da chave por rodada

Fonte: Autor

Rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Desloc.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Cifragem: é feita no sentido da direita para esquerda

Decifagrem: é feita no sentido da esquerda para direita

Tabela 4.14: Permutação de compressão (reduz de 56 para 48 bits)

Fonte: Autor

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

4.6.2 – Quebrando o *DES*

O algoritmo *DES* de criptografia está bem disseminado no mundo. Durante algumas décadas, o termo criptografia era, praticamente, sinônimo do padrão *DES.* Entretanto, com o crescente avanço na área de tecnologia, este algoritmo já não é mais considerado seguro, porque está sujeito a ataques de força bruta. Em julho de 1998, uma máquina, avaliada em 220 mil dólares, chamada *EFF DES Cracker*, levou três dias para quebrar o algorítimo *DES Challenge II*. Em janeiro de 1999 quebrou o *DES Challenge III* em 22 horas usando computação distribuída

por meio de milhares de computadores espalhados pela *Internet*. Em 19 de maio de 2005, foi desaprovado seu uso pelo governo americano.

Após a análise dos principais conceitos, das técnicas de criptografia e do algoritmo DES, fica evidenciado que esta prática requer um entendimento pleno do funcionamento dos três processos básicos de cifragem (substituição, XOR e permutação), pois estes serão usados repetitivamente durante os processos de criptografia e de decriptografia.

No capítulo 5 serão comentadas as principais ferramentas utilizadas, os métodos escolhidos para a esteganografia e a criptografia, as dificuldades e as limitações encontradas durante a elaboração do protótipo desenvolvido neste projeto.

Capítulo 5 – Protótipo

A implementação do protótipo tem como objetivo final o mascaramento de uma mensagem ou um arquivo por meio de sua inserção dentro de outro arquivo de vídeo comum no formato MPEG-1, relatado no capítulo 2. Para esse efeito são aplicadas as técnicas de esteganografia e criptografia *DES*, explanadas, respectivamente, nos capítulos 3 e 4. Com isso, a mensagem será transmitida de forma mais segura, dentro desse arquivo aparentemente ordinário. Caso haja interceptação na transmissão do vídeo a informação secreta não será perceptível à visão humana, isto é, não será revelada a terceiros.

5.1 – Ferramentas de Desenvolvimento

5.1.1- Equipamento

Para a implementação do protótipo foi empregado um computador com a seguinte configuração: processador modelo *Intel Pentium* M de 1,73 GHz; memória RAM de 1 GB DDR2 rodando a 533 MHz.

5.1.2- *Software*

5.1.2.1 – Sistema Operacional

A elaboração do protótipo foi feita no ambiente do sistema operacional *Windows XP Professional* versão 2002 contendo o *Service Pack 2*.

5.1.2.2 – Programas para visualização de vídeos (*video players*)

Para a realização da análise comparativa visual entre o vídeo na versão original e o vídeo na versão esteganografada foram empregado dois programas: o *Bsplayer* versão 1.00 RC1 da empresa *Webteh,* e o *Windows Media Player* versão 9.0 da *Microsoft.*

5.1.2.3 – Compilador

Para a compilação do código fonte, que gera o arquivo executável, foi adotada a ferramenta Dev-C++ versão 4.9.9.2 da empresa *Bloodshed Software*.

5.1.2.4 – Ferramenta de esteganálise

Para a análise comparativa da estrutura do arquivo (hexadecimal) entre o vídeo na versão original e o vídeo na versão esteganografada foi utilizada a ferramenta de edição hexadecimal denominada *Hex Workshop* versão 4.23 da empresa *BreakPoint Software*.

5.1.3 – Linguagem e Algoritmo

5.1.3.1 – Linguagem de Programação

Para a implementação do protótipo, foi selecionada a linguagem de programação estruturada ANSI C. Os fatores determinantes para a escolha dessa linguagem foram a facilidade de programação para manipulação binária e de manutenção do código fonte.

5.1.3.2 – Algoritmo

Foi empregado um algoritmo para efetuar a cifragem do arquivo ou mensagem que vai ser inserida no vídeo do formato MPEG-1, algoritmo de criptografia *DES*.

5.3 – Fluxograma

O fluxograma tem como objetivo a representação do controle de fluxo de informações entre as atividades de um sistema. Na Figura 5.1 é apresentado um fluxograma contendo todas as atividades realizadas pelo protótipo tanto para esteganografia quanto para a desesteganografia.

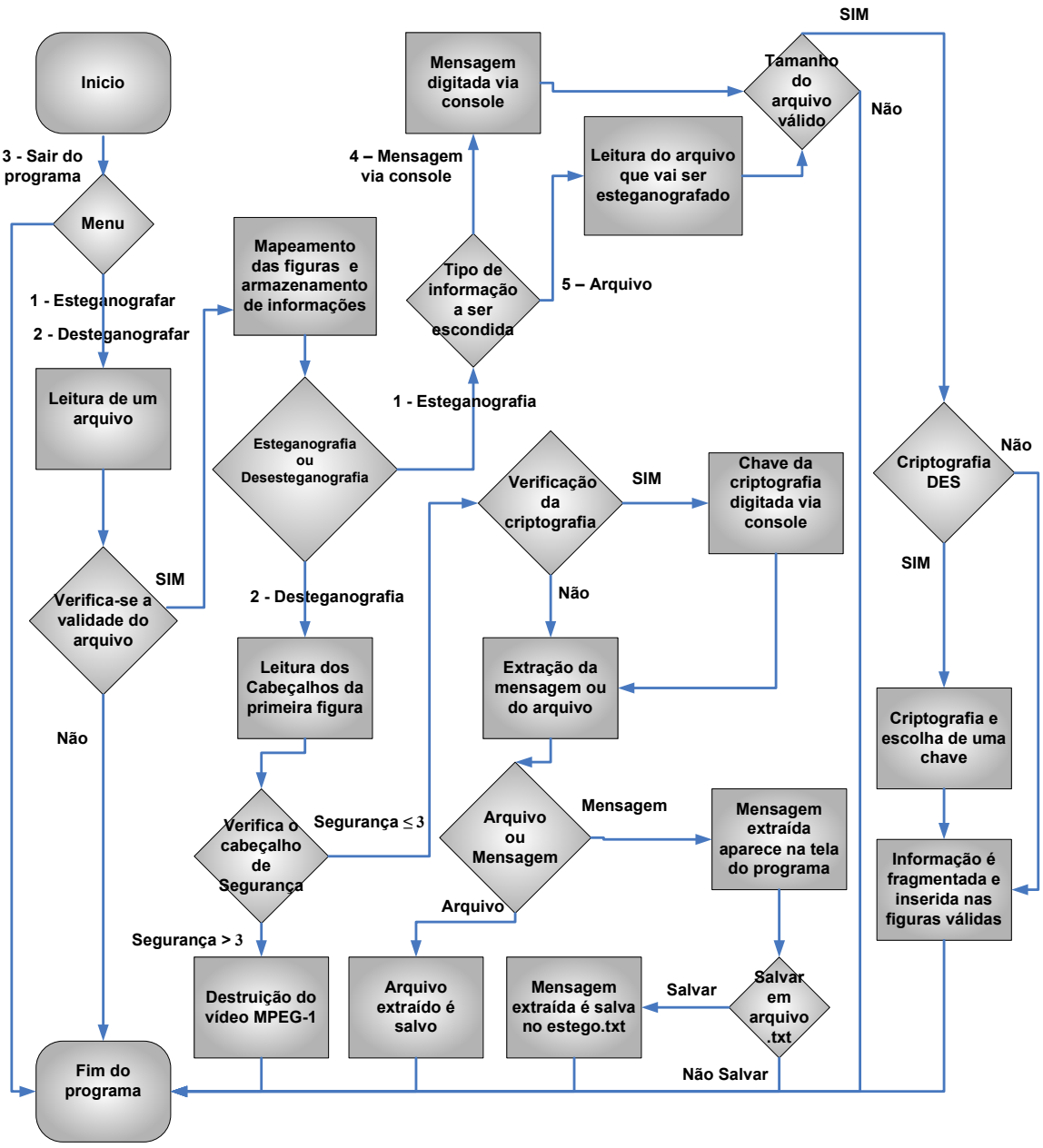

Figura 5.1 – Fluxograma do protótipo

Fonte: Autor

5.3.1 – Fluxograma do processo de Esteganografia

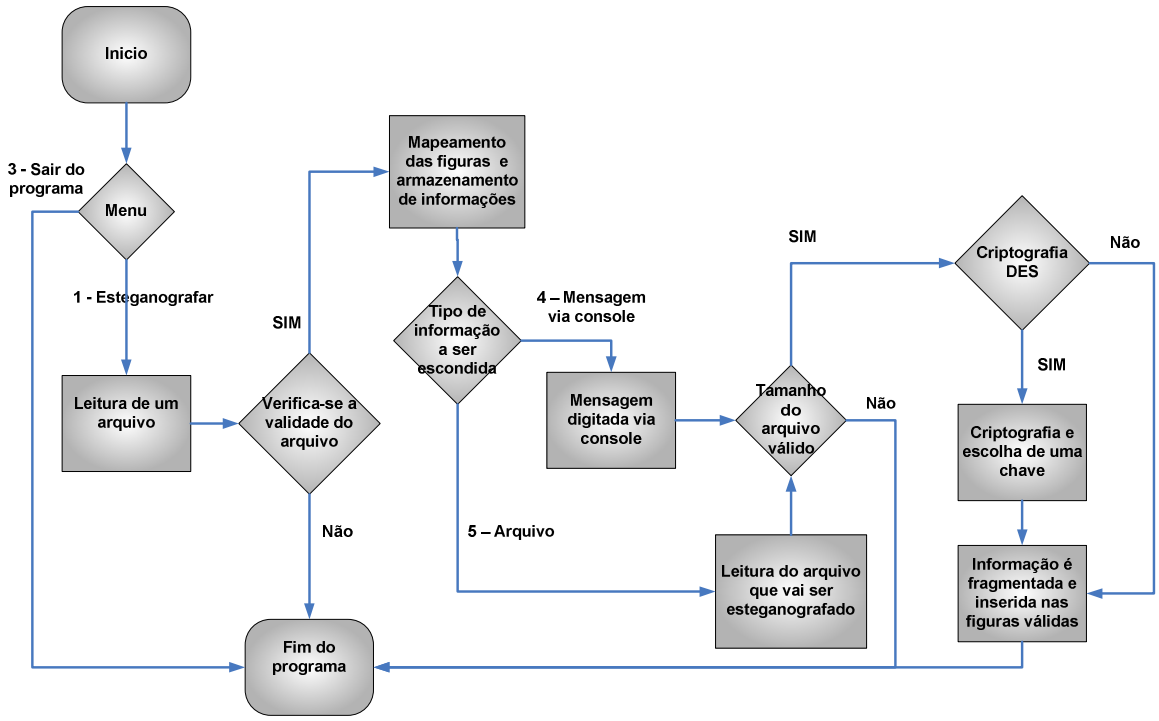

Figura 5.2 – Fluxograma do processo de Esteganografia

Fonte: Autor

Na figura 5.2 é apresentado um fluxograma de como é feito o processo de esteganografia no protótipo. O bloco inicial indica que o programa de "*stealth-inf.exe*" foi executado. Como resultado dessa inicialização é apresentado um menu com as seguintes opções: 1- Esteganografia; 2- Desesteganografia; 3- Sair do Programa.

Após a escolha da opção "1- Esteganografia", o usuário irá escolher o arquivo de vídeo do formato MPEG-1 que será utilizado como mensagem de cobertura. O programa irá verificar a validade do arquivo escolhido por meio da leitura do arquivo. Para tal feito o programa irá analisar o cabeçalho inicial e o final.

O cabeçalho inicial do formato MPEG-1 é constituído de quatro *bytes* como a seguinte seqüência de valores em hexadecimal: "00","00","01","B3". Já o

cabeçalho final desse formato de vídeo é constituído, também, por quatro *bytes* como a seqüência a seguir: “00”,“00”,“01”,“B7”. Quaisquer valores diferentes destas seqüências inicial e final nos *bytes* do arquivo será considerado pelo protótipo como arquivo inválido e o programa será encerrado.

Verificada a validade do arquivo de vídeo, o programa irá mapear as figuras 30, 60, 90, 120 assim por diante, pois a técnica de esteganografia é feita no intervalo de uma figura deformada a cada 30 figuras, sendo este o intervalo padrão adotado no protótipo. A cada figura encontrada, são armazenadas, em uma estrutura, informações referentes a sua posição no arquivo, o tamanho total e o número da figura.

Feito esse mapeamento, o programa apresentará um menu para escolha do tipo de informação que o usuário deseja ocultar (mensagem digitada via teclado ou arquivo). Após essa escolha será analisado o tamanho da informação caso ele ultrapasse o valor máximo que o vídeo MPEG-1 suporta o programa será encerrado por erro, pois o tamanho da informação é considerado inválido.

Se a mensagem possuir tamanho válido, será apresentando ao usuário um menu com a opção da utilização ou não da criptografia *DES*. Caso seja escolhida a utilização desse algoritmo de cifragem, o usuário irá escolher uma chave criptográfica de até 8 caracteres e, em seguida, cifrará o conteúdo da mensagem.

Após a cifragem do arquivo, o protótipo executará o processo de esteganografia por substituição de dados. O primeiro passo desta etapa é a inserção dos cabeçalhos essenciais para o processo de desesteganografia.

Esses cabeçalhos contêm as principais informações da mensagem ocultada, podendo ser visualizados na tabela 5.1.

Tabela 5.1: Tabela de cabeçalhos

Fonte: Autor

Cabeçalhos essenciais	Função específica
Cabestego	Verifica a presença de esteganografia no arquivo MPEG-1
Cabcripto	Verifica a presença ou não de criptografia DES
Cabsegurança	Verifica o número de tentativas de extração, feitas em um determinado vídeo esteganografado. No processo de esteganografia esse valor é determinado como "zero". Se o valor desse cabeçalho apontar 3 ou mais tentativas de extrair a informação sigilosa o arquivo de vídeo será destruído por motivo de segurança
Cabext	Armazena a extensão de um arquivo escolhido para ser ocultado. Por exemplo "pdf"
Smensagem	Armazena o tamanho em *bytes* da informação esteganografada no vídeo. Esse cabeçalho é importante porque possibilita a recuperação efetiva da mensagem

Nesse processo a mensagem vai ser dividida em blocos de tamanho igual ao tamanho disponível em cada figura pertencente a uma progressão aritmética com razão (r) igual a 30, pertencente ao número total de figuras. Essas figuras pertencentes ao conjunto da PA são denominadas figuras "esteganografáveis", ou seja, as figuras que irão receber os blocos da mensagem que se deseja esconder. Ao ser encerrada essa tarefa o programa conclui o processo de esteganografia com sucesso.

5.3.2 – Fluxograma do processo de Desesteganografia

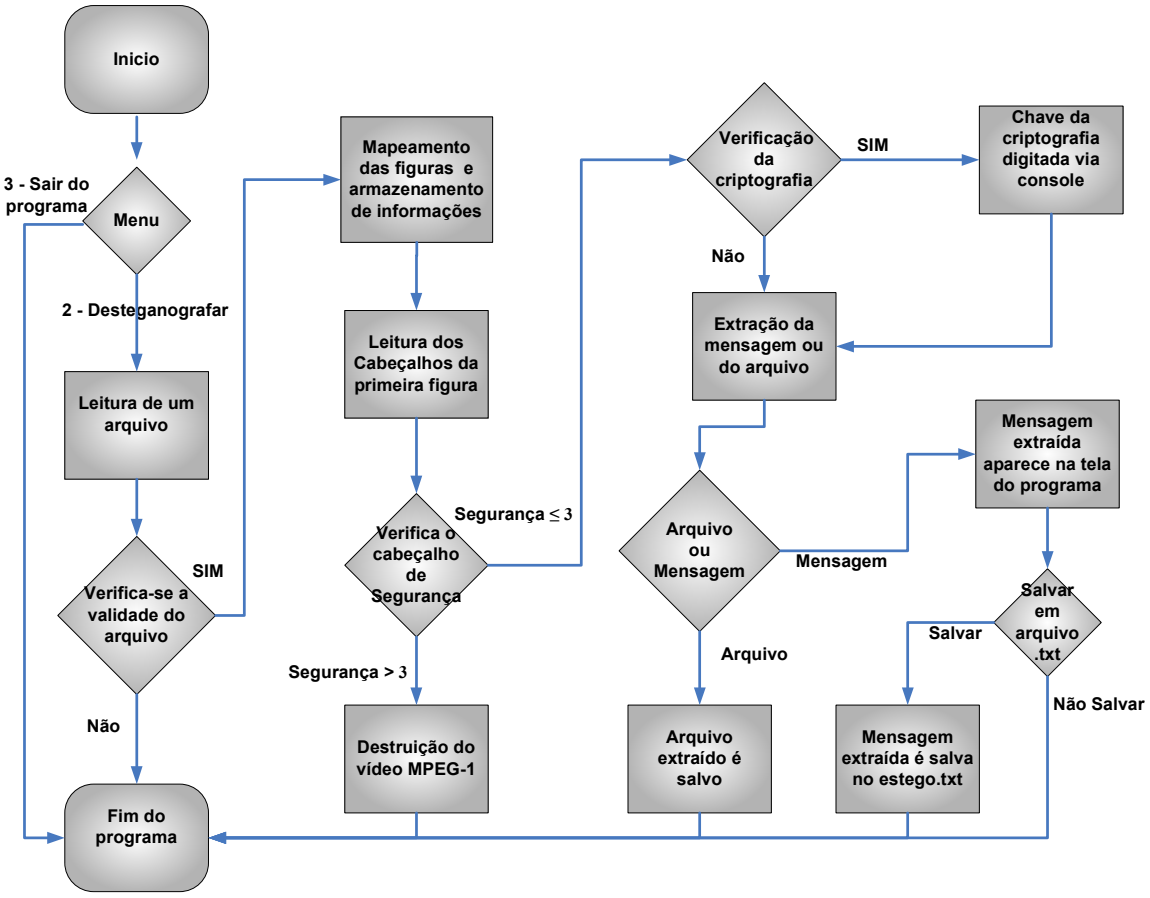

Figura 5.3 – Fluxograma do processo de Desesteganografia

Fonte: Autor

Na figura 5.3 é apresentado um fluxograma de como é feito o processo de desesteganografia no protótipo. O bloco de início indica que o programa de *"stealth-inf.exe"* foi inicializado. Como resultado dessa inicialização, oferece um menu com as seguintes opções: 1- Esteganografia; 2- Desesteganografia; 3- Sair do Programa.

Após a escolha da opção "2- Desesteganografia", o usuário irá escolher o arquivo de vídeo do formato MPEG-1 que será analisado para possível extração da mensagem (desesteganografia). O programa irá verificar a validade do arquivo escolhido por meio da leitura do arquivo. Para tal feito o programa irá analisar o cabeçalho inicial e o final.

O cabeçalho inicial do formato MPEG-1 é constituído de quatro *bytes* como a seguinte seqüência de valores em hexadecimal: “00”,“00”,“01”,“B3”. Já o cabeçalho final desse formato de vídeo é constituído, também, por quatro *bytes* como a seqüência a seguir: “00”,“00”,“01”,“B7”. Quaisquer valores diferentes destas seqüências inicial e final nos *bytes* do arquivo será considerado pelo protótipo como arquivo inválido e o programa será encerrado.

Verificada a validade do arquivo de vídeo, o protótipo iniciará o mapeamento das figuras 30, 60, 90, 120 assim por diante, pois a técnica de esteganografia utilizou um intervalo de uma figura deformada a cada 30 figuras encontradas. Este é o intervalo padrão adotado no protótipo. Essa lógica é a mesma utilizada no processo de esteganografia.

Após o mapeamento o programa analisará a figura número 30, que contém as informações dos cabeçalhos descritos na tabela 5.1. Caso não seja encontrado o cabestego (verificador da presença de esteganografia), o programa será encerrado, pois o vídeo não apresenta informação ocultada. Se o cabestego apresentar o valor “s” ou “S” o vídeo apresenta informação esteganografada.

Em seguida será analisado o cabcripto. Se esse cabeçalho apresentar o valor “s” ou “S” a mensagem ocultada está cifrada e será solicitada ao usuário a senha para a decifragem. Qualquer outro valor presente no cabcripto significará que a informação ocultada é um texto claro.

A análise em seqüência é o cabsegurança. Esse cabeçalho tem como finalidade evitar a quebra da chave criptográfica por força bruta. O cabeçalho é definido com valor zero no momento em que é aplicada a técnica de esteganografia. Já no processo de desesteganografia esse valor é incrementado a cada tentativa de extração da mensagem. Esse controle permite que ocorram 3 tentativas de extração, e ao ultrapassar esse número-limite, o arquivo de vídeo é apagado do sistema. Portanto, fica evidente que esse cabeçalho objetiva a confidencialidade da informação.

Dando seqüência é analisado o cabext, que verifica a existência de um arquivo ou de uma mensagem de texto ocultada. Se o cabeçalho apresentar os valores “1”,“1”,“1” a informação ocultada é um mensagem de texto. Caso o

cabeçalho apresente outros valores eles serão referentes à extensão do arquivo ocultado. Por exemplo, o formato “doc” para arquivo de texto do Microsoft Word.

O último cabeçalho verificado é o smensagem, que apresenta o tamanho total da informação ocultada. Esse cabeçalho é fundamental, pois é a partir dele que a mensagem poderá ser recuperada com êxito, pois ele indicará em quantas figuras está armazenada a informação e qual é o *byte* final da informação ocultada, para que a extração seja feita com exatidão.

Enfim, após a verificação de todos os cabeçalhos, o protótipo extrairá a mensagem e a retornará para o usuário. Caso seja uma mensagem de texto, ela será apresentada na tela com a opção de salvá-la em um arquivo no formato “txt”.

5.4 – Execução do Programa

Como resultado desta pesquisa, foi desenvolvido um programa denominado s*tealth-info.exe*. Esse programa foi compilado no Dev-C++ versão 4.9.9.2 e tem como finalidade a inserção ou a extração de um arquivo de qualquer formato ou de um texto por meio da esteganografia.

O funcionamento do programa é relatado a seguir:

O primeiro passo para realizar a esteganografia em um vídeo do padrão MPEG-1 é a inicialização do *stealth-info.exe,* localizando o diretório onde ele está armazenado.

Em seguida, o procedimento de inicialização é executado a partir de:

- Duplo *click* com o botão esquerdo do mouse no ícone do programa no ambiente *Windows*;
- Digitação do comando “*stealth-info.exe*” no *Prompt* de comando;

Logo após esse procedimento irá aparecer o menu inicial do programa, como pode ser visto na Figura 5.4.

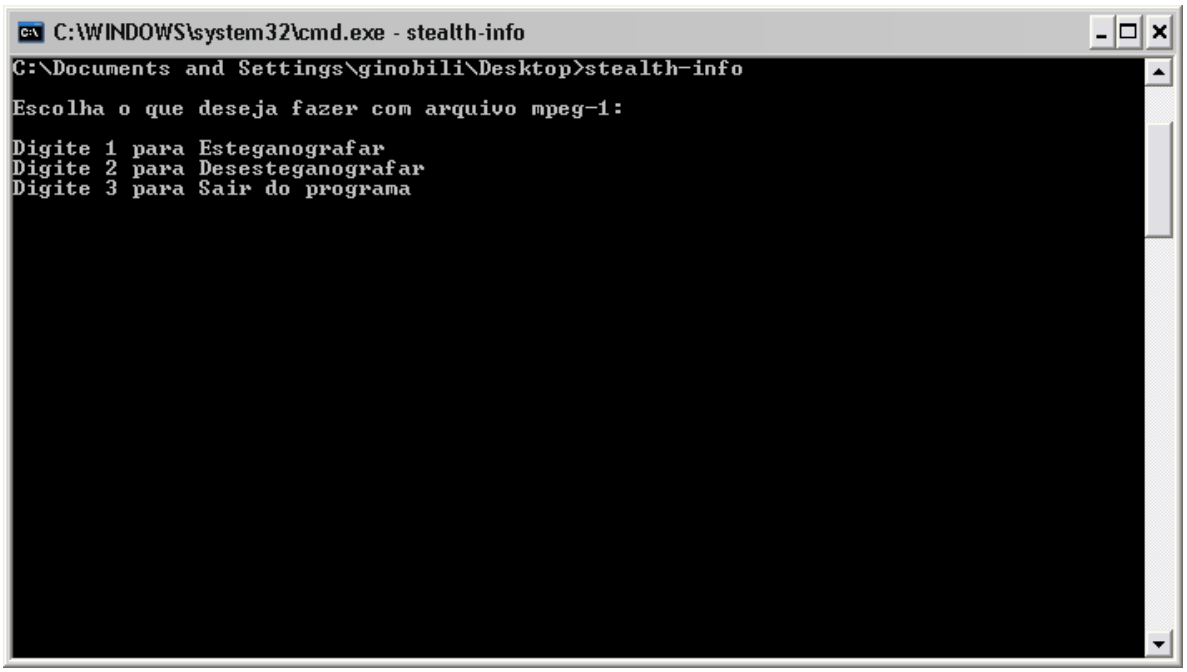

Figura 5.4 – Tela inicial do programa *stealth-info*

Fonte: Autor

Nesse menu são oferecidas três opções:

- 1- Esteganografar - processo de inserção da informação dentro de um vídeo MPEG-1;
- 2- Desesteganografar - processo de extração da informação presente em um vídeo MPEG-1;
- 3- Sair do Programa - processo de encerramento do programa;

5.4.1 – Esteganografia

Como segundo passo, para fins didáticos, será escolhida a primeira opção (Esteganografar) digitando número “1”. Após esse procedimento irá aparecer uma tela igual à da Figura 5.5.

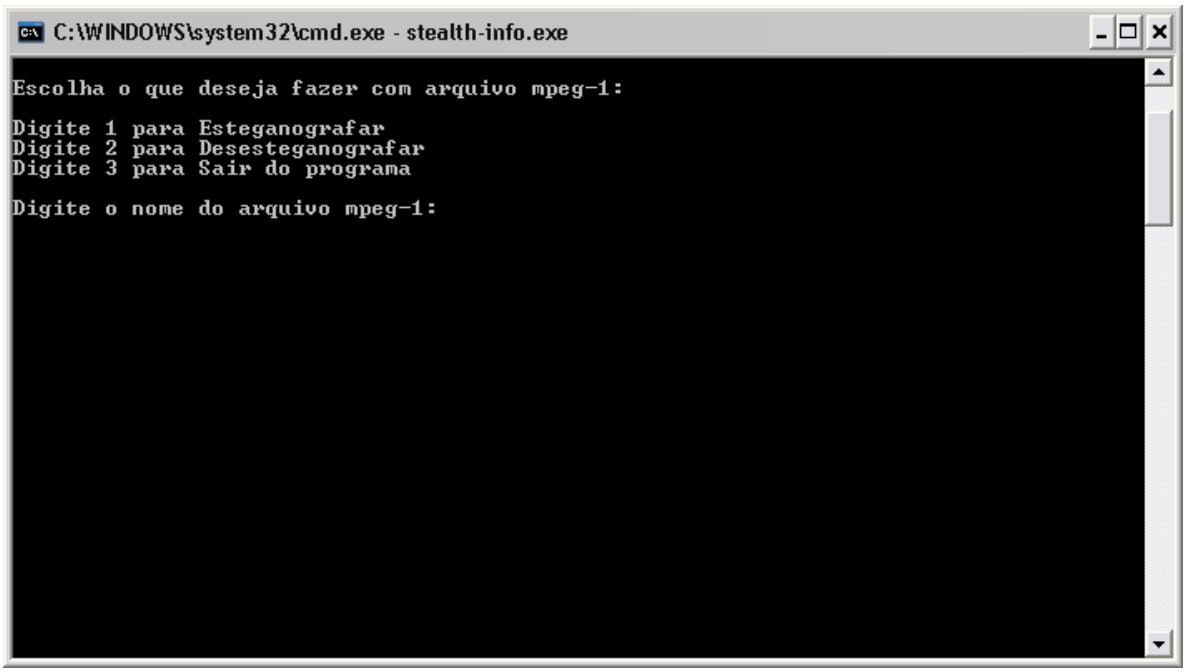

Figura 5.5 – Tela para escolha do vídeo MPEG-1 que será esteganografado
Fonte: Autor

Nessa segunda tela será solicitado que o usuário digite o nome do arquivo de vídeo que será usado como portador no processo de esteganografia.

O terceiro passo consiste na escolha do arquivo pelo usuário. Nesse momento, o programa irá analisar a estrutura hexadecimal dos cabeçalhos de inicialização (quatro primeiros *bytes*) e de finalização (quatro *bytes* finais) do arquivo, para determinar se a entrada foi válida, ou seja, se o arquivo pertence ao padrão MPEG-1. Para exemplificar, será considerado neste caso o arquivo "*hockey1.mpeg*", um arquivo de vídeo que atende as condições analisadas, ou seja, que apresenta os cabeçalhos iniciais e finais válidos para o formato MPEG-1. Dessa forma, irá aparecer na seqüência, uma tela semelhante à da Figura 5.6.

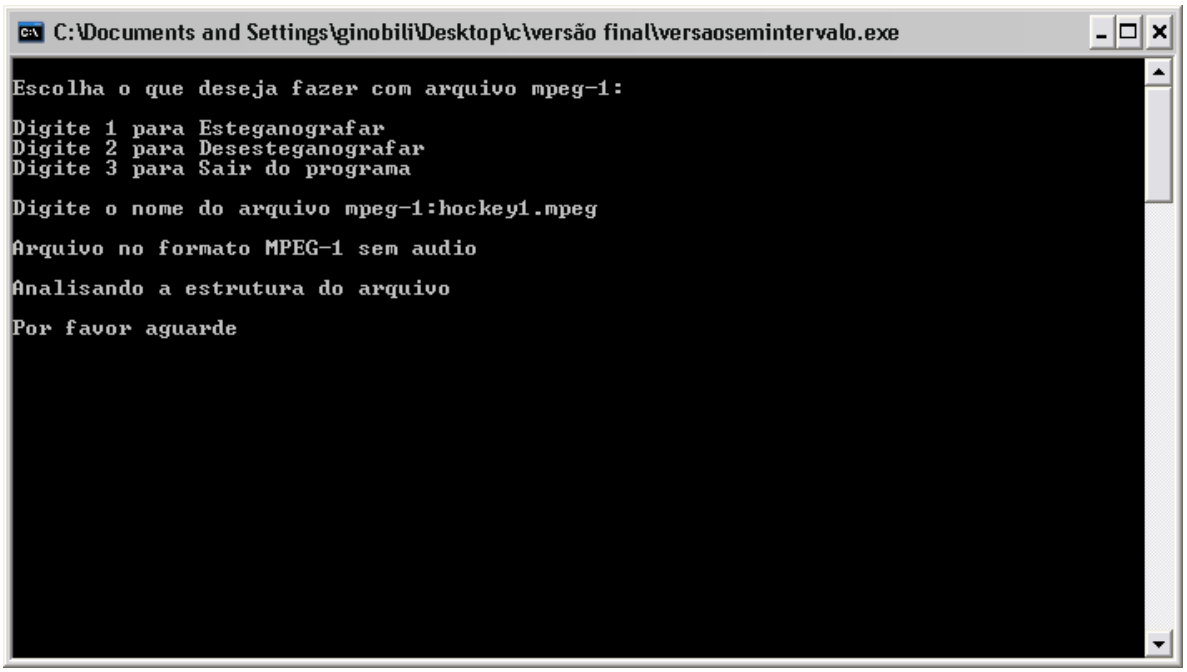

Figura 5.6 – Tela de espera da análise da estrutura do vídeo
Fonte: Autor

Na terceira tela o usuário irá aguardar enquanto o programa executa um algoritmo de mapeamento das figuras presentes no vídeo MPEG-1. Para identificar cada figura, o algoritmo procura por um cabeçalho de identificação da figura, uma seqüência de quatro *bytes* com valores em hexadecimal iguais a “00”,”00”,”00”,”01”. A cada figura encontrada o algoritmo armazena, em uma estrutura de dados, informações referentes ao número da figura, a posição do primeiro *byte* válido para esteganografia após o cabeçalho de identificação da figura, e o tamanho em *bytes* ocupado pela figura no arquivo de vídeo. Aparecerá uma tela semelhante à da Figura 5.7.

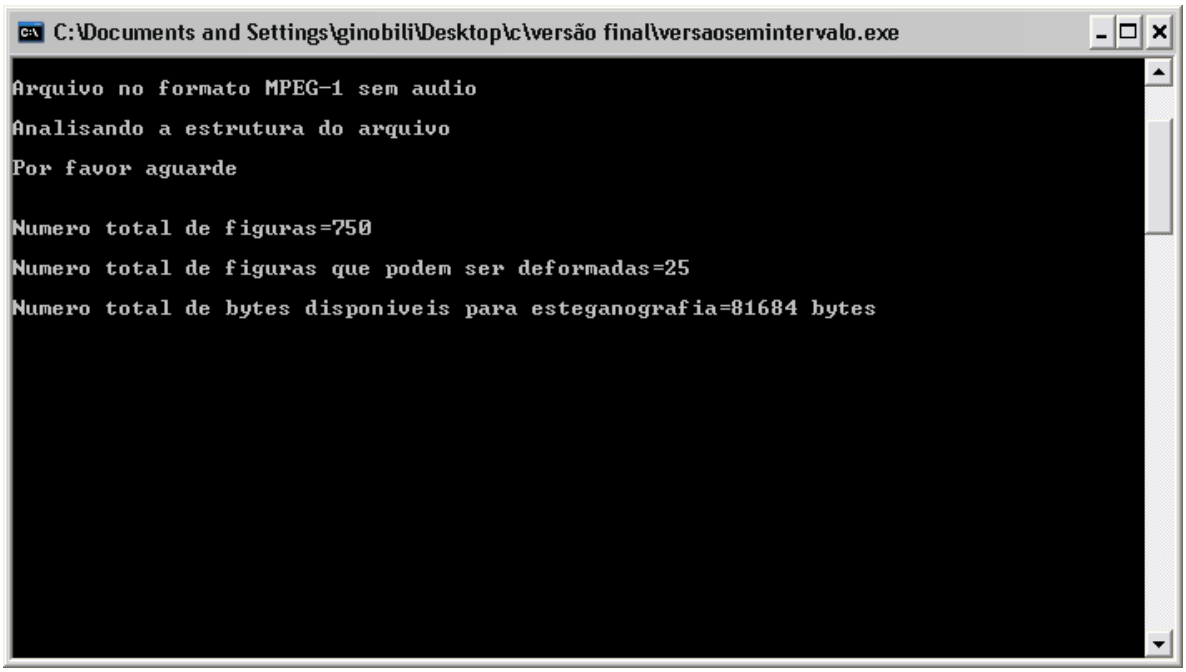

Figura 5.7 – Tela de resultados da análise da estrutura do vídeo
Fonte: Autor

Nessa quarta tela são informados ao usuário os resultados da análise da estrutura do vídeo "*hockey1.mpeg*". Dentre as informações que são relatadas temos:

- Número Total de Figuras – é o número de figuras encontradas no vídeo;
- Número Total de Figuras que podem ser Deformadas – é o resultado da divisão do Número Total de Figuras pelo Intervalo entre as Figuras em que a esteganografia será feita;
- Número total de *bytes* disponíveis para esteganografia – é a soma total do tamanho em *bytes* de cada figura que será esteganografada. Essas figuras são definidas pelo Intervalo entre as Figuras.

Neste protótipo foi determinado em 30 figuras o Intervalo entre as Figuras em que a esteganografia será feita. Isto quer dizer que, a cada 30 figuras encontradas, uma vai receber a inserção de fragmentos dos dados da informação que se pretende ocultar.

Esse intervalo é necessário porque a inserção dos dados da informação sigilosa deforma completamente as figuras escolhidas para tal feito. Assim, foi estabelecido esse intervalo de 30 figuras, pois, em geral, os vídeos do padrão MPEG-1 apresentam uma taxa de amostragem entre 24 e 30 quadros (figuras) por segundo. Então, ao utilizar esse intervalo de inserção, o vídeo é deformado de maneira que fique pouco perceptível ao olho humano.

Ao analisar a Figura 5.8, tem-se que o vídeo "*hockey1.mpeg*" apresenta um total de 750 figuras. Foi estabelecido que a cada 30 figuras uma delas poderá ser esteganografada, o que resultaria em 25 figuras deformáveis. O resultado do Número de *bytes* disponíveis para esteganografia é obtido pelo somatório do tamanho em *bytes* das figuras de números 30, 60, 90, 120, 150, 180, 210, 240, 270,..., até 750. Essas figuras serão exatamente as que irão receber os dados da informação a ser ocultada.

Logo após o usuário analisar as informações apresentadas anteriormente, o programa irá executar o quarto passo e apresentará uma tela semelhante à da Figura 5.8.

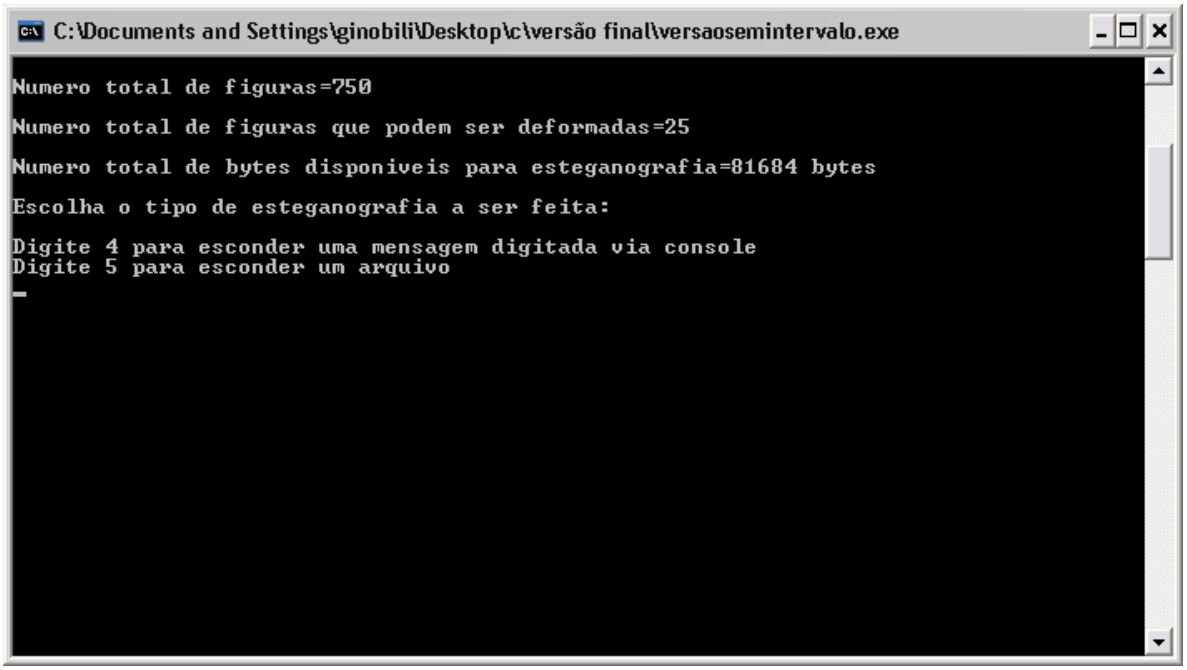

Figura 5.8 – Tela para escolha do tipo de informação que será esteganografada
Fonte: Autor

Na quinta tela serão apresentadas ao usuário as seguintes opções:

- tecle 4 - para esconder uma mensagem digitada via console, o que consiste na inserção de uma mensagem de texto digitada no próprio teclado;
- tecle 5 - para esconder um arquivo, o que consiste na inserção de outro arquivo, em qualquer formato, inclusive no próprio formato de vídeo MPEG-1;

Assim, caso o usuário escolha a opção número “4“, aparecerá uma tela semelhante à da Figura 5.9.

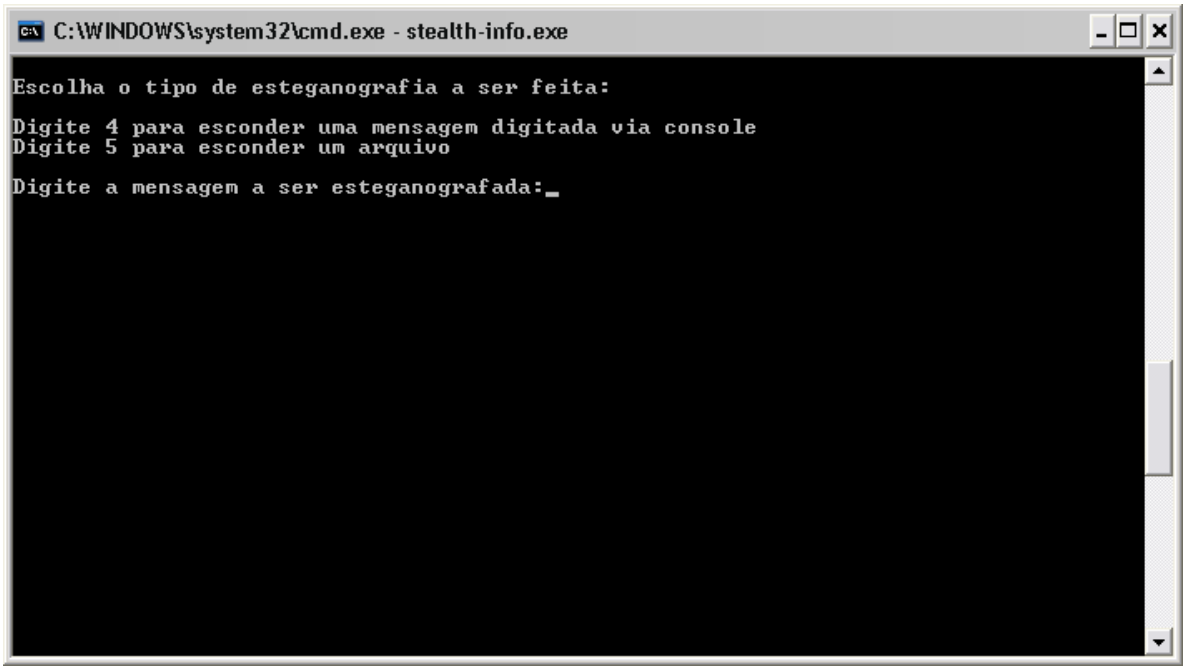

Figura 5.9 – Tela para digitação da mensagem que o usuário deseja ocultar
Fonte: Autor

Nessa tela, o usuário vai escolher e digitar a mensagem que deseja esteganografar. Logo após, o algoritmo vai comparar os valores do tamanho da mensagem digitada pelo usuário em relação ao Tamanho Total de *bytes* disponíveis para esteganografia. Se o valor ultrapassar o total dos *bytes* disponíveis, o programa será encerrado por erro, pois a entrada excede o tamanho máximo aceito.

Caso o usuário escolha a opção número “5”, aparecerá uma tela semelhante à da Figura 5.10.

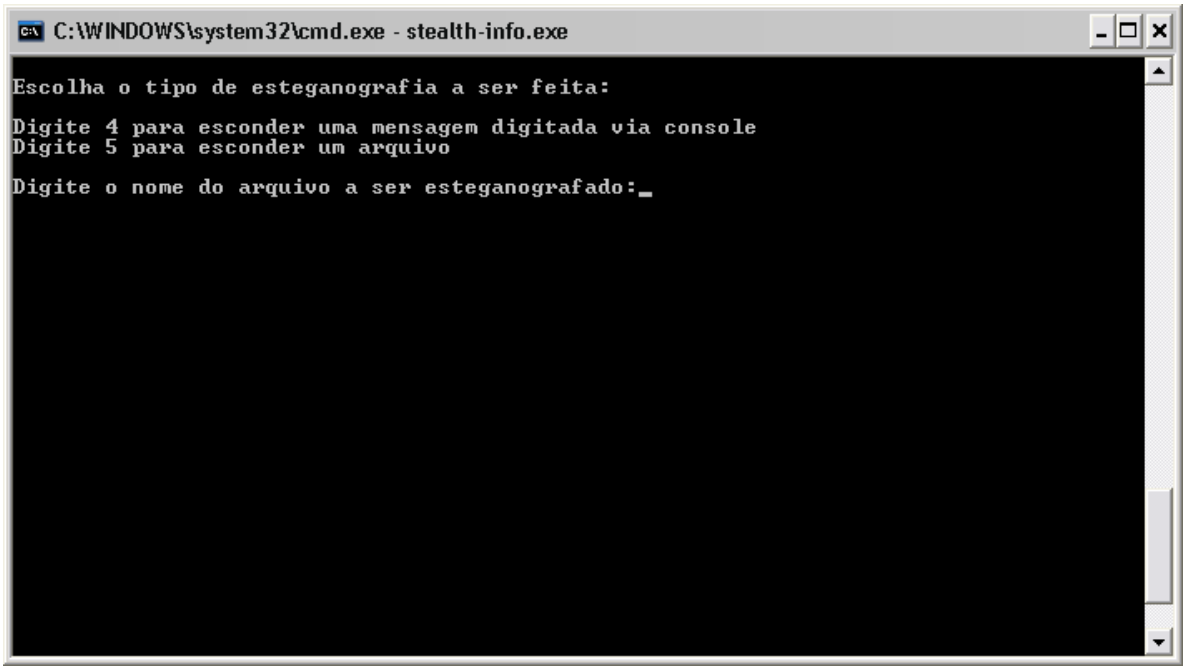

Figura 5.10 – Tela para a escolha do arquivo que o usuário deseja ocultar
Fonte: Autor

Nessa tela, o usuário vai escolher o arquivo que deseja esteganografar. Em seguida, o programa vai verificar se o arquivo escolhido existe. Caso seja inexistente, o programa vai encerrar por erro. Logo após essa verificação, o algoritmo vai estabelecer uma comparação entre os valores do tamanho do arquivo escolhido pelo usuário e do Tamanho Total de *bytes* disponíveis para esteganografia. Se o valor ultrapassar o total dos *bytes* disponíveis, o programa será encerrado por erro, pois o arquivo excede o tamanho máximo aceito.

No passo seguinte independente da escolha do tipo de esteganografia (arquivo ou mensagem) feita pelo usuário, aparecerá uma tela semelhante à da Figura 5.11.

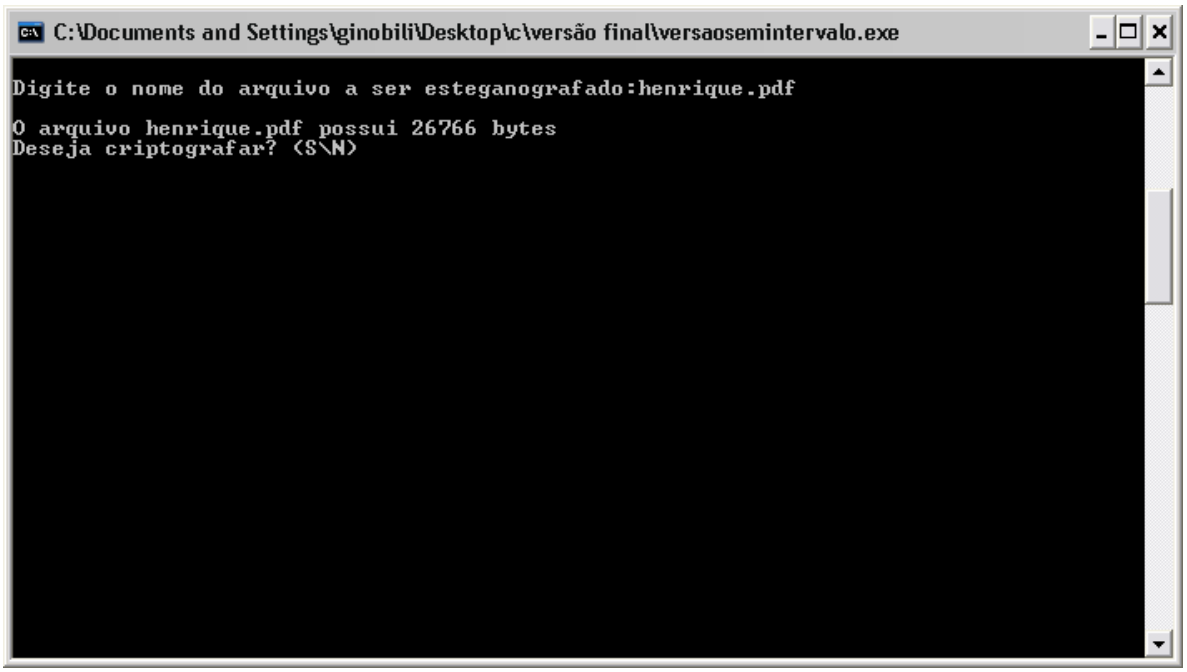

Figura 5.11 – Tela para a escolha da utilização da Criptografia DES
Fonte: Autor

Nessa tela, o usuário escolherá entre utilizar ou não a criptografia DES para aumentar a confidencialidade da informação. Caso opte por não cifrar a informação que irá ser esteganografada, a informação continuará segura. Entretanto, se o arquivo "*hockey1.mpeg*" for sujeito à interceptação e uma ferramenta de esteganálise lhe for aplicada, haverá possibilidade de o analista de segurança detectar e recuperar a informação ocultada, ou seja, terceiros poderão tomar conhecimento de seu conteúdo.

Outro problema associado ao desuso da criptografia DES é a possibilidade de coincidência entre a seqüência de 4 *bytes* referentes ao cabeçalho de identificação de figura do vídeo MPEG-1 (em hexadecimal: "00","00","00" e "01") e o arquivo escolhido para ser ocultado. Essa coincidência gera um problema no algoritmo usado para desesteganografia, pois, ao inserir seqüências de *bytes* iguais às do cabeçalho de identificação de imagens, o algoritmo irá entender que após cada seqüência de 4 *bytes* encontrada igual ao cabeçalho existe uma figura e, assim, o número de figuras do vídeo irá aumentar. Como conseqüência disso,

os dados esteganografados jamais poderão ser recuperados e o vídeo poderá apresentar deformações perceptíveis ao olho humano.

A partir de teste feitos pelo autor, foi constatado que os seguintes arquivos formatos escolhidos para serem ocultados sem criptografia apresentavam o problema de coincidência: DOC, JPEG, PDF e, logicamente, o próprio MPEG-1.

Caso o usuário opte pela utilização de criptografia, será solicitado que escolha uma chave criptográfica com tamanho máximo de 8 caracteres. Se o tamanho da chave exceder, o programa requisitará, novamente, a escolha de uma chave válida. A tela de solicitação da chave de criptografia pode ser visualizada na Figura 5.12.

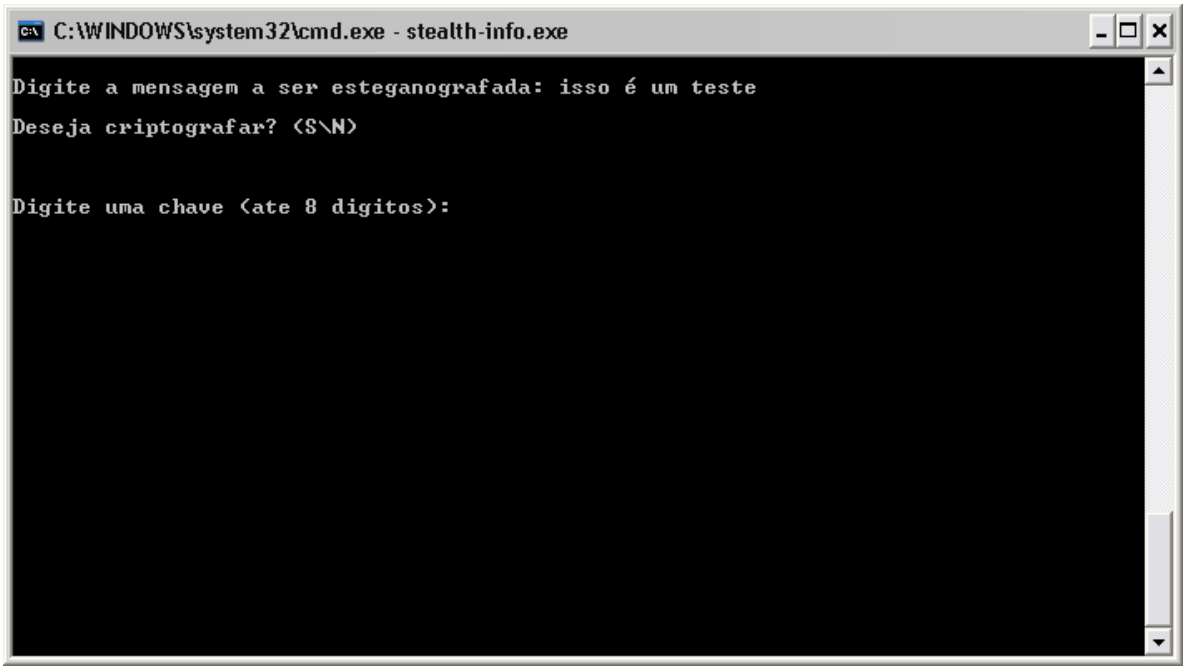

Figura 5.12 – Tela para a escolha da chave criptográfica
Fonte: Autor

O algoritmo de criptografia do programa realiza o processo semelhante à criptografia DES, explicada no capítulo 4 deste projeto.

No passo seguinte, independente da escolha ou não do uso da criptografia pelo usuário, o programa executará o algoritmo da técnica de esteganografia por substituição ou sobrescrita de dados no arquivo "*hockey1.mpeg*".

Esse algoritmo preenche a primeira figura a ser esteganografada com cabeçalhos fundamentais para o processo de desesteganografia.

No exemplo do vídeo "*hockey1.mpeg*", que está sendo usado para melhor entendimento do protótipo, foram identificadas 25 figuras "esteganografáveis" no intervalo de uma figura a cada 30. Isso leva a que as figuras 30, 60, 90, 120, 150, 180, 210, 240, 270,..., até 750 sejam as figuras elegíveis para ocultamento da informação sigilosa. As figuras "esteganografáveis" serão preenchidas de acordo com o tamanho da informação escolhida, caso ela seja menor que o tamanho disponível. Provavelmente as últimas figuras não serão deformadas. Com isso, fica evidente que, quanto maior a mensagem, maior a deformação do vídeo.

Dentre os cabeçalhos fundamentais que serão inseridos na primeira figura deformável, no caso em estudo a figura de número 30, tem-se:

- cabestego – cabeçalho verificador da presença de esteganografia no arquivo MPEG-1;
- cabcripto – cabeçalho verificador da presença ou não de criptografia DES;
- cabsegurança – cabeçalho que verifica o número de tentativas de extração, feitas em um determinado vídeo esteganografado. No processo de esteganografia esse valor é determinado como "zero". Se o valor desse cabeçalho apontar 3 ou mais tentativas de extrair a informação sigilosa o arquivo de vídeo será destruído por motivo de segurança;
- cabext – cabeçalho que armazena a extensão de um arquivo escolhido para ser ocultado. Por exemplo "*pdf*";
- smensagem – cabeçalho que armazena o tamanho em *bytes* da informação esteganografada no vídeo. Esse cabeçalho é importante porque possibilita a recuperação efetiva da mensagem;

Logo após a inserção desses cabeçalhos fundamentais, a informação escolhida para ser ocultada será inserida no vídeo e particionada, se necessário, de acordo com o preenchimento dos espaços disponíveis das figuras "esteganografáveis".

Por fim, a mensagem será inserida com sucesso no arquivo de vídeo "*hockey1.mpeg*" e o programa irá mostrar uma tela informando que a esteganografia foi feita com sucesso, semelhante à da Figura 5.13.

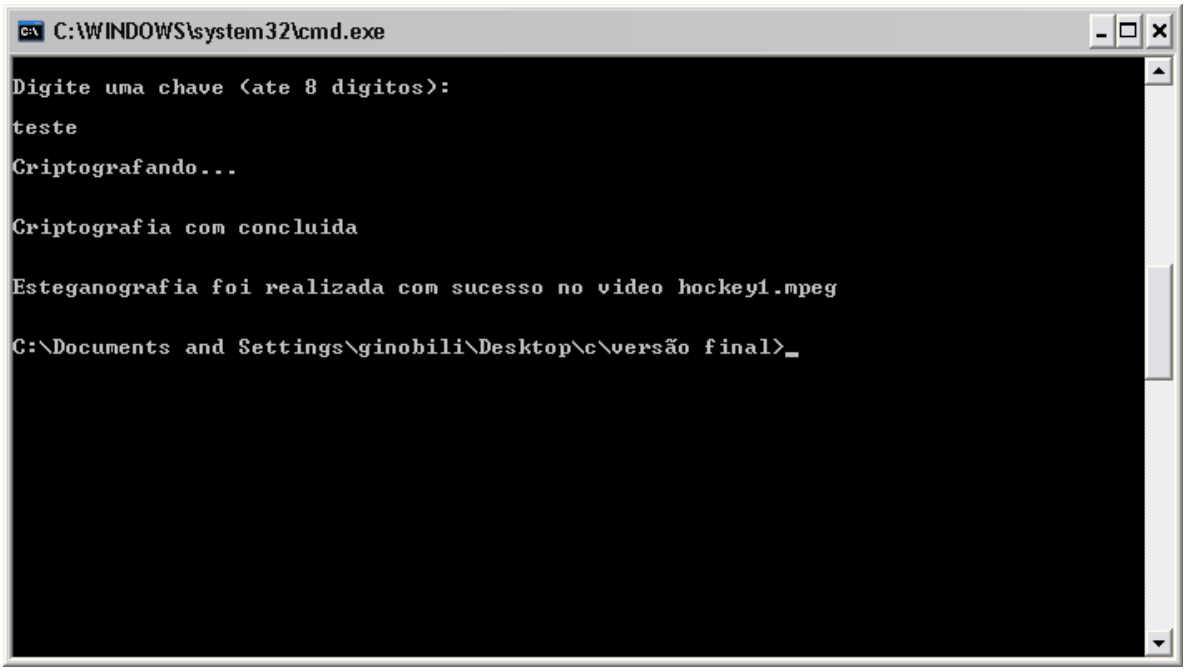

Figura 5.13 – Tela de encerramento do programa para processo de esteganografia.
Fonte: Autor

5.4.2 – Desesteganografia

Apresentadas as etapas de esteganografia, será demonstrado, em seguida, seu processo inverso, ou seja, as etapas de extração da mensagem mascarada em um vídeo MPEG-1.

O primeiro passo desse procedimento é a inicialização do programa "*stealth-info.exe*", seguindo-se a escolha da opção número dois (Desesteganografar), presente no menu inicial do programa, o que pode ser visto na figura 5.2, anteriormente apresentada.

Em função dessa escolha, o programa irá mostrar uma tela semelhante à da figura 5.3, também apresentada anteriormente no processo de esteganografia.

Nessa tela, o usuário irá digitar o nome do arquivo que deseja verificar se possui esteganografia e, em caso positivo, vai iniciar o processo de extração da mensagem. Nesse procedimento de desesteganografia será mantido o exemplo do vídeo do padrão MPEG-1 "*hockey1.mpeg*", que foi utilizado no item 5.4.1.

Logo após a escolha do arquivo, o programa irá analisar a estrutura hexadecimal dos cabeçalhos de inicialização (quatro primeiros *bytes*) e de finalização (quatro *bytes* finais) do arquivo para determinar se a entrada foi válida, ou seja, se o arquivo pertence ao padrão MPEG-1.

Em seguida, o usuário irá aguardar que o programa execute um algoritmo de mapeamento das figuras presentes no vídeo MPEG-1. Esse procedimento é semelhante ao mapeamento feito no procedimento de esteganografia. O algoritmo identificará e armazenará o resultado em uma estrutura de dados com as informações das figuras "esteganografáveis" referentes ao número da figura, à posição do primeiro *byte* válido para esteganografia após o cabeçalho de identificação da figura, e ao tamanho em *bytes* ocupado pela figura no arquivo de vídeo. Conforme foi esclarecido no item 5.4.1, as figuras "esteganografáveis" aparecem a cada intervalo de 30 figuras.

Ao terminar a construção dessa estrutura de dados, o algoritmo, no exemplo do vídeo "*hockey1.mpeg*", novamente terá encontrado 25 figuras deformáveis - as figuras de números: 30, 60, 90, 120, 150, 180, 210, 240, 270,..., até 750. Logo após essa identificação, o programa irá analisar a primeira figura com possibilidade de apresentar esteganografia. Essa figura, no exemplo que está sendo utilizado, é a figura número 30. O resultado dessa análise pode ser visto na Figura 5.14.

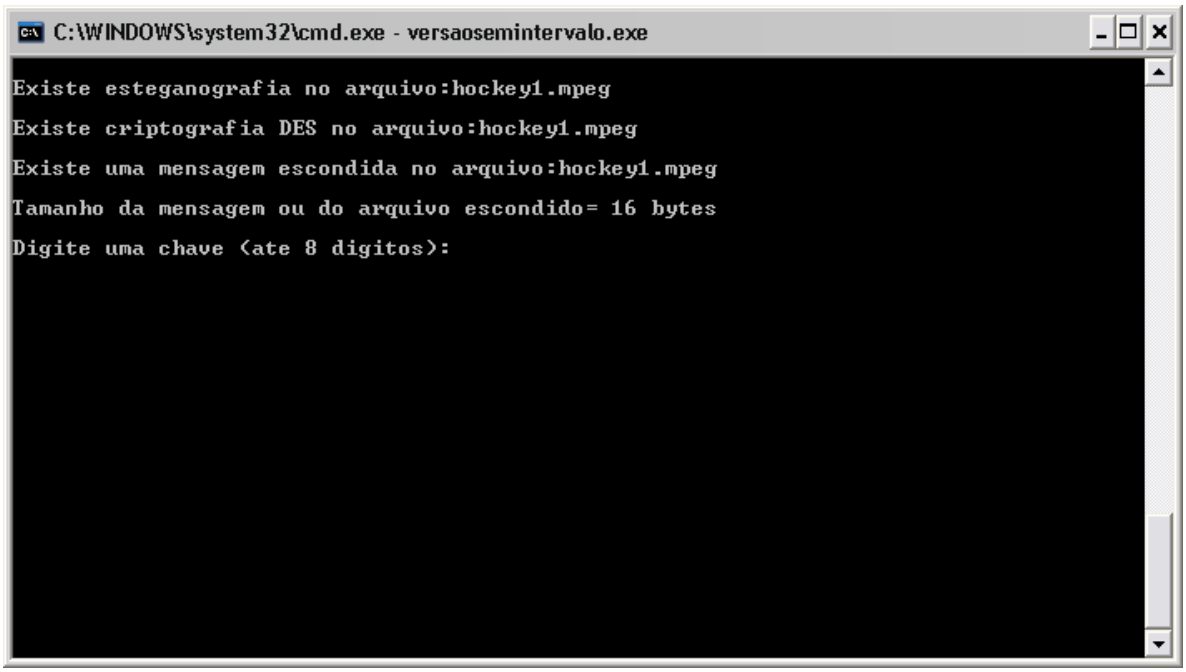

Figura 5.14 – Tela que apresenta a análise da figura de número trinta do vídeo "*hockey1.mpeg*"
Fonte: Autor

Nessa tela, são mostrados os resultados da análise dos cabeçalhos fundamentais presentes na figura de número 30. O primeiro cabeçalho analisado é o cabestego. Este cabeçalho verifica a presença de esteganografia no arquivo de vídeo. Caso o valor dele seja "S", existe alguma informação ocultada. Caso seja qualquer valor diferente de "S", o programa será encerrado, informando que o arquivo de vídeo selecionado não apresenta esteganografia.

O segundo cabeçalho analisado é o cabcripto, que verifica o uso ou desuso de criptografia DES. Se esse cabeçalho apresentar o valor "s" ou "S", existe uma chave criptográfica na informação ocultada dentro do arquivo de vídeo. Caso seja qualquer valor diferente de "s" ou "S", a informação não está cifrada.

O terceiro cabeçalho analisado é o cabsegurança, que verifica o número de tentativas de extração feitas no vídeo que está sendo analisado ("*hockey1.mpeg*"). Na primeira tentativa de extração, o valor do cabeçalho é "0". Nas demais, esse cabeçalho soma mais "1" ao valor inicial, ou seja, a cada tentativa é adicionado "1" ao valor total do cabeçalho, até ultrapassar a terceira tentativa, quando o vídeo

será deletado do sistema. Essa medida visa aumentar a segurança da informação ocultada no arquivo MPEG-1 em caso de interceptação por terceiros do arquivo de vídeo. Assim, mesmo tendo acesso ao programa *stealth-info.exe,* seria evitada a sucessiva aplicação de tentativas de extração da informação ocultada, bem como de variações de uso das chaves criptográficas, para quebrar sua segurança. Com isso, para garantir o sigilo da informação, o arquivo é deletado, eliminando-se a possibilidade de insistência em descobrir a chave criptográfica.

O quarto cabeçalho analisado é o cabext, que verifica a extensão (formato) do arquivo. Caso não exista uma extensão do arquivo e sim uma seqüência de três valores “1”, ”1” e ”1” da tabela ASCII, isso será interpretado pelo programa como a existência de uma mensagem digitada via console.

O quinto e último cabeçalho analisado é o smensagem, que identifica o tamanho total da informação ocultada no arquivo de vídeo. Vale lembrar que o tamanho disponível para esteganografia do arquivo nem sempre é utilizado por completo. Assim, esse cabeçalho é importante, pois possibilita a recuperação efetiva da mensagem.

Dessa forma, ao analisar as informações do “*hockey1.mpeg*”, o programa concluiu que existe esteganografia no arquivo, que foi manipulado no item 5.4.1. Além disso, ele também identifica que foi utilizado o recurso da criptografia e solicita ao usuário que digite a chave criptográfica. Também é identificada a existência de uma mensagem ocultada e não de um arquivo.

Ao terminar a análise, o programa informa o tamanho da mensagem que será extraída do vídeo, a qual será armazenada em uma variável, aguardando a entrada da chave criptográfica.

Se no arquivo de vídeo não houvesse sido aplicada a criptografia, o programa automaticamente iniciaria a extração da mensagem, buscando nas figuras “esteganografáveis” os fragmentos da mensagem para sua reconstrução e, finalmente, para visualização do usuário.

Caso existisse um arquivo e não uma mensagem, o processo de recuperação seria semelhante ao da mensagem. Porém, após a extração do arquivo ocultado, seria iniciado seu processo de reconstrução, no formato

“estego.extensão”, sendo que essa extensão seria aquela informada pelo cabext, anteriormente comentado.

Por exemplo, se existisse um arquivo ocultado e esse fosse um arquivo de texto do *Microsoft Word,* o cabext apresentaria a extensão “DOC”. Então, ao reconstruir o arquivo, ele seria denominado “estego.doc”, com todas as propriedades do arquivo original, existentes antes do processo de esteganografia.

Considerando-se, no exemplo, a existência de criptografia, o usuário deve digitar a chave criptográfica. Em caso de sucesso na inserção da chave, a mensagem será decifrada e o programa a apresentará totalmente recuperada na tela, como pode ser visto na Figura 5.15.

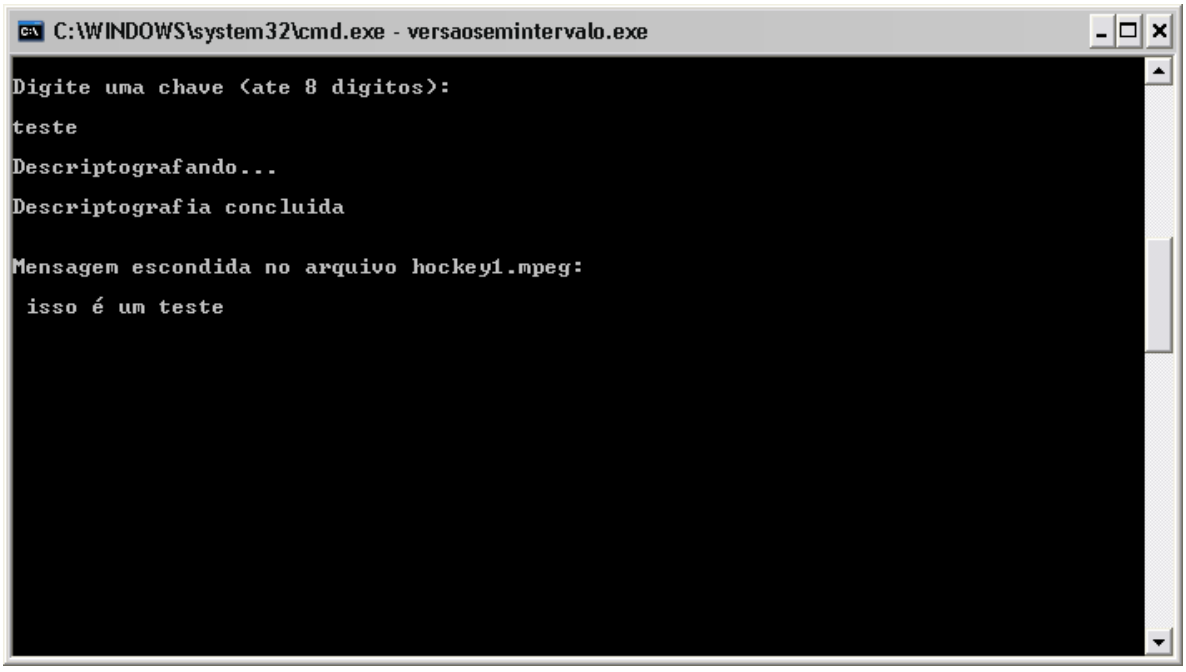

Figura 5.15 – Tela que apresenta a mensagem desesteganografada
Fonte: Autor

Caso a chave inserida não seja válida, o programa, também, apresentará na tela uma mensagem, que, entretanto, permanecerá cifrada, não tendo valor real como informação para o usuário.

Em se tratando de um arquivo ocultado, a chave inserida inválida também permitiria sua reconstrução, porém o arquivo seria considerado inválido ao ser

executado, pois ainda apresentaria cifragem em sua estrutura binária. Com a chave correta, o arquivo ocultado, seria reconstruído com sucesso, mantendo as propriedades do arquivo original e sendo válido em sua execução.

Ao terminar os processos de decifragem e de apresentação da mensagem extraída para o usuário, o programa oferece a possibilidade de salvar o texto em um arquivo do formato “TXT”. Em alguns casos, a mensagem sigilosa pode ser muito longa ou mesmo demandar consulta por mais de uma vez. Essa possibilidade pode ser vista na figura 5.16.

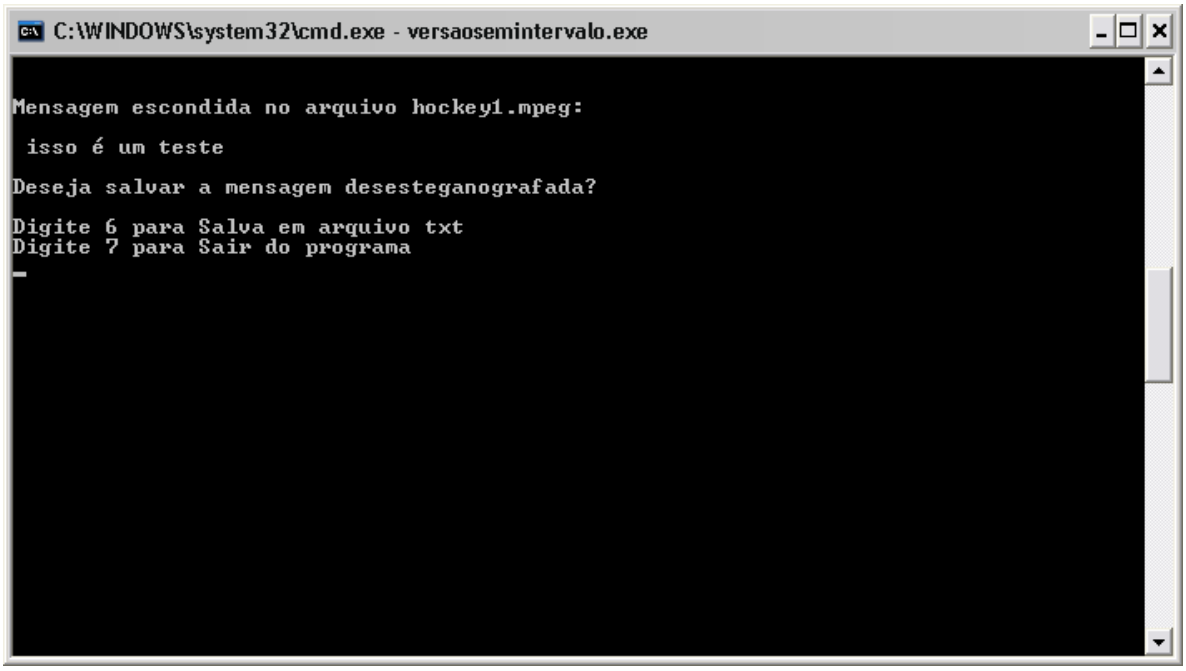

Figura 5.16 – Tela que apresenta opção de salvar a mensagem em arquivo txt.
Fonte: Autor

Por fim, o programa é encerrado, com a indicação de que a desesteganografia foi feita com sucesso, acrescentando o nome do arquivo criado com o conteúdo da informação extraída. Ao extrair uma mensagem, ela é salva em um arquivo de texto denominado “mensagem estego.txt”. Quando um arquivo é extraído do vídeo, ele é salvo com o nome “estego.extensão do arquivo”. Essa “extensão do arquivo” é informada pelo cabext. Ambas as telas de encerramento podem ser vistas nas Figuras 5.17 e 5.18 respectivamente.

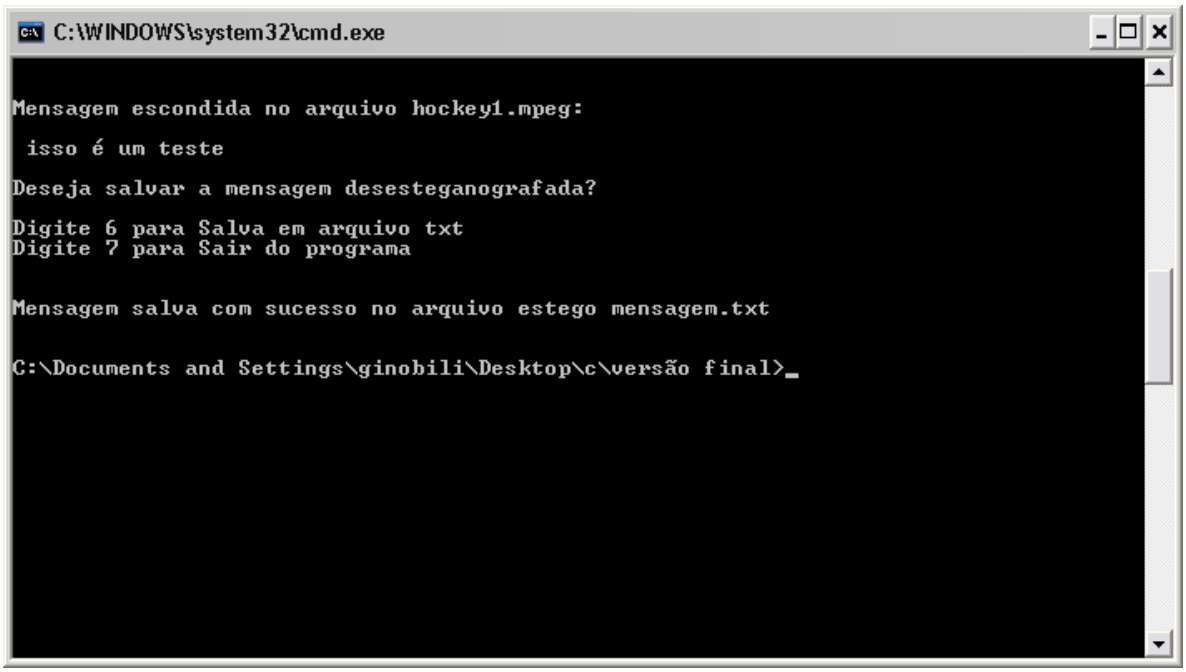

Figura 5.17 – Tela de encerramento de uma mensagem salva com sucesso.

Fonte: Autor

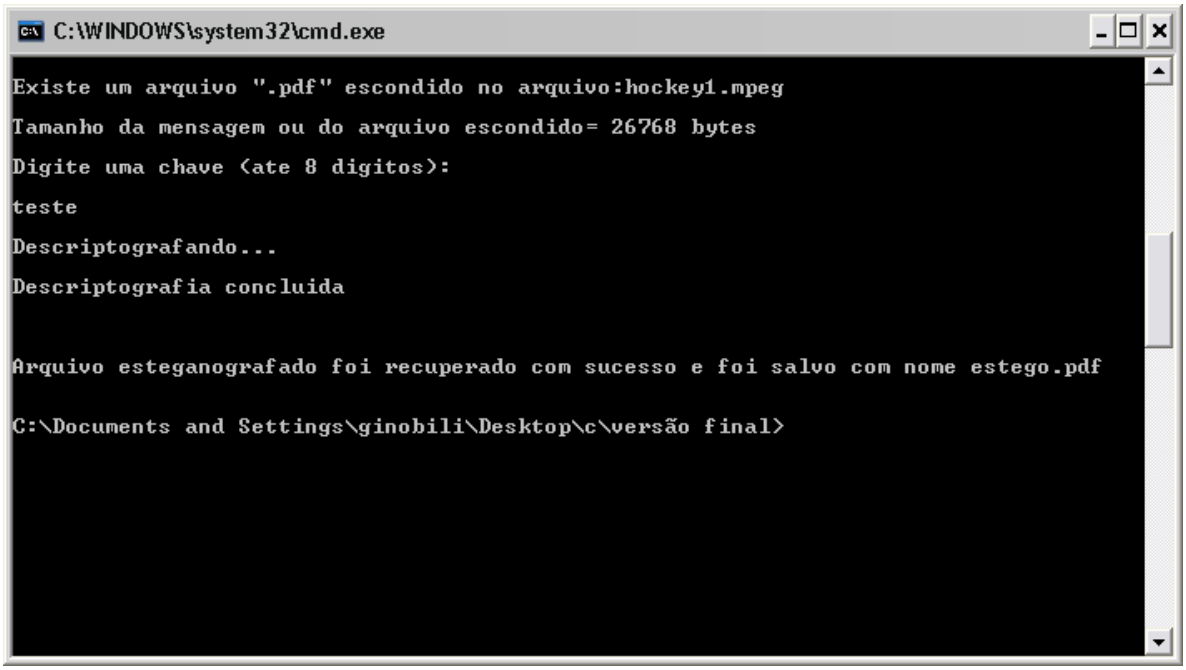

Figura 5.18 – Tela de encerramento de um arquivo recuperado com sucesso

Fonte: Autor

5.5 – Dificuldades e Limitações

5.5.1 – LSB

A principal dificuldade encontrada na elaboração deste protótipo foi a utilização da técnica de esteganografia LSB (Least Significant Bit). A partir dessa técnica, o *bit* considerado menos significativo de um número binário pode ser modificado sem que ocorram alterações visíveis na qualidade de uma imagem.

Essa técnica é muito difundida em programas de esteganografia de imagens. Cada imagem tem para cada *pixel*, quatro canais, que são representados por um *byte* cada.Os canais são: verde, vermelho, azul e alfa (mas só usaremos o verde, vermelho e o azul).

Na figura 5.19 mostra um exemplo de como é feito o processo LSB.

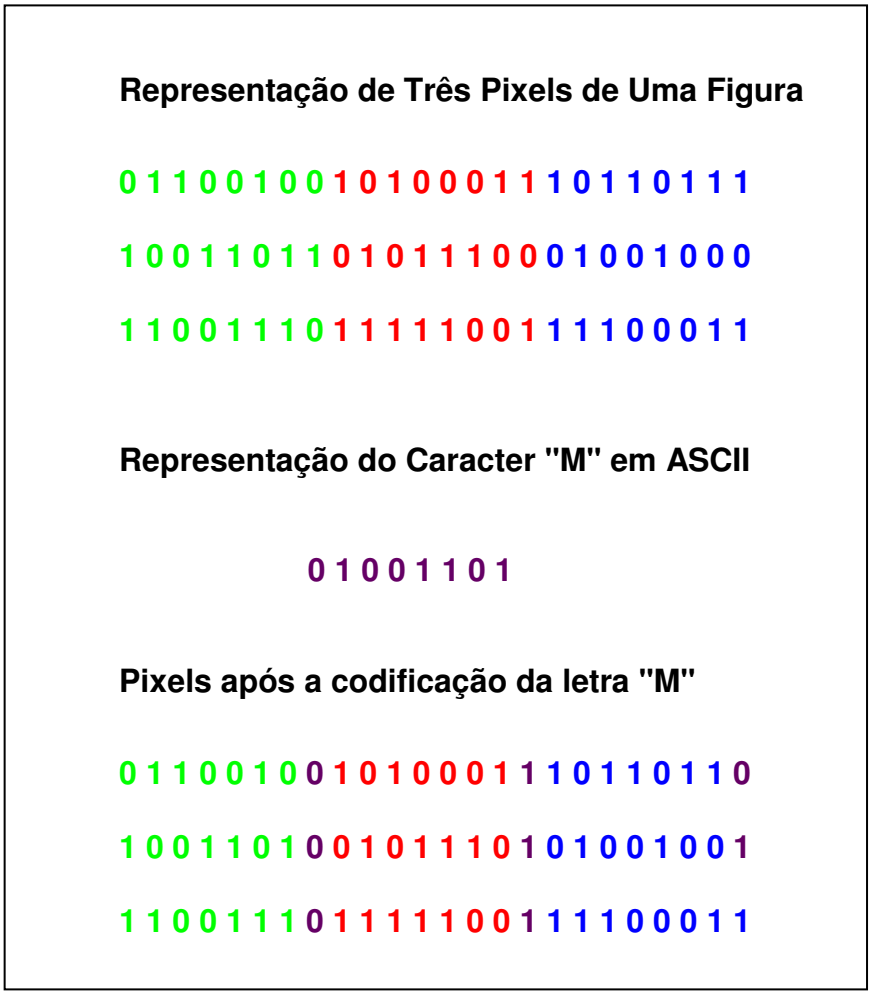

Figura 5.19 – Exemplo da técnica LSB

Fonte: Autor

Após o entendimento de como funciona o LSB, pode-se explica o porquê do não uso da técnica no protótipo. Essa técnica necessita acessar a camada de dado das imagens do vídeo para poder alterar os *bits* menos significativos das cores. Entretanto o formato de vídeo MPEG-1 apresenta os seguintes métodos de compressão: Sub-amostragem das informações de cromância; Quantização; Compensação de Movimento; DTC (Transformada Discreta do Cosseno); VLC (Variable Length Coding), Interpolação de Figuras. Esses métodos de compressão trabalham em associação e isso gera uma certa dificuldade no acesso pleno aos dados das cores. Além disso, a cromância (cor) tem seus valores sub-amostrados, reduzindo a possibilidade de aproveitamento deste tipo de dados.

Portanto, ao constatar essa realidade foi escolhida a técnica de substituição ou sobrescrita de dados, que utiliza algumas imagens do vídeo para esconder a mensagem ocultada, deformando-as, sem afetar a qualidade do vídeo.

5.5.2 – ISO- IEC – 11172-2

Uma das limitações que o protótipo apresentou durante seu desenvolvimento foi o uso de arquivo de vídeo no formato MPEG-1 sem áudio, pois essa ISO refere-se ao vídeo MPEG-1. A ISO - IEC 11172-1 é a que contém as informações necessárias para a manipulação do Sistema MPEG-1 (Áudio e Vídeo).

Dessa forma, o protótipo apenas disponibiliza a técnica de esteganografia para vídeo MPEG-1 sem áudio.

5.5.3 – Fenômeno da Coincidência

Outra limitação do protótipo foi identificada durante a fase de testes e correções.

Foi observado, em alguns arquivos escolhidos para serem ocultados, que o processo de desesteganografia não era feito corretamente, ou seja, a extração do arquivo ocultado feita no vídeo MPEG-1 não era obtida com sucesso.

Inicialmente, acreditava-se que isso era resultado de algum erro de lógica na programação do processo de mapeamento da figuras. Entretanto, após algumas análises da estrutura binária dos arquivos ocultados, foi constatado que esse arquivo apresentava em sua estrutura uma seqüência de 4 bytes em hexadecimal igual a: “00”,”00”,”00” e ”01. Coincidentemente essa seqüência de *bytes* é a mesma utilizada para identificar os cabeçalhos das figuras do formato de vídeo MPEG-1. Como conseqüência disso, ao ocultar esse tipo de arquivo, que apresenta o fenômeno da coincidência, o arquivo de vídeo esteganografado aumentava a quantidade dessa seqüência em sua estrutura. Então ao mapear as figuras eram encontrados mais cabeçalhos de figuras do que figuras disponíveis. Como resultado dessa prática, o vídeo apresentava deformações visíveis durante

a sua execução e não conseguia recuperar a mensagem no processo de desesteganografia.

A alternativa de contornar esse problema foi a utilização da criptografia *DES* para que essas seqüências coincidentes fossem cifradas e conseqüentemente modificadas, não deformando o vídeo ao serem ocultadas.

É recomendável que o arquivo seja cifrado antes de ser inserido no vídeo original. Dessa forma, o vídeo esteganografado mantém o número original de figuras, o cabeçalho de identificação de figuras não coincide com outras partes do arquivo e a mensagem é recuperada com sucesso.

Portanto, para uso de arquivos na esteganografia de vídeo MPEG-1 recomenda-se que a criptografia seja utilizada a fim de garantir a integridade da informação secreta e do vídeo.

Capítulo 6 – Testes e Simulações

Para efetuar os testes e simulações do protótipo foram utilizadas as seguintes ferramentas:

- *Hex Workshop* versão 4.23 (da empresa *BreakPoint Software)* – um editor de hexadecimais que foi utilizado para efetuar a comparação binária entre o arquivo original e o esteganografado;
- *histogramampeg1.exe* (desenvolvido pelo autor) – um programa que armazena em um arquivo de texto ".txt" informações referentes aos valores em decimal de cada byte pertencentes ao arquivo original e ao arquivo esteganografado;
- *Excel* versão XP (da empresa *Microsoft)* – um editor de planilha que foi utilizado para gerar os gráficos dos histogramas do arquivo original e do arquivo esteganografado a partir das informações colhidas em arquivo de texto ".txt" pelo programa "*histogramampeg1.exe"*;

6.1 – Comparação binária

O primeiro método utilizado para esteganálise foi a comparação binária entre os arquivos de vídeo no formato MPEG-1: "*hockey1.mpeg"* e "*hockey1ESTEGANOGRAFADO.mpg"*. O segundo arquivo foi renomeado para facilitar a distinção entre o arquivo original e o esteganografado. A comparação binária é feita por meio da opção *checksum*, que verifica os bytes do arquivo nos quais os valores comparados divergem. Essa análise pode ser vista nas figuras 6.1 e 6.2.

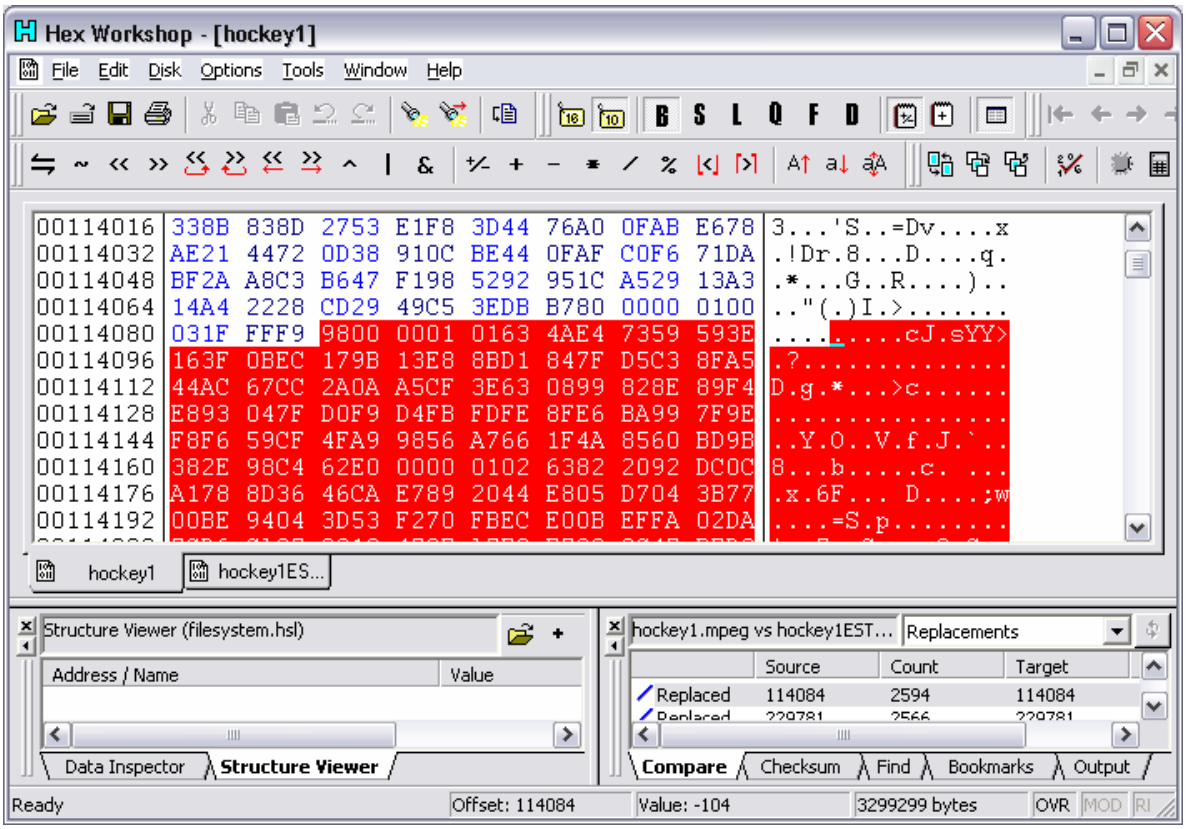

Figura 6.1 – "*hockey1.mpeg*" (original)

Fonte: Autor

A figura 6.1 mostra o resultado da comparação entre o arquivo original e o esteganografado. A região em vermelho representa os dados divergentes do arquivo do vídeo "*hockey1.mpeg*". Nesta região, os valores dos *bytes* apresentam diferenças, pois no arquivo esteganografado eles foram substituídos por fragmentos da mensagem ocultada.

Nesta figura são mostrados os *bytes* originais.

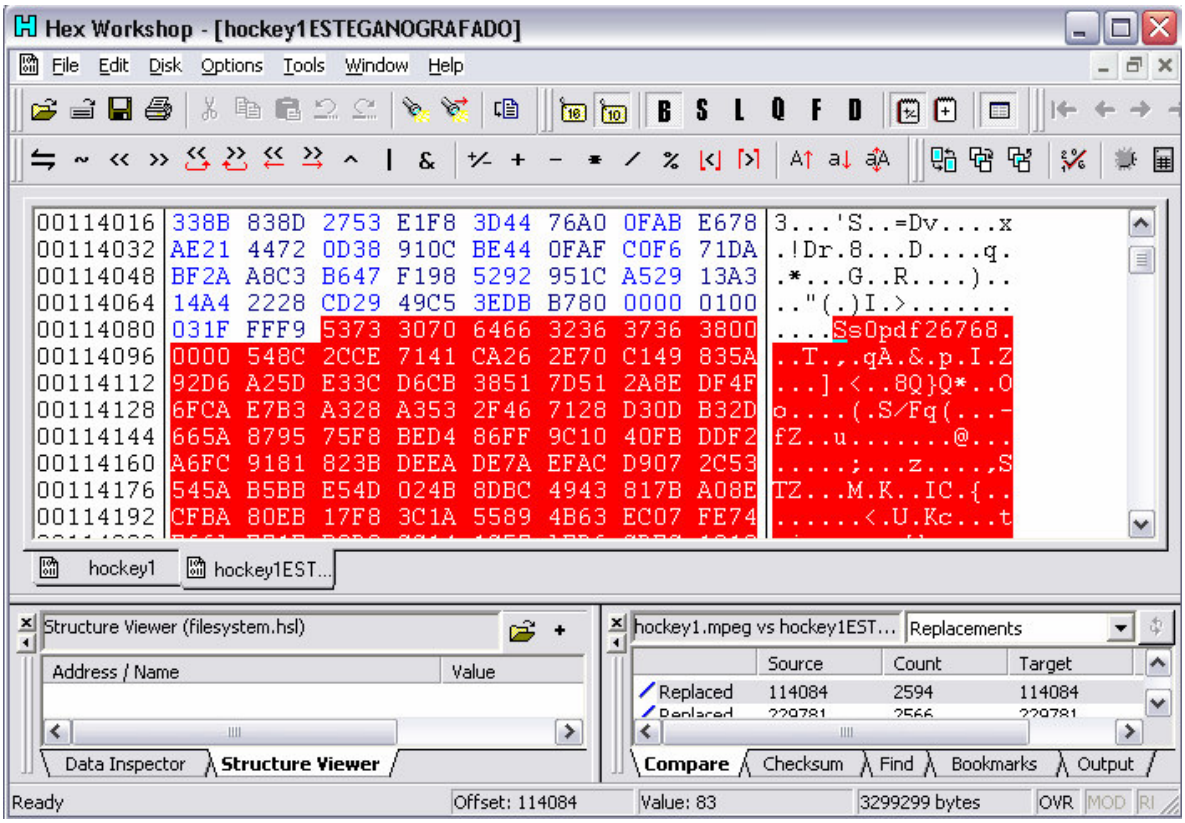

Figura 6.2 – "*hockey1ESTEGANOGRAFADO.mpeg*" (esteganografado)

Fonte: Autor

A figura 6.2 mostra o resultado da comparação entre o arquivo original e o esteganografado. A região em vermelho representa os dados divergentes do arquivo do vídeo "*hockey1ESTEGANOGRAFADO.mpeg*". Neste arquivo, os valores em vermelho representam fragmentos do conteúdo da mensagem ocultada.

Neste caso são mostrados os *bytes* modificados pelo processo de esteganografia.

As regiões que apresentam divergências são os pontos do arquivo no qual o conteúdo da mensagem foi recebido por meio da técnica de esteganografia.

Na figura a seguir (Fig 6.3) é apresentado o resumo das regiões que receberam os dados da mensagem ocultada. Essas regiões estão situadas em todas as figuras com fragmentos da mensagem.

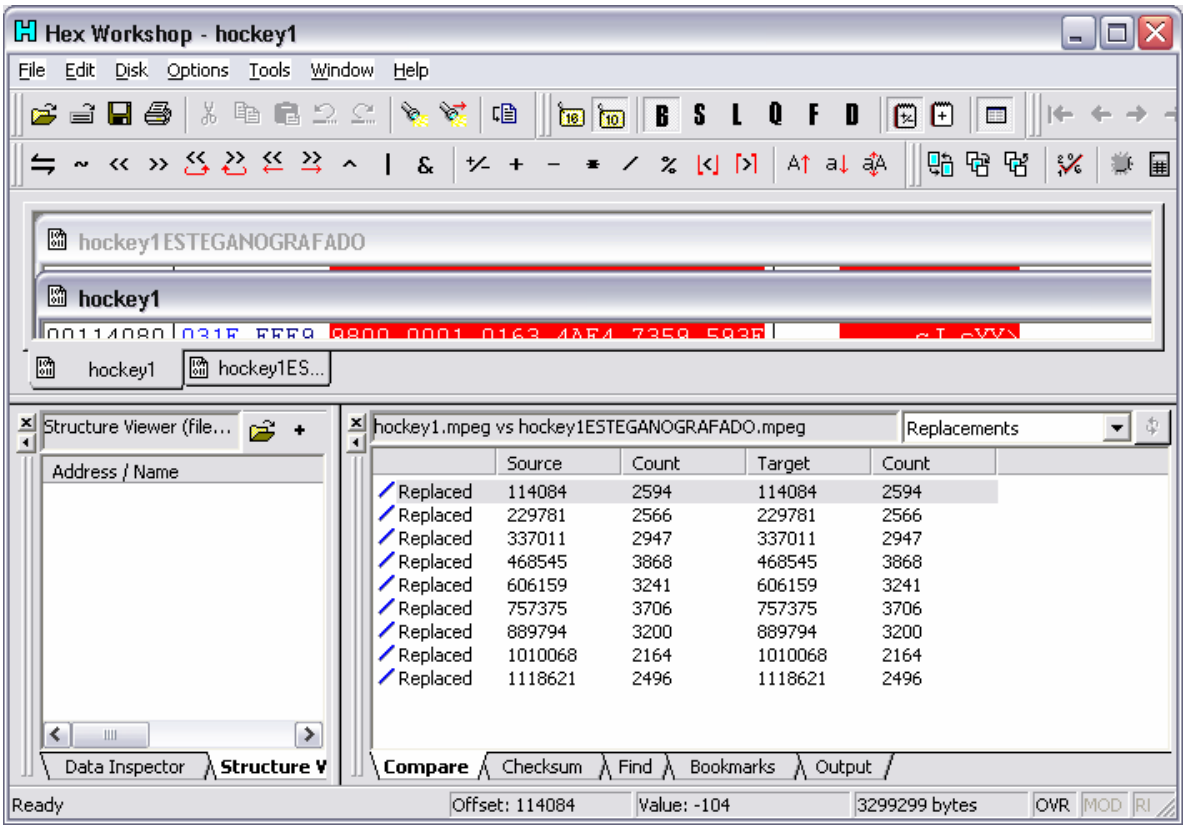

Figura 6.3 – Resultados da comparação binária

Fonte: Autor

Na figura 6.3 há uma tabela com informações referentes às regiões que divergem na comparação de dados. O termo *Replaced* indica a existência de substituição dos bytes originais pelos bytes da mensagem. Já o termo *Source,* indica o byte inicial no qual ocorreu a alteração. O significado do termo *Count* é o número total de *bytes* a partir do *byte* inicial (apresentado no *Source*) no qual ocorreu a substituição de dados. É visível, a partir do número total, o intervalo entre o primeiro e o último dado modificado.

Por fim, foram identificadas nove substituições de blocos de dados a fim de verificar a existência de alterações em arquivos visualmente idênticos.

6.2 – Histogramas

O segundo método de esteganálise adotado foi a geração de histogramas a partir da coleta dos valores em decimal de cada *byte* dos arquivos *"hockey1.mpeg"* (vídeo original) e "*hockey1ESTEGANOGRAFADO.mpg"* (vídeo esteganografado). Nas criações dos histogramas, foi utilizado o programa *"histogramampeg1.exe"* objetivando recolher informações referentes a cada *byte* desses arquivos. A ferramenta *Excel* foi utilizada na geração gráfica dos histogramas. Os dados coletados acerca do arquivo original e do arquivo esteganografado foram usados no preenchimento de duas planilhas. A primeira surge em vista das informações coletadas no arquivo original, enquanto a segunda é feita com base no arquivo esteganografado. Considerando essas planilhas, foram gerados dois histogramas visualizados nas figuras 6.4 e 6.5.

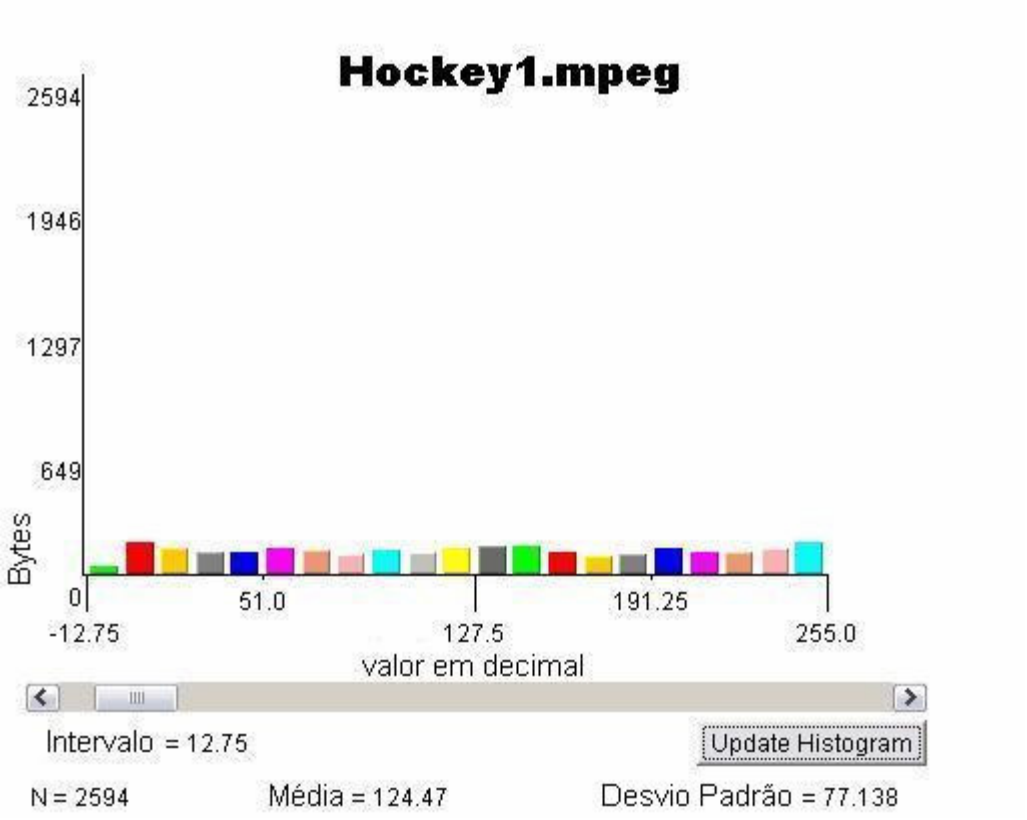

Figura 6.4 – Histograma do vídeo "*hockey1.mpeg*" (original)

Fonte: Autor

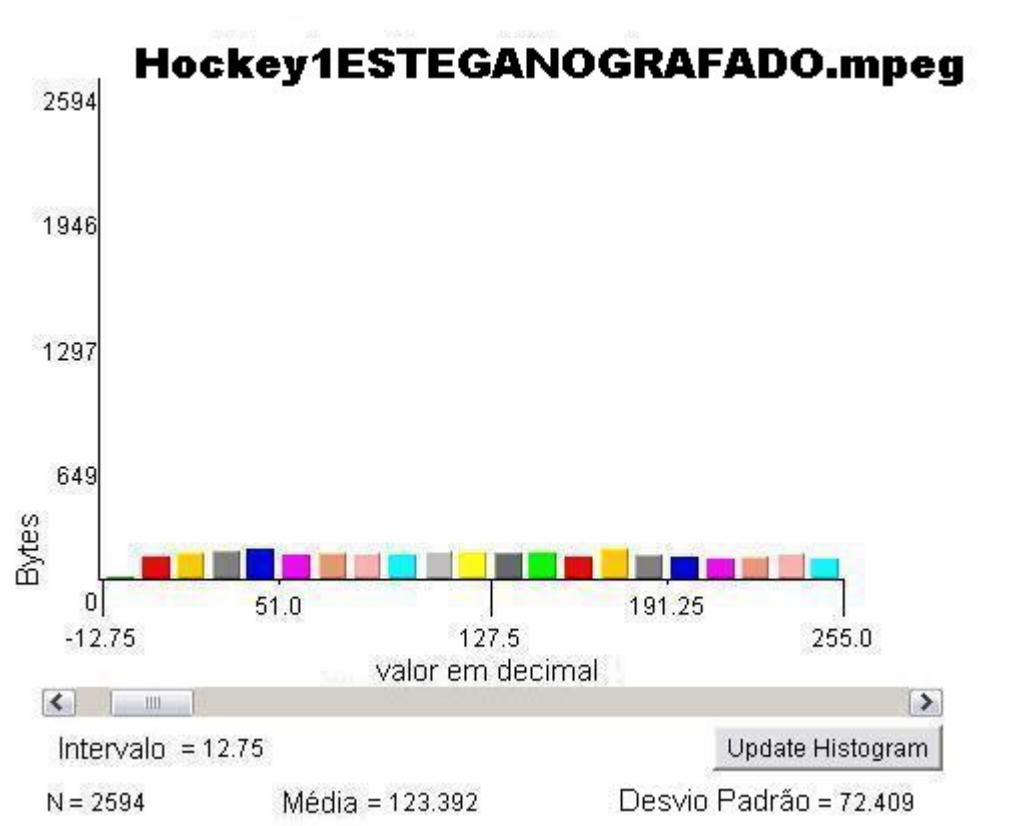

Figura 6.5 – Histograma do vídeo "*hockey1ESTEGANOGRAFADO.mpeg*"
Fonte: Autor

Ao analisar os dois histogramas, pode-se perceber que os valores dos *bytes* de mesma posição do arquivo original e do esteganografado apresentam dirvergências visíveis.

Esses dois histogramas estão analisando apenas os *bytes* referentes à trigésima figura dos vídeos. A razão da escolha dessa figura foi o fato do protótipo utilizar um intervalo de 30 figuras para o processo de esteganografia. Dessa forma, ao gerar os histogramas dessa figura de cada vídeo ficaria possível visualizar a diferença entre os valores dos *bytes* de mesma posição, evidenciado a existência esteganografia no arquivo "hockey1ESTEGANOGRAFADO.mpeg".

Capítulo 7 – Conclusão

7.1- Considerações Finais

Este projeto procurou apresentar uma visão geral da esteganografia, incluindo informações referentes aos seus aspectos históricos e as principais técnicas utilizadas para ocultação de dados digitais, com destaque para a técnica de substituição ou sobrescrita de dados em arquivo de vídeo, e alguns métodos e ferramentas aplicadas na esteganálise.

Além da esteganografia, foi apresentada uma síntese do padrão MPEG-1, o formato de vídeo escolhido para ocultar as informações sigilosas. Nesse estudo foram evidenciados alguns aspectos referentes à evolução desse padrão de vídeo, as principais características da estrutura do MPEG-1 especificadas pela ISO/IEC 11172-2 e os principais métodos de compressão utilizados no padrão.

Outro elemento apresentado neste projeto foi a criptografia DES (*Data Encryption Stardard*), adotada para ampliar a confidencialidade das informações escolhidas para serem mascaradas. Os principais pontos expostos foram a História e fundamentos da criptografia, sua terminologia, o algoritmo DES e as diferenças entre os termos esteganografia e criptografia, freqüentemente confundidos quando se discute as técnicas de segurança da informação.

Por fim, foi apresentada a implementação de um protótipo, que realizou de maneira bem sucedida o objetivo proposto por este projeto - ocultar uma informação dentro de um arquivo de vídeo no formato MPEG-1 combinado com a cifragem da informação por criptografia DES.

7.2- Limitações e Dificuldades

Uma das limitações do protótipo aqui desenvolvido é a aplicação da esteganografia apenas em arquivos MPEG-1 sem áudio, pois as especificações que foram utilizadas para implementação do projeto seguiram a ISO/IEC 11172-2 referente somente ao padrão de vídeo. A especificação que inclui o sistema

MPEG-1 (vídeo e áudio) é descrita na ISO/IEC 11172-1, enquanto a especificação referente ao áudio é ISO/IEC 11172-3. Ambas só podem ser adquiridas por compra por meio da *International Standard Organization*.

Como o objetivo do projeto era esconder uma informação (mensagem ou arquivo) por esteganografia em um vídeo no padrão MPEG-1, apenas a ISO/IEC 11172-2 foi aplicada em sua implementação. Portanto, este projeto não suporta sistemas MPEG-1 com áudio.

Foram encontradas algumas dificuldades durante a elaboração deste protótipo, uma das quais foi o levantamento de informações quanto a técnicas de esteganografia em arquivo de vídeo, visto que a maioria do material encontrado trata de métodos para esteganografia em imagens. Outra dificuldade encontrada foi a mudança da técnica de esteganografia escolhida para implementação.

Na proposta era prevista a implementação baseada na técnica do *bit* menos significativo (LSB). Entretanto, as técnicas de compressão existentes no MPEG-1 dificultaram o acesso e o melhor aproveitamento dos dados de cada figura presente no vídeo.

Dessa forma, ao aprofundar os conhecimentos sobre as demais técnicas existentes, foi possível constatar que a técnica de substituição ou sobrescrita de dados poderia ser aplicada e mais bem aproveitada do que a técnica LSB, pois a substituição de dados foi implementada com a idéia de fragmentar a informação a ser esteganografada e inserir cada fragmento dessa informação em uma figura a cada trinta figuras. A escolha desse intervalo decorreu da comprovação de que os arquivo de vídeo do padrão MPEG-1 têm a taxa de amostragem entre 24 e 30 figuras a cada segundo. Esse valor de intervalo seria suficiente para que a deformação das figuras do vídeo não fosse perceptível à visão humana.

7.3- Projetos Futuros

No entendimento de que os mecanismos voltados para a segurança da informação, por sua própria natureza, são constantemente objeto de ataques, demandando sempre novas atualizações, cabe esclarecer que os estudos

desenvolvidos neste projeto não são exaustivos, podendo servir de base para outros projetos acadêmicos relacionados aos temas esteganografia e criptografia.

A seguir, serão expostas algumas propostas sugeridas pelo autor:

7.3.1- Outros formatos de Arquivo

A técnica de esteganografia pode ser aplicada em outros formatos de arquivo de vídeo como AVI, MOV, WMV e outra versão do próprio MPEG. Além disso, pode-se utilizar também arquivos de texto, como DOC, PDF e RTF, páginas de Internet (HTML), arquivo de som, como WAV, MP3 e WMA, arquivos executáveis (EXE) e formatos de imagens, como JPEG, BMP, GIF e TIFF.

7.3.2- Outras técnicas de criptografia

Pode-se usar, ainda, técnicas diferenciadas de criptografia:

- *Triple DES*: uma alternativa do *DES* original, com a variação de três diferentes chaves;
- *IDEA* (*International Data Encryption Algorithm*): algoritmo muito forte e resistente a muitas formas de criptoanálise;
- *RC2:* algoritmo de chave privada de tamanho variável, proporcionando criptografia em alto volume aliada à performance. É duas vezes mais rápido que o algoritmo *DES*;
- *RC4:* algoritmo com características semelhantes ao *RC2*, porém com uma performance dez vezes mais rápida do que o algoritmo *DES*;
- *MARS:* algoritmo feito pela *IBM* que utiliza bloco de 128 *bits* e uma chave de tamanho variável, indo de 128 *bits* até 400 *bits;*
- *Rijndeal*: algoritmo de blocos iterativos com tamanho de bloco e de chaves variáveis, podendo ser especificados para 128, 192 e 256 *bits*;

7.3.3- Esteganálise e Criptoanálise

Esteganálise é a ciência responsável pela detecção de esteganografia em arquivos suspeitos e Criptoanálise é a ciência responsável pela quebra de

códigos, decifrando a informação, sem conhecer a chave utilizada. São dois temas interessantes que podem ser trabalhados em conjunto para a elaboração de um protótipo que detecte e quebre informações ocultadas ou cifradas.

Referências:

[Almeida, 2006] ALMEIDA, Michello Viana. GUIMARÃES, Paulo Ovídio I., ELIAS, Flávio. Criptografia DES – Data Encryption Standard. Brasília, 2006. Mestrado em Engenharia de Redes e Tecnologias da Informação. UnB – Departamento de Engenharia Elétrica.
Disponível em: http://www.redes.unb.br/security/criptografia/des/ Acesso em: 15 de setembro de 2006.

CASTRO, Alexandre Ramires. ZAVALIK, Claudimir. LACERDA, Guilherme Silva. Um Estudo sobre Criptologia. Acesso em 2 de novembro de 2006. Disponível em: http://br.geocities.com/xandre_l/trab_pesq/publicacoes/

[Chiariglione, 2006] CHIARIGLIONE, Leonardo. MPEG. Moving Picture Experts Group Homepage. Disponível em: http://www.chiariglione.org/mpeg/standards.htm > - Acesso em: 20 outubro de 2006.

COLE, Eric. Hiding in Plain Sight:Steganography and the Art of Covert Communication. Editora Wiley Publishing. Indiana, 2003.

[Fahn, 1992] FAHN, Paul. *Answers to Frequently Asked Questions about Today's Cryptography*. RSA Laboratories, 1992.

Federal Information Processing Standard Publication – FIPS PUB 46-3, "Data Encryption Standard (DES)", National Bureau of Standards (NBS), outubro de 1999. Acesso em 21 de outubro de 2006. Disponível em: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[Fernandes, 1999] FERNANDES, Juliana do Nascimento. Uma Arquitetura de Acesso a Arquivos de Vídeo Digital. Rio de Janeiro, 1999. Projeto Final de Curso submetido – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, p.32.

[Hinz, 2000] HINZ, Marco Antônio Mielke. Um estudo descritivo de novos algoritmos de criptografia. Pelotas, 2000. Bacharelado em Informática. Universidade Federal de Pelotas.

IEC. International Eletrotechnical Comission Homepage.
Disponível em: http://www.iec.org/ Acesso em: 22 de agosto de 2006

ISO. International Organization for Standardization Homepage.
Disponível em: http://www.iso.org./ Acesso em: 22 de agosto de 2006.

[ISO 111772-2, 1993] ISO; IEC. International Standard 11172-2, Video Encoding section. 1993

[KOBUSZEWSKI, 2004] KOBUSZEWSKI, André. Protótipo de software para ocultar textos compactados em arquivos de áudio utilizando esteganografia. Blumenau, 2004. Trabalho de Conclusão de Curso submetido – Ciências da Computação, Universidade Regional de Blumenau, p.25.

MÜLLER, Didier. Stéganographie. Disponível em: http://www.apprendre-enligne. net/crypto/stegano/ > - Acesso em 14 setembro 2006.

[Petitcolas et al., 1999] Petitcolas, F. A., Anderson, R. J., and Kuhn, M. G. (1999). Information hiding - a survey. In Proceedings of IEEE. Special issue on Protection on multimedia content.

[Pfitzmann, 1996 apud Rocha, 2003 p.7-9] Pfitzmann, B. (1996). Information hiding terminology. In Proceedings of the first international information-hiding workshop. Springer–Verlag, Berlim.

[Rocha, 2003] ROCHA, Anderson de R. Camaleão: um software para segurança digital utilizando esteganografia. 2003. 108 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Federal de Lavras, Lavras.

SCHILDT, Herbert. C Completo e Total. 3ª Edição. Editora Makron *Books* LTDA. São Paulo, 1997.

[TKOTZ, 2006] TKOTZ, Viktoria. O algoritmo DES ilustrado. Acesso em 15 de setembro de 2006. Disponível em: http://www.numaboa.com.br/criptologia

[Vasconcelos, 2005] VASCONCELOS, Cristina Nader. Segmentação de vídeo no domínio comprimido baseada na história da compactação. Rio de Janeiro, 2005. Dissertação de Mestrado. Programa de Pós-Graduação em Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[WEBER, 2006] WEBER, Raul Fernando. Criptografia Contemporânea. Porto Alegre, 2006. Instituto de Informática – UFRGS