FERNANDO DE OLIVEIRA GIL
LEANDRO JOSÉ AGUILAR ANDRIJIC MALANDRIN
ROBERTO HIDEO MORIGAKI

SEA – SISTEMA ESTEGANOGRÁFICO DE ARQUIVOS

Trabalho de conclusão de curso de graduação apresentado à Escola Politécnica da Universidade de São Paulo

São Paulo
2007

FERNANDO DE OLIVEIRA GIL
LEANDRO JOSÉ AGUILAR ANDRIJIC MALANDRIN
ROBERTO HIDEO MORIGAKI

SEA – SISTEMA ESTEGANOGRÁFICO DE ARQUIVOS

Trabalho de conclusão de curso de graduação apresentado à Escola Politécnica da Universidade de São Paulo

Área de concentração:
Engenharia da Computação

Orientador: Professor Doutor
Paulo S. L. M. Barreto

São Paulo
2007

DEDICATÓRIA

"Aos meus pais Paulo e Elvira, à minha tia Cida, à minha irmã e às minhas avós, pelo incentivo e compreensão. À Nathalia por estar sempre ao meu lado."
Fernando Gil

"Á minha família. Unida. Sempre."
Leandro Malandrin

"Aos meus pais, pelo seu apoio e incentivo por todo o meu percurso."
Roberto Hideo Morigaki

AGRADECIMENTOS

Agradecemos aos nossos familiares que tanto nos incentivaram e motivaram nos momentos difíceis, para que pudéssemos realizar nossos sonhos. Aos nossos amigos, que proporcionaram bons momentos de confraternização e descontração.

Agradecemos ao Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto, pela orientação que foi essencial para o sucesso deste trabalho. Também agradecemos ao Me. Márcio Augusto de Lima e Silva, do Laboratório de Arquitetura e Redes de Computadores da EPUSP, pela sugestão do tema abordado neste projeto.

Finalmente, a todos os professores da Universidade de São Paulo, que direta ou indiretamente contribuíram na nossa formação.

FERNANDO DE OLIVEIRA GIL
LEANDRO JOSÉ AGUILAR ANDRIJIC MALANDRIN
ROBERTO HIDEO MORIGAKI

SEA – SISTEMA ESTEGANOGRÁFICO DE ARQUIVOS

São Paulo
2007

RESUMO

Nesse trabalho descreve-se um sistema de arquivos diferente, baseado em uma técnica conhecida como esteganografia. Esse sistema de arquivos, chamado SEA, procura esconder não só o conteúdo da informação, mas também a própria existência dos dados, tendo assim o potencial de se tornar uma ferramenta para o armazenamento de informações privativas. Isso foi feito utilizando criptografia e dispersão de dados, de forma a permitir a recuperação de arquivos mesmo quando colisões acontecem devido à impossibilidade de detectar a existência de arquivos previamente armazenados. Sistemas semelhantes já foram criados fora do Brasil. As contribuições desse trabalho são a solução para problemas de segurança encontrados na utilização do conjunto das técnicas mencionadas e a avaliação dos resultados para o armazenamento de arquivos relativamente grandes.

ABSTRACT

This work describes a different kind of file system, based on a technique known as steganography. This file system, called SEA, tries to hide not only the information contents, but also its existence, therefore having the potential to become a tool used for privative information storage. This was accomplished through the use of cryptography and data dispersion, in a way that makes possible the data recovery even when collision happens, due to the fact that there is no way to detect the existence of data previously stored. Similar systems have been already created outside Brazil. The contributions of this work are the solution for security issues found when the two techniques mentioned are used together and the analysis of the results obtained with storage of comparatively large files.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ABREVIATURAS

SEA - Sistema Esteganográfico de Arquivos

FID – File Identificator

FAT – File Alocation Table

API – Application Programming Interface

XML – Extensible Markup Language

CFB – Cipher Feedback

SUMÁRIO

1. INTRODUÇÃO

O SEA é um sistema esteganográfico de arquivos. A esteganografia, apesar do nome complicado, não passa da arte de esconder a existência de alguma coisa, de forma que a torna praticamente impossível de ser descoberta por uma pessoa que não conhece o segredo. Por essa descrição não é tão difícil se encontrar utilizando a esteganografia na vida real.

No entanto, dentro de sistemas computacionais, a utilização da esteganografia é mais complicada. Durante muitos anos, baseou-se nas formas comuns de uso do mundo real para a criação das bases para o uso em sistemas computacionais. Assim, da mesma forma que uma pessoa pode utilizar as chamadas "tintas invisíveis" para escrever mensagens secretas em imagens, utilizam-se bits menos visíveis em imagens digitais para esconder informações. E essa forma de utilização se estende para outras mídias, tais como sons e vídeos. Apesar de muito interessante, esse método de uso tem limitações, pois requer a existência de arquivos de imagem para armazenar conteúdo.

Um sistema esteganográfico de arquivos não tem esse tipo de problemas. Os dados são armazenados no disco rígido da mesma forma que todos os demais arquivos do computador. No entanto, dadas as características do sistema, não podem ser identificados, mesmo que cada byte e porta lógica do hardware sejam investigados.

Essas características também trazem problemas e portanto, novas questões. Por exemplo, quando um novo arquivo é armazenado, não devem existir informações relativas às posições dos dados armazenados anteriormente. Como evitar ou garantir a recuperação de dados mesmo quando pedaços de um arquivo são sobrescritos? Quanto da responsabilidade pela manutenção da integridade dos arquivos deve ficar à cargo do sistema e do usuário? Quais são os tamanhos de arquivo ideais para a utilização em um sistema que permite colisões? Que técnicas de redundância podem ser utilizadas com esteganografia?

O SEA é uma aplicação desse tipo, baseada em idéias de trabalhos semelhantes feitos no exterior. Uma das principais contribuições do trabalho é o levantamento de problemas de segurança que podem acontecer com a utilização da

criptografia e dispersão de dados em conjunto, e a implementação de soluções para esse problema. Além disso, ao contrário dos trabalhos mencionados, foi feita a análise da capacidade do sistema quando trabalhando com arquivos comparativamente grandes. Por último, novas extensões para o sistema são propostas, de onde podem ser originadas aplicações comerciais.

2. JUSTIFICATIVA E OBJETIVOS

A segurança da informação vem recebendo uma atenção crescente devido ao enorme número de novas ameaças que surgem a cada dia. Essas ameaças não são mais direcionadas primariamente à obtenção de dados das grandes e médias empresas mas sim dos usuários comuns.

O problema – e a causa do triunfo dessas ameaças – é que esses usuários normalmente não têm conhecimento suficiente para se proteger de forma adequada. E é por causa disso que surgem os seguintes problemas vêm se tornando mais comuns: roubo de identidade, acesso às informações bancárias confidenciais, revelação de informações confidenciais, etc.

Além disso, os usuários comuns estão sujeitos a um problema muito mais sério. Com a chegada da chamada Web 2.0, técnicas como correlação de dados e buscas avançadas de informações ficam cada vez ao alcance de qualquer pessoa. Isso, juntamente à criação desenfreada de blogs e sites de redes de relacionamento permite que dados não necessariamente secretos, mas pessoais, sejam obtidos, categorizados e muitas vezes utilizados contra o seu dono. É de interesse dos usuários que informações desse tipo não sejam acessadas por qualquer um, protegendo assim a sua privacidade.

Uma das formas de proteção existente, largamente utilizado hoje em dia por grandes corporações, é a criptografia. A criptografia utiliza artifícios matemáticos para transformar o conteúdo de uma informação em um código de caracteres muito difícil de ser quebrado. No entanto, essa situação pede que esses usuários sejam munidos de novas ferramentas para o controle de seus dados.

A esteganografia é uma das opções disponíveis. Essa é uma técnica que procura esconder a própria existência da informação, não só o seu conteúdo, oferecendo assim um nível de segurança mais abstrato, no entanto muito mais poderoso. Ela já vem sendo utilizada atualmente em programas que buscam embutir mensagens secretas em imagens ou em música, sem modificar as características visuais ou sonoras dos arquivos originais.

O SEA utiliza a esteganografia para esconder a existência de qualquer arquivo escolhido pelo usuário. Uma pessoa qualquer, mesmo observando todos os bits

gravados no disco rígido do computador desse usuário não consegue separar o que faz parte dos arquivos originais e o que é informação aleatória, não conseguindo afirmar dessa forma que um arquivo existe de fato. Além disso, uma vez que o SEA se apóia fortemente em criptografia, uma segunda camada de segurança é adicionada, protegendo também o conteúdo das informações armazenadas.

As aplicações desse tipo de sistema são variadas, pois eles permitem aos usuários negarem a existência de informações sem a chance de serem refutados nessa afirmação. Como exemplo pode ser citado o caso de um funcionário de uma corporação que deseja manter secreto um novo projeto com um cliente. Esse projeto, se revelado, poderia indicar para os concorrentes qual é a estratégia da empresa. Outro exemplo seria um chefe de família que deseja manter em segredo a documentação relativa à compra de um carro novo para um dos filhos de forma a só revelar o presente no dia de seu aniversário.

O projeto proposto busca criar um sistema desse tipo, avaliando algumas de suas características - confiabilidade, usabilidade, etc. – além de oferecer respostas para problemas causados pelos altos requisitos de segurança existentes nesse tipo de solução.

3. ESTEGANOGRAFIA

Esteganografia é a arte de esconder informações de forma que nenhuma pessoa, fora o seu criador e a pessoa a qual a mensagem se destina, consiga detectar sequer a existência dessas informações,. A palavra esteganografia vem do grego *escrita escondida*.

Um dos primeiros casos conhecidos de uso da esteganografia é atribuído a um general grego chamado Histiaeus, que tatuou uma mensagem na cabeça de um escravo que teve seu cabelo raspado. Depois que o cabelo do escravo cresceu, ele foi enviado para entregar a mensagem.

O livro Steganographia, publicado em 1499 por Trithemius em três volumes, tratava sobre esteganografia e criptografia, só que disfarçado como um livro sobre magia, espíritos e assuntos do tipo. Os primeiros 2 volumes, aparentemente tratavam sobre o uso de espíritos para a comunicação em longas distâncias. Por volta de 1600, uma "chave" de decodificação do livro foi publicada e descobriu-se que esses volumes tratavam de esteganografia e criptografia. O terceiro volume, até recentemente era acreditado como sendo realmente sobre magia negra. Na verdade, tratavam apenas de mais conteúdo do mesmo tipo, codificado utilizando uma pequena variação da cifra utilizada nos dois primeiros volumes (1).

A esteganografia é conhecida por ser utilizada em setores militares, governamentais e, infelizmente, por entidades criminosas. No entanto, nos últimos anos, observou-se uma tendência crescente do uso dessa técnica também por empresas comerciais, que começam a notar que a criptografia pode não ser a resposta ideal para algumas aplicações.

3.1. ESTEGANOGRAFIA E CRIPTOGRAFIA

A criptografia é uma das técnicas de segurança mais amplamente adotadas atualmente e infelizmente, devido a essa adoção generalizada, passou a ser vista por muitos usuários e empresas como uma técnica mais do que suficiente para

garantir a segurança de informações. Essa técnica utiliza ferramentas matemáticas para manipular os dados originais até obter uma nova seqüencia de dados, derivada a partir da original e de uma chave de criptografia. Sendo assim, a criptografia consegue proteger o conteúdo da informação.

A esteganografia, como mencionado, busca esconder a própria existência da informação. Dessa forma, pode-se entender que a esteganografia apresenta uma proteção em um nível mais baixo que a criptografia, oferecendo assim um maior segurança para os dados.

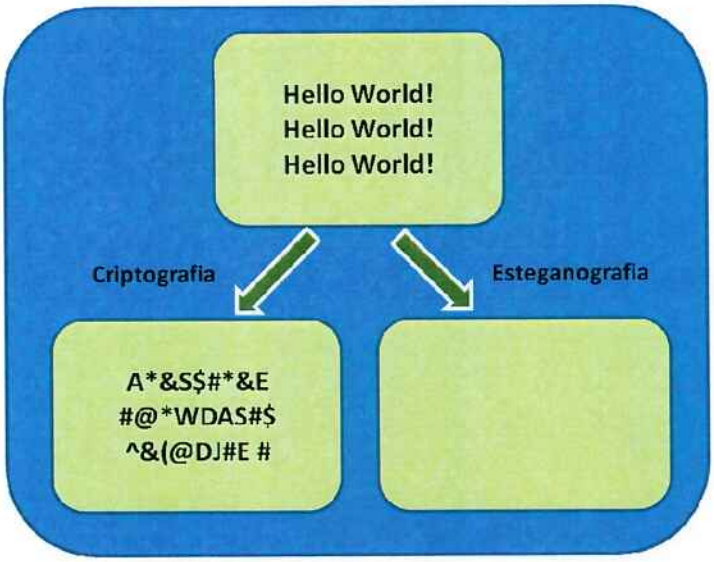


Figura 1 – Diferença entre criptografia e esteganografia

Essas duas técnicas não são de forma alguma incompatíveis. De fato, muitas aplicações buscam utilizar as duas técnicas em conjunto, implementando assim um principio de segurança da informação conhecido como defesa em profundidade. No caso do SEA, a utilização da criptografia além de recomendada, é vital para a viabilidade do sistema.

Esse principio é um dos motivos que vêm levando usuários (tanto pessoas como empresas) por todo o mundo a buscarem novas ferramentas para proteger os seus dados. Durante muito tempo a segurança das comunicações foi baseada diretamente na utilização de firewalls e na criação de canais seguros criptografados. No entanto, notou-se que a partir da mera existência de comunicação entre dois pontos já pode permitir a dedução de informações. Por exemplo: saber que um time de futebol europeu teve uma reunião com um empresário de um jogador brasileiro

pode ser tão importante quanto sabe qual é o valor da venda desse jogador? A esteganografia vai de encontro a esse problema.

3.2. TÉCNICAS CONHECIDAS

Para atender ao propósito de esconder a existência dos dados, a esteganografia depende de dois fatores principais: a criação de um algoritmo que permita esconder uma mensagem em um meio que não fique perceptivelmente modificado e que apenas os dois pontos do canal de comunicação conheçam o algoritmo utilizado. Dessa forma, é possível pensar em inúmeras formas de utilizar a esteganografia, nos mais diversos meios. Por exemplo, utilizando tintas a base de limão ou leite mensagens invisíveis podem ser criadas, as quais só são reveladas quando o papel é esquentado.

Apesar disso, atualmente a esteganografia está associada à técnicas computacionais, envolvendo a modificação de mídias digitais de forma que essas passem a ter mensagens embutidas no seu conteúdo.

3.2.1. Imagens

Provavelmente a forma mais comum do uso de esteganografia. A esteganografia aplicada a imagens busca transformar os bits de pixels menos significativos nos bits da mensagem que se deseja esconder. Logicamente, esse processo normalmente envolve uma perda da qualidade da imagem original. No entanto, dependendo do algoritmo utilizado, a imagem contendo a mensagem secreta e a imagem original não apresentam diferenças que possam ser identificadas a olho nu pelo ser humano.

Existem atualmente diversas aplicações espalhadas pela Internet (2)(3) que se destinam a essa propósito e por talvez por causa desse motivo essa seja a forma mais difundida do uso de esteganografia hoje em dia. Eric Cole, no seu livro Hiding in Plain Sight (4) afirma que após coletar 500 imagens aleatoriamente do site e-Bay e analisá-las com um algoritmo de sua criação que detecta mudanças típicas

utilizadas por softwares de esteganografia, encontrou mais de 150 imagens com mensagens escondidas.

As figuras abaixo, extraídas de (5), mostram a técnica sendo aplicada ao problema de redução de nível de segurança de imagens confidenciais:

Figura 2 – Imagem base

Figura 3 – Imagem confidencial

Figura 4 – Imagem base contendo imagem confidencial

Figura 5 – Imagem confidencial extraída a partir da imagem base

Quando observada com mais atenção, podemos notar que a imagem extraída da imagem base apresenta diferenças em relação à imagem confidencial, assim como que a imagem base contendo a imagem confidencial tem diferenças em relação à imagem base. No entanto, essas diferenças passam despercebidas se cada uma é apresentada separadamente. Para as situações onde se deseja esconder mensagens que não podem ser modificadas (ao contrário do caso acima), alguns cuidados precisam tomados, no entanto a técnica continua sendo basicamente a mesma.

3.2.2. Sons

Também é muito comum a utilização de arquivos digitais sonoros para esconder mensagens através da esteganografia. Em muitos casos, o conteúdo dos bits da faixa de som inaudíveis para o ouvido humano são utilizados como repositório para os bits da mensagem secreta. Na aplicação mp3stego (6), o autor utiliza o núcleo da codificação MP3 para inserir os dados na seqüência de bits que constitui o arquivo de som. A escolha dos bits a serem modificados é feita a partir de uma seqüência pseudo-aleatória baseada no algoritmo de hash.

3.2.3. Vídeo

A esteganografia pode ser utilizada também em vídeo, através de técnica muito semelhante à descrita anteriormente para imagens. Baseia-se na colocação de

pedaços das mensagens em quadros distintos que podem ser recuperados no destino desde que o algoritmo utilizado seja conhecido.

3.3. ESTEGANOGRAFIA EM SISTEMAS DE ARQUIVOS

Apesar de muito utilizadas, as técnicas descritas anteriormente são limitadas no seguinte aspecto: elas assumem a existência de um arquivo em um formato específico e com propriedades especificas no qual a mensagem será inserida. No entanto, para grandes quantidades de dados, esse tipo de abordagem se torna no mínimo suspeita, se não inviável.

Para essas situações surge a necessidade de uma técnica mais abrangente, que permita que mensagens secretas de qualquer espécie tenham sua existência escondida em meio a dados também de qualquer espécie. Em The Steganographic File System (7), é introduzido um sistema de arquivos desse tipo.

O sistema proposto realiza inicialmente a gravação de dados aleatórios no disco rígido do usuário. Esse arquivo é criptografado e quebrado em blocos de dados. Esses blocos são gravados no espaço físico disponível nas posições indicadas por uma seqüência pseudo-aleatória derivada da chave do usuário e de algumas outras propriedades do arquivo e do sistema. Dessa forma, sem o conhecimento da chave do usuário, uma pessoa que está observando os dados no disco não consegue distinguir o que faz parte dos arquivos do usuário e o que é informação aleatória. O sistema de arquivos Mnemosyne (8), apresenta um sistema de arquivos que utiliza o mesmo conceito utilizando uma plataforma distribuída.

Esses sistemas apresentam a vantagem de permitir a gravação de dados em qualquer formato e em qualquer quantidade sobre um substrato de dados aleatórios qualquer.

Entretanto, o fato do sistema não poder armazenar informações sobre a localização dos arquivos (como por exemplo em uma tabela de alocação de arquivos) causa um grande problema. A cada novo arquivo gravado, e até durante a gravação de um mesmo arquivo, existe a possibilidade de que dois blocos diferentes sejam escritos no mesmo lugar, corrompendo a integridade de um dos arquivos do disco.

A solução desse problema envolve a utilização de técnicas de redundância, e constituem um dos pontos mais interessante desses sistemas. A garantia da possibilidade de recuperação do arquivo fica associada a características do arquivo e do sistema. Nunca é possível assumir, dentro do esquema mencionado, que um arquivo tem 100% de chance de ser recuperado com sucesso. Entretanto é possível calibrar variáveis do sistema tais como tamanho dos blocos, tamanho do disco e número de usuários de forma a garantir uma chance muito alta de sucesso. Entre as técnicas de redundância que podem ser utilizadas pode-se citar a cópia dos blocos em diversas posições do volume esteganográfico, como proposto em (7), algoritmos de dispersão de dados, apresentados em (9) e utilizados no sistema Mnemosyne em (8).

O SEA, cujo desenvolvimento é apresentado no restante desse documento, é um sistema desse tipo. Durante o seu desenvolvimento esse e outros problemas foram encontrados e solucionados. Os testes realizados e as conclusões obtidas podem ser encontrados na seção Resultados.

4. ANÁLISE DE SOFTWARE

O SEA é um sistema voltado à segurança de dados e portanto envolve uma série de requisitos funcionais e não-funcionais relativos à manutenção da privacidade dos dados inseridos no sistema pelos usuários.

Logicamente, fora esses requisitos vitais para o desenvolvimento, existem diversas funcionalidades que devem ser implementadas, de forma a permitir que o usuário interaja de forma simples e clara com o sistema. Umas das principais causas da não proliferação e aceitação natural de sistemas de segurança é a dificuldade em estabelecer porque o sistema atende as necessidades do usuário e a dificuldade de utilização desse sistema, devido à falta de conhecimento dos usuários sobre o assunto

O SEA é um sistema com grande enfoque na usabilidade. Espera-se que os usuários, para armazenar arquivos dentro do volume esteganográfico, não precisem entender exatamente o que é feito em cada uma das rotinas executadas pelo sistema.

4.1. VISÃO DO PRODUTO FINAL

Com objetivo de oferecer uma visão mais clara do que é o SEA, essa seção apresenta uma descrição geral do processo envolvido na utilização de um sistema esteganográfico de arquivos.

O SEA, do ponto de vista do usuário, será uma pequena aplicação a ser instalada no seu computador. Essa aplicação, após a instalação, será configurada pelo usuário, através do fornecimento do volume esteganográfico a ser usado, identificação e senha. Caso ele deseje criar um novo volume esteganográfico, essa opção também é oferecida. A aplicação irá pedir qual o tamanho do volume a ser criado e esse espaço será preenchido com dados aleatórios inicialmente. Esse volume constitui o chamado substrato esteganográfico, onde os dados do usuário serão gravados de forma a esconder a sua existência.

O sistema apresenta uma interface básica dividida em duas áreas. Em uma delas, o usuário poderá listar todos os arquivos existentes no seu computador, enquanto que na outra serão apresentados todos os arquivos presentes no volume esteganográfico. Nesse volume o usuário irá criar uma estrutura de arquivos semelhante a que ele tem no seu disco rígido, com diretórios e subdiretórios.

Ao adicionar arquivos na estrutura virtual de diretórios, o sistema realiza todos os procedimentos necessários para gravar o arquivo no substrato esteganográfico. Ao remover um arquivo o processo inverso é feito e o arquivo é gravado de volta no disco rígido do computador.

Além disso, a aplicação também deve permitir a realização de uma renovação dos dados já gravados, uma vez que existe a possibilidade de que dois usuários, utilizando o mesmo substrato ou volume esteganográfico, sobrescrevam blocos de um usuário ou outro. A renovação não é o método principal de garantir a integridade dos dados, uma vez que um algoritmo de dispersão (redundância) é utilizado.

Vale observar que qualquer usuário tem direito a usar um volume esteganográfico já existente, bastando para isso selecioná-lo na interface. Isso permite que uma pessoa possa criar duas estruturas diferentes de dados dentro do mesmo volume esteganográfico, por exemplo uma contendo dados pessoais e outra os dados do seu trabalho. O acesso a cada uma fica assim dependente da senha utilizada. Para o sistema, essas duas estruturas pertencerão a usuários diferentes.

4.2. REQUISITOS FUNCIONAIS

Os requisitos funcionais são todos os requisitos que poderão ser observados diretamente pelo usuário como funcionalidades dentro do sistema. É importante observar que algumas dos requisitos estabelecidos aqui são necessários devido ao fato que esse projeto visa criar um protótipo de aplicação real, assim como estudar as características de performance e confiabilidade de sistemas esteganográficos.

4.2.1. Criação de um espaço lógico

A aplicação irá permitir que o usuário escolha um local dentro do seu disco rígido para criar um espaço lógico a ser utilizado. Esse espaço corresponde ao disco esteganográfico que será preenchido com dados aleatórios formando um substrato para a gravação de dados. O tamanho desse espaço será especificado pelo próprio usuário.

4.2.2. Sistema de arquivos de teste

Será fornecida dentro da aplicação uma forma para o usuário criar uma estrutura de arquivos, com diretórios, a fim de organizar os seus dados. Esse sistema poderá fazer uso de conhecimento de como os arquivos são armazenados no substrato para otimizar sua performance. Deverá ser dedicada uma especial atenção a esse sistema para evitar possíveis falhas de segurança.

4.2.3. Canal de auditoria

A aplicação irá apresentar formas de, através de uma escolha do usuário e certas checagens a serem especificadas posteriormente, recuperar informações sobre a forma com que os arquivos foram gravados no disco esteganográfico. Essas informações não irão de forma alguma permitir a recuperação completa dos dados, no entanto deverão permitir que entidades de verificação descubram falhas de segurança, acessos indevidos aos arquivos de usuários. Além disso esse canal de auditoria, irá fornecer os meios de obter as informações necessárias aos testes do sistema.

4.2.4. Método de renovação

Será possível na aplicação que o usuário decida quando que ele deseja realizar a renovação dos dados no disco, para aumentar a probabilidade do arquivo não estar danificado quando for preciso recuperá-lo.

4.2.5. Rotinas de testes

A aplicação irá apresentar interfaces para obtenção de dados relativos à performance e aos diversos testes não funcionais a serem especificados durante a fase de projeto. Essas rotinas poderão ser ligadas e desligadas conforme a necessidade e não poderão estar presentes em uma versão final da aplicação. Os dados obtidos dessa rotina poderão permitir a recuperação total dos dados caso se ache necessário.

4.2.6. Sistema multiusuário

A criação dos espaços lógicos mencionados não pode ser vinculada ao usuário e sim ao sistema. Dessa forma, deve ser permitido a dois usuários gravar seus arquivos em um mesmo volume esteganográfico caso desejem. A situação de aplica também a um usuário que deseja criar duas estruturas de diretórios diferentes utilizando o mesmo volume esteganográfico.

4.3. REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais correspondem às características desejadas do sistema, mas que não podem ser associadas diretamente a uma funcionalidade do sistema, mas sim ao conjunto de todas as essas funcionalidades.

4.3.1. Arquivos ocultos

Não deve ser possível detectar a existência de arquivos através da observação dos dados da unidade lógica.

4.3.2. Nenhuma informação confidencial será gravada fora do espaço esteganográfico

A partir do momento que o substrato esteganográfico estiver constituído, a aplicação não poderá armazenar de forma permanente nenhuma informação confidencial, codificada ou não fora desse espaço. Entende-se por informação confidencial senhas e qualquer informação que o usuário deseje armazenar de forma esteganográfica. Espera-se que a gravação de dados fora do espaço lógico de armazenamento fique limitada a dados essenciais de configuração da aplicação.

4.3.3. Proteção em profundidade

Mesmo que um atacante tenha acesso à unidade lógica e às seqüências de blocos, os dados armazenados devem ser criptografados, de preferência com dois passos de criptografia, um anterior à fase de dispersão de dados e outro antes da escrita dos dados no substrato esteganográfico.

4.3.4. Proteção contra sobrescrita

O sistema deve permitir que o arquivo seja recuperado mesmo que uma pequena quantidade de blocos tenha se perdido. Os blocos podem ser perdidos por sobrescrita causada por outro usuário ou por outro arquivo do usuário.

4.3.5. Interface gráfica

O sistema deve prover uma interface gráfica para a execução das funções de responsabilidade do usuário.

4.3.6. Aleatoriedade dos dados

Os dados da unidade lógica devem apresentar distribuição aleatória com ou sem a presença de arquivos.

4.3.7. Estrutura de dados

Somente o usuário poderá conhecer a estrutura de dados armazenada. Nenhuma informação de como e quais arquivos estão armazenados estará disponível ao sistema.

4.3.8. Criação de um espaço lógico

A aplicação irá permitir que o usuário escolha um local dentro do seu disco rígido para criar um espaço lógico a ser utilizado. Esse espaço corresponde ao disco esteganográfico que será preenchido com dados aleatórios formando um substrato para a gravação de dados. O tamanho desse espaço será especificado pelo próprio usuário.

4.3.9. Utilização de algoritmos de criptografia conhecidos

O sistema deve utilizar algoritmos de criptografia conhecidos e testados pelo mercado. Incluem-se aqui algoritmos de hash, criptografia simétrica e assimétrica.

4.4. MODELO DE CASOS DE USO

O objetivo dessa seção é especificar com detalhes todas as funções executadas pela aplicação que são derivadas dos requisitos funcionais apresentados anteriormente.

- **Usuário** – Pessoa que deseja armazenar seus arquivos de forma esteganográfica no sistema.

- **Auditor** – Pessoa que irá auditar o sistema. Essa pessoa têm acesso à todas as funcionalidades destinadas ao usuário final, mas pode visualizar as informações de auditoria. Dessa forma, é possível executar uma série de ações

e verificar se o sistema está gravando os resultados dessas ações no formato esperado.

Figura 6 – Modelo de casos de uso

4.4.1. Criar volume esteganográfico

Descrição: Criação de um volume esteganográfico, onde serão armazenados os arquivos que o usuário selecionar dentro da aplicação.

Evento iniciador: Requisição de um usuário para a criação de um novo volume esteganográfico.

Pré-condições: Nenhuma pré-condição.

Pós-condições: Arquivo do volume esteganográfico criado e registrado no sistema.

Fluxo principal:

1. O usuário solicita a criação de um volume esteganográfico.
2. O sistema solicita o nome e a localização do arquivo onde será criado o SEA, bem como o tamanho do espaço de armazenamento disponível.
3. O usuário fornece as informações.
4. O sistema cria um arquivo vazio de acordo com as informações especificadas.
5. O sistema preenche o arquivo com dados aleatórios.
6. O sistema notifica o usuário que o SEA foi criado.

Fluxo alternativo:

Nenhum fluxo alternativo.

Fluxo de exceção:

No passo 4, caso o sistema não consiga criar o arquivo no local especificado:

1. O sistema exibe uma mensagem informando que não conseguiu criar o arquivo no local especificado pelo usuário.
2. Volta ao passo 2.

4.4.2. Abrir volume esteganográfico

Descrição: Seleção de um novo volume esteganográfico, já criado, a ser utilizado pelo sistema.

Evento iniciador: Requisição de um usuário para a abertura de um novo volume esteganográfico.

Pré-condições: Nenhuma pré-condição.

Pós-condições: Volume esteganográfico aberto mostrando os arquivos contidos na estrutura de diretórios.

Fluxo principal:

1. O usuário solicita a abertura de um volume esteganográfico já criado.
2. O sistema solicita o nome e a localização do arquivo onde está o arquivo.
3. O usuário fornece as informações.
4. O sistema abre o volume esteganográfico e requisita ao usuário o seu login e senha.
5. O usuário fornece as informações.
6. O sistema busca os dados existentes para aquele usuário dentro daquele volume esteganográfico e os exibe na área da estrutura virtual.

Fluxo alternativo:

No passo 6, caso o usuário não forneça as informações:

1. O sistema mantém o volume esteganográfico aberto em uso, mas não busca informações relativas ao usuário.

Fluxo de exceção:

No passo 4, caso o sistema não consiga criar o arquivo no local especificado:

1. O sistema exibe uma mensagem informando que não conseguiu criar o arquivo no local especificado pelo usuário.
2. Volta ao passo 2.

4.4.3. Criar estrutura de diretórios

Descrição: Criação de uma nova estrutura de diretórios para um usuário.

Evento iniciador: Requisição de um usuário para a abertura de um novo volume esteganográfico.

Pré-condições: Volume esteganográfico aberto.

Pós-condições: Estrutura de diretórios criada dentro do volume aberto, com nenhum arquivo ou diretório.

Fluxo principal:

1. O usuário solicita a criação de uma nova estrutura de diretórios dentro do volume esteganográfico aberto.
2. O sistema solicita o login e senha do usuário.
3. O usuário fornece as informações.
4. O sistema cria uma nova estrutura de diretórios dentro do volume, sem realizar o armazenamento de nenhuma informação do usuário fora dessa estrutura.

Fluxo alternativo:

Nenhum fluxo alternativo.

Fluxo de exceção:

Nenhum fluxo de exceção.

4.4.4. Abrir estrutura de diretórios

Descrição: Abertura de uma estrutura já existente de diretórios.

Evento iniciador: Requisição de um usuário para abertura da estrutura de diretórios.

Pré-condições: Volume esteganográfico aberto.

Pós-condições: Estrutura de arquivos aberta e sendo exibida para o usuário.

Fluxo principal:

1. O usuário requisita a visualização de seus arquivos dentro do volume aberto.
2. O sistema solicita o login e senha de usuário.
3. O usuário fornece as informações.
4. O sistema busca dentro do volume esteganográfico o arquivo contendo a descrição da estrutura de diretórios do usuário e exibe essa estrutura.

Fluxo alternativo:

Nenhum fluxo alternativo.

Fluxo de exceção:

No passo 4, caso o login e senha fornecidos não permitam a localização de um arquivo contendo uma descrição de estrutura:

1. O sistema exibe uma mensagem avisando que o login e senha fornecidos não permitiram a recuperação de uma nova estrutura.
2. Volta ao passo 2.

4.4.5. Gravar arquivos

Descrição: Gravação de novos arquivos no volume esteganográfico.

Evento iniciador: Requisição de gravação de um novo arquivo dentro da estrutura do usuário no volume esteganográfico.

Pré-condições: Estrutura de diretórios criada ou aberta.

Pós-condições: Estrutura de diretórios atualizada com o novo arquivo do usuário.

Fluxo principal:

1. O usuário indica ao sistema que deseja gravar um novo arquivo.
2. O sistema lê o conteúdo do arquivo e realiza os procedimentos necessários para a sua gravação, no diretório especificado pelo usuário.

Fluxo alternativo:

No passo 2, caso o sistema detecte que um arquivo com o mesmo nome já existe no diretório indicado:

1. O sistema pergunta ao usuário se ele deseja atualizar o arquivo já existente ou não.
2. Usuário indica sua resposta.
3. Volta para passo 2.

Fluxo de exceção:

Nenhum fluxo de exceção.

4.4.6. Ler arquivo

Descrição: Recuperação de um arquivo gravado dentro de um volume esteganográfico.

Evento iniciador: Requisição de leitura de um arquivo já existente dentro de um volume esteganográfico.

Pré-condições: Estrutura de diretórios criada ou aberta.

Pós-condições: Arquivo requisitado gravado no diretório indicado pelo usuário.

Fluxo principal:

1. O usuário indica ao sistema que deseja recuperar um determinado arquivo.
2. Sistema requisita ao usuário onde o arquivo deve ser gravado no disco rígido original.
3. Sistema realiza todos os procedimentos necessários para recuperar o arquivo original do usuário.

Fluxo alternativo:

No passo 2, caso o sistema detecte que um arquivo com o mesmo nome já existe no diretório indicado:

1. O sistema pergunta ao usuário se ele deseja atualizar o arquivo já existente ou não.
2. Usuário indica sua resposta.
3. Vai para o passo 3.

Fluxo de exceção:

No passo 3, caso o sistema conclua que o arquivo não pode ser recuperado:

1. O sistema exibe uma mensagem de erro, avisando que o arquivo não pode mais ser recuperado, pois está corrompido.
2. Sistema recupera todo o conteúdo que é possível daquele arquivo, indicando para o usuário onde ocorreu o problema.

4.4.7. Renovar arquivo

Descrição: Renovação do conteúdo gravado por um usuário dentro de um volume esteganográfico.

Evento iniciador: Requisição de um usuário para que seus arquivos sejam renovados.

Pré-condições: Volume esteganográfico aberto.

Pós-condições: Arquivo indicado com todos os blocos re-gravados no volume esteganográfico.

Fluxo principal:

1. Sistema recebe a requisição que um determinado arquivo deve ser renovado.
2. Sistema requisita ao usuário que digite seu login e senha.
3. Usuário fornece as informações.
4. Sistema lê o conteúdo do arquivo ou arquivos a serem renovados e realiza todos os procedimentos como se estivesse gravando um novo arquivo no mesmo lugar de um arquivo antigo.

Fluxo alternativo:

Nenhum fluxo alternativo.

Fluxo de exceção:

Nenhum fluxo de exceção.

4.4.8. Verificar informações de auditoria

Descrição: Busca dentro do sistema por informações de auditoria.

Evento iniciador: Requisição de um auditor ou desenvolvedor para a visualização de informações de auditoria.

Pré-condições: Nenhuma pré-condição.

Pós-condições: Informações relativas à auditoria exibidas na tela.

Fluxo principal:

1. Sistema recebe a requisição do auditor para exibir o log de auditoria.
2. Sistema requisita ao usuário o login e senha dos responsáveis pela auditoria.
3. Usuário fornece as informações.
4. Sistema exibe as informações relativas à auditoria, executando todos os procedimentos necessários para obter a mesma.

Fluxo alternativo:

Sem fluxos alternativos.

Fluxo de exceção:

No passo 4, caso a senha digitada não seja correta:

1. Sistema exibe uma mensagem indicando que o login e senha digitados estão incorretos.
2. Volta para o passo 2.

5. PROJETO DE SOFTWARE

Essa seção irá apresentar o mapeamento dos requisitos de software e casos de uso apresentados na seção anterior para a implementação final do SEA.

5.1. ARQUITETURA PROPOSTA

O SEA é um sistema de arquivos que depende de duas técnicas em especial para a se tornar viável: criptografia e redundância.

A criptografia é utilizada não só para proteger o conteúdo dos dados dos arquivos do usuário e evitar ataques contra o próprio sistema (veja próxima seção), mas também é vital para que a existência dos arquivos seja camuflada em meio aos dados aleatórios gravados no disco rígido.

A redundância do sistema é necessária para garantir uma chance próxima de 100% que o arquivo será recuperado mesmo para os casos que informações de dois blocos diferentes de dados sejam gravadas no mesmo lugar devido ao requisito do sistema que se refere a não existência de informações sobre os dados armazenados gravadas foram do volume esteganográfico. Essa redundância é oferecida através de uma técnica chamada dispersão de dados, que será detalhada nos próximos itens.

Dessa forma, o SEA apresenta a arquitetura apresentada na Figura 7 – Arquitetura do sistema. Cada um dos módulos apresentados possui métodos relativos tanto a gravação dos dados dentro do volume esteganográfico como relativos a recuperação desses dados. Não foram implementados módulos separados para cada uma das funções. Na figura, todos os módulos do nível 1 não trocam informações entre si e funcionam como módulos de suporte para as operações do módulo do nível 2.

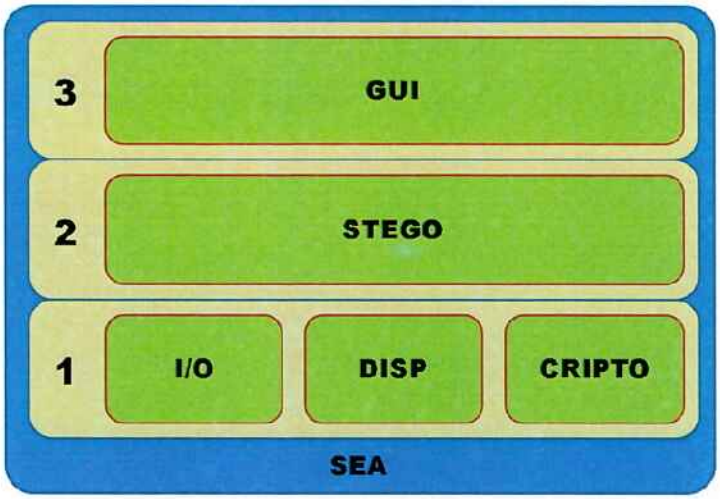


Figura 7 – Arquitetura do sistema

5.1.1. Cripto

O módulo Cripto realiza a criptografia dos dados. Nele existem os métodos necessários para geração das chaves de criptografia a partir dos dados do usuário, configuração de algoritmos de criptografia e configuração de tamanhos de chaves utilizadas. Como será observado adiante, existem algumas vulnerabilidades que podem ser eliminadas com a utilização de criptografia em dois momentos diferentes do processo de esteganografia: antes e depois da dispersão de dados.

Figura 8 – Diagrama de classes – Módulo *Crypto*

5.1.2. Disp

Quando novos dados são gravados no substrato esteganográfico em posições aleatórias desse espaço, não se tem nenhum conhecimento dos dados já armazenados e portanto, colisões podem acontecer. A probabilidade dessas colisões é muito grande, de acordo com o paradoxo do aniversário.

Assim é necessário que algum sistema dentro do SEA crie redundância dos dados. A solução mais simples para esse problema envolve a cópia da mesma informação, ou bloco de dados nesse caso, diversas vezes dentro do espaço disponível. Para o SEA, decidiu-se usar uma abordagem diferente e mais eficiente, conhecida como algoritmo de dispersão de dados.

Essa técnica, introduzida por (9), foi adotada inicialmente por (8) no sistema esteganográfico Mnemosyne. A dispersão de dados funciona da seguinte forma: a partir de um número m de blocos do arquivo original, uma seqüencia de n novos blocos é criada, com n > m. Recuperando-se m blocos dentro dos n criados, recupera-se a informação original, através da resolução de sistemas lineares. Mais detalhes dessa técnica podem ser encontrados na seção Implementação e nas referências citadas.

Comparando a dispersão de dados com a cópia simples, podemos notar que a primeira técnica oferece uma capacidade de recuperação da informação original muito maior que a segunda. Por exemplo, considerando um conjunto de 5 blocos de informação (sendo essa a unidade típica utilizada no SEA para a dispersão de dados). Se a cópia simples fosse utilizada, com fator de replicação de 3, teríamos 15 blocos a serem gravados no disco, sendo que desses, no máximo 3 blocos do mesmo tipo podem ser perdidos. No caso da dispersão, utilizando o fator n típico de 15, teríamos 15 blocos a serem gravados, dos quais até 10 blocos quaisquer podem ser corrompidos algum sistema sem que a informação seja perdida.

O diagrama de classes abaixo permite uma visualização das principais funções do módulo. Pode ser observada a existência de métodos relativos tanto à dispersão dos dados como para a recuperação dos dados, onde é utilizada a classe SystemSolver, que possui métodos relativos a resolução de sistemas lineares.

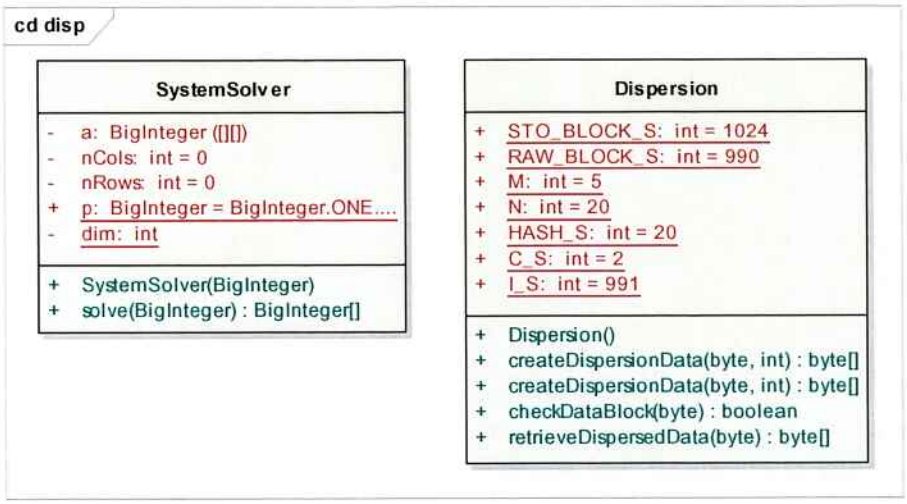


Figura 9 – Diagrama de classes – Módulo *Disp*

5.1.3. I/O

O módulo de I/O corresponde às funções que irão gravar ou ler o volume esteganográfico em uso. Além disso, ele fornece as funções necessárias para determinação da próxima posição onde devem ser gravados os blocos no volume. Observe que esse módulo não é responsável diretamente por algumas das requisições de sistema relacionados com a leitura e gravação de dados no disco rígido.

Neste módulo é onde pode atendimento dos requisitos de auditoria do sistema. Uma vez que ele é o responsável pela manipulação dos dados no volume esteganográfico, é possível monitorar quais usuários estão gravando dados em quais posições. No entanto, as funções de auditoria não se concentram apenas nesse módulo, como era de se esperar.

Pelo mesmo motivo, cuidados foram tomados durante o desenvolvimento desse módulo para não permitir que falhas conceituais, durante o projeto, criassem vulnerabilidades para o sistema como um todo durante a implementação.

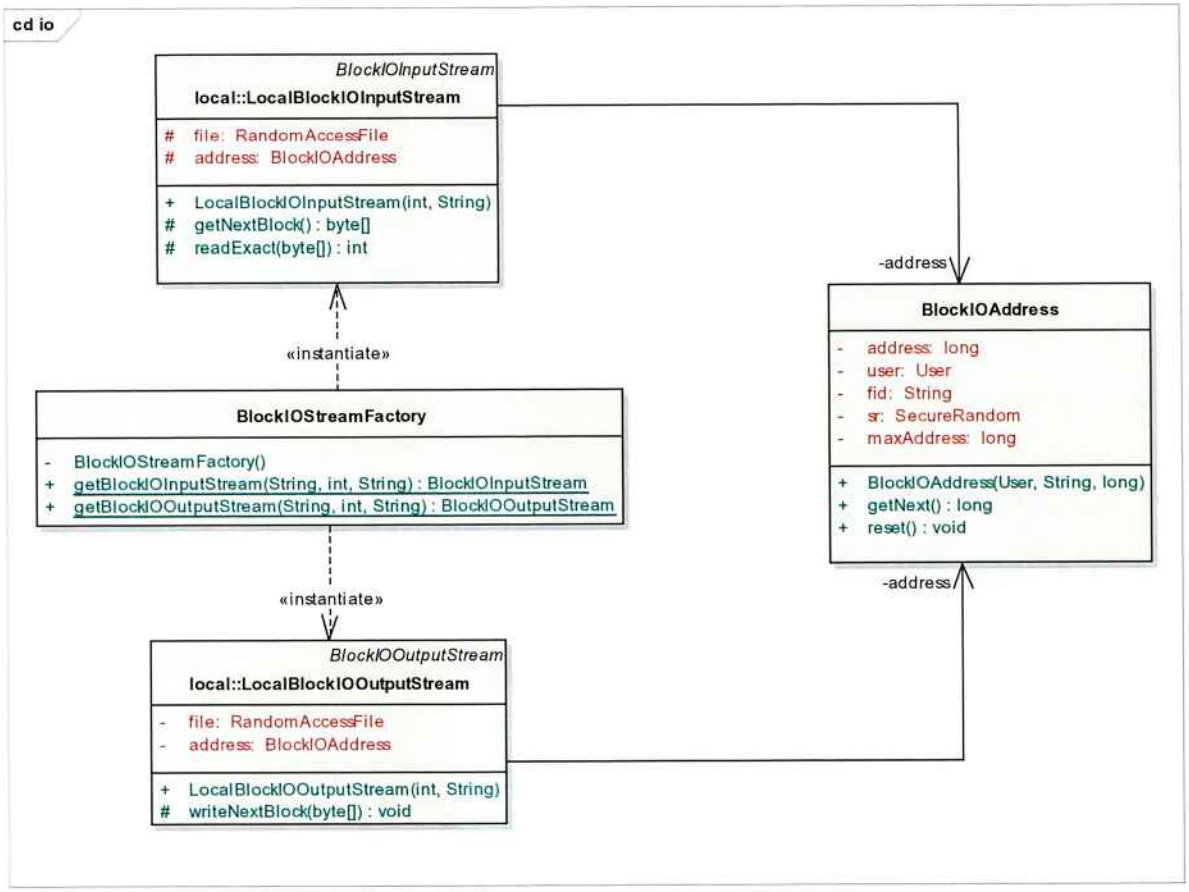


Figura 10 – Diagrama de classes – módulo *I/O*

5.1.4. Stego

O Stego é o módulo mais importante do SEA. Nele estão acumuladas todas as funções que fazem do SEA um sistema esteganográfico de arquivos, além de outras funções essenciais:

- **Controle de dados do usuário –** Os usuários a partir da interface gráfica inserem informações, fazem requisições para o sistema e obtêm respostas. Todas essas informações são geradas ou armazenadas dentro dessa camada do sistema. Como exemplos podem-se citar a criação de diretórios dentro do volume esteganográfico, a inserção de novos arquivos, o controle de chaves de criptografia derivadas das senhas.

- **Chamadas dos módulos –** Na próxima seção serão observadas quais são as funções e a ordem das chamadas realizadas do módulo de esteganografia

para os módulos de suporte da camada 1 da arquitetura apresentada. A coordenação do fluxo de dados entre essas funções é o papel principal do módulo Stego.

- **Comunicação com a interface** – Dado os requisitos de usabilidade do sistema, foi necessária a criação de uma interface gráfica intuitiva para o usuário. As decisões sobre o funcionamento dessa interface ficam também por conta do módulo Stego.

O módulo Stego é dividido em um conjunto de classes principais, ao qual iremos nos referir como main, e a um subpacote, chamado *Stream*.

O conjunto *main* tem como foco principal realizar o controle do volume esteganográfico e dos dados do usuário. Assim, ele possui funções relacionadas a abertura de novas estruturas de diretórios, chamadas no SEA de FATs, criação dos diretórios dos usuários, criação de arquivos e renovação dos arquivos no volume esteganográfico. Uma visão mais detalhada desse pacote pode ser vista na Figura 11.

Já o pacote stream realiza o controle das iterações entre os diversos módulos da camada 3 da arquitetura do SEA. Uma visão mais detalhada desse pacote pode ser encontrada na Figura 12 – Diagrama de classes – módulo *Stego* – pacote *stream*.

Esse pacote utiliza uma série de cadeias de dados concatenadas. Cada uma dessas cadeias é responsável por uma tarefa relacionada a uma etapa do processo de esteganografia e elas são interligadas de forma que a saída de um dos processos possa funcionar como entrada do outro. Apesar de aparentemente direta essa conexão precisa ser controlada pela camada Stego, uma vez que os dados relativos aos testes a serem realizados pelo sistema podem ser obtidos nessas interfaces.

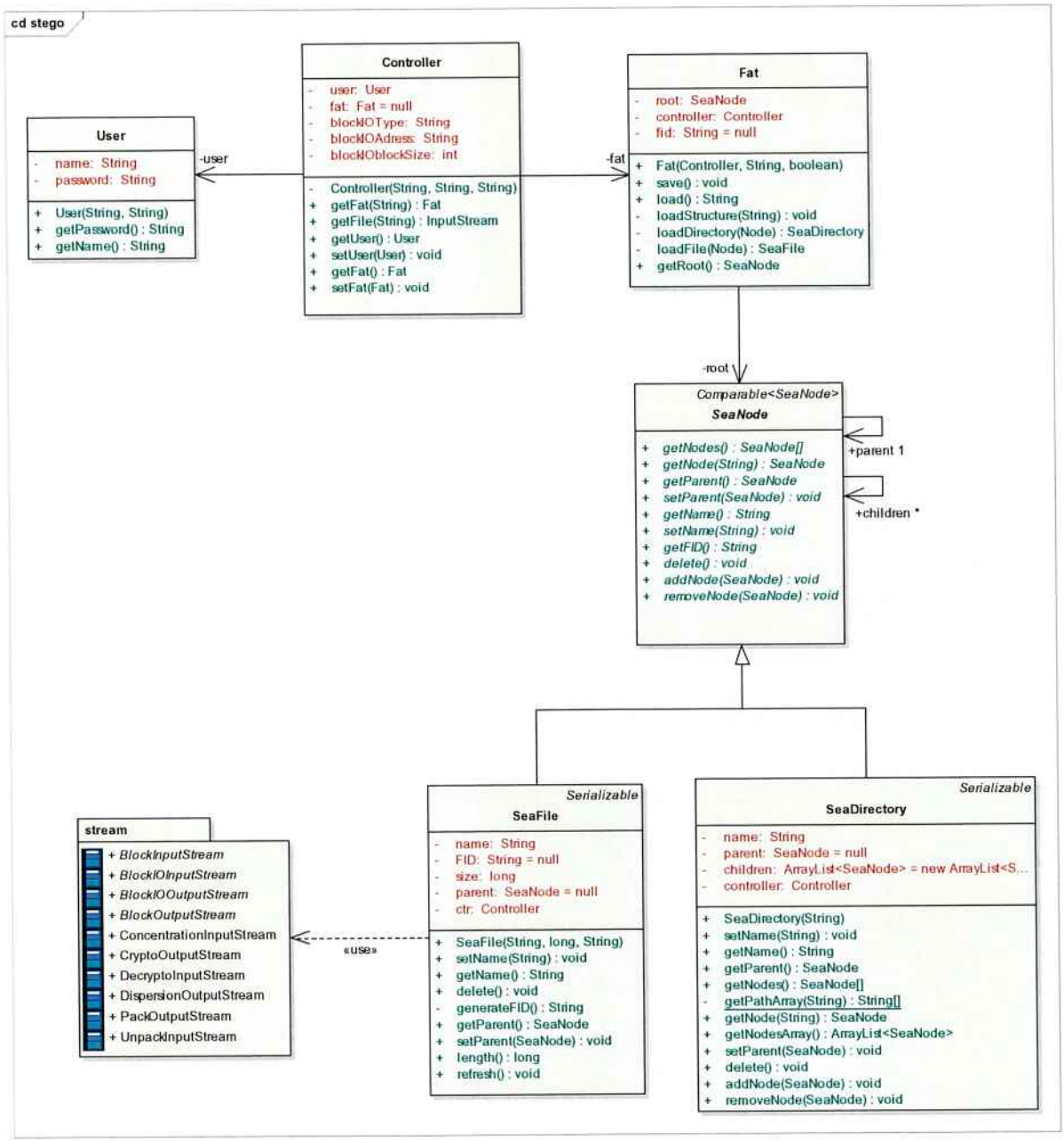


Figura 11 – Módulo Stego – Diagrama de classes principal

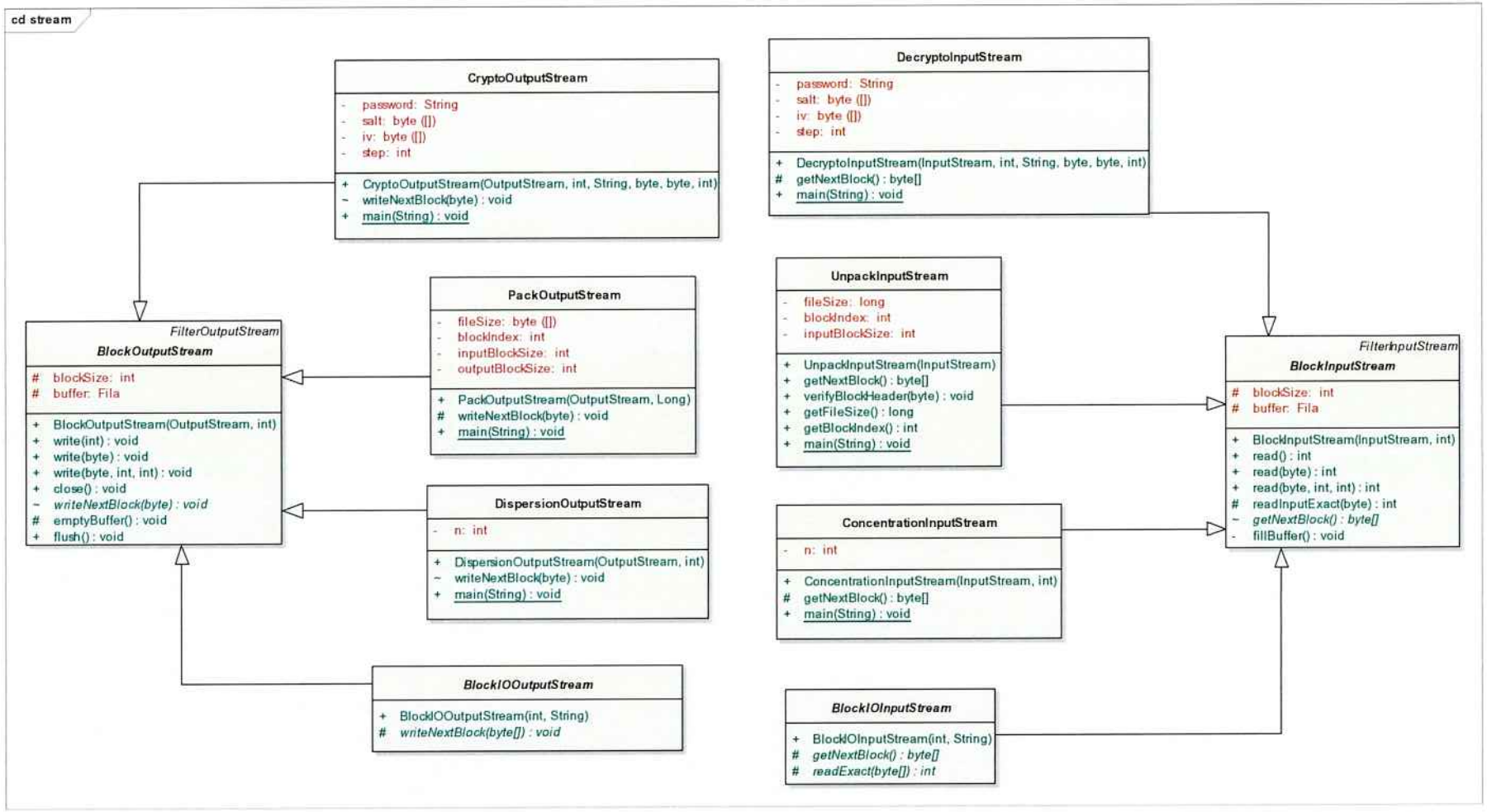


Figura 12 – Diagrama de classes – módulo *Stego* – pacote *stream*

5.2. PROCESSO DE ESTEGANOGRAFIA

Quando um usuário solicita que um arquivo seja gravado no disco esteganográfico, é necessário que uma ordem especifica de tratamento dos dados seja seguida. Numa primeira abordagem do problema, ficou decidido que os dados seguiriam pelo seguinte fluxo dentro do sistema. Essa é a abordagem mais comum nesses sistemas.

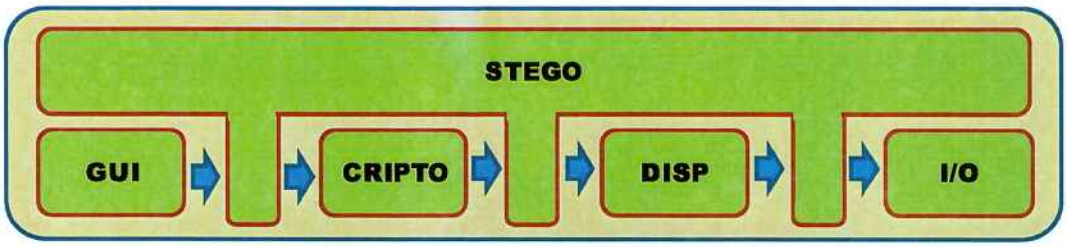


Figura 13 – Processo utilizado na abordagem comum

No entanto, durante o desenvolvimento do projeto, observou-se que possíveis ataques poderiam ser lançados contra o sistema. Essa observação gerou um artigo, o qual pode ser encontrado no Anexo B: Pitfalls in Steganographic File Systems e onde é dada uma descrição mais profunda do problema e da solução mencionados a seguir.

De maneira geral, foi notado que, quando algoritmos de dispersão nos moldes propostos por (9) são utilizados em sistemas de arquivos esteganográficos, da forma indicada na figura acima, é possível que um atacante descubra, através da análise de cada par de blocos gravados no sistema quais são os que pertencem a um mesmo usuário. Esse ataque acontece sobre os dados de saída do módulo Disp na figura.

A partir daí, algumas opções de ataque são possíveis, sendo o pior deles um ataque a disponibilidade do sistema, já que o atacante, por exemplo outro usuário, pode decidir corromper todos os blocos do usuário-vítima.

A solução encontrada, muito simples, é a utilização de um segundo passo de criptografia, realizado após a dispersão dos dados. Dessa forma, o processo final de esteganografia utilizado pelo SEA é apresentado na figura abaixo.

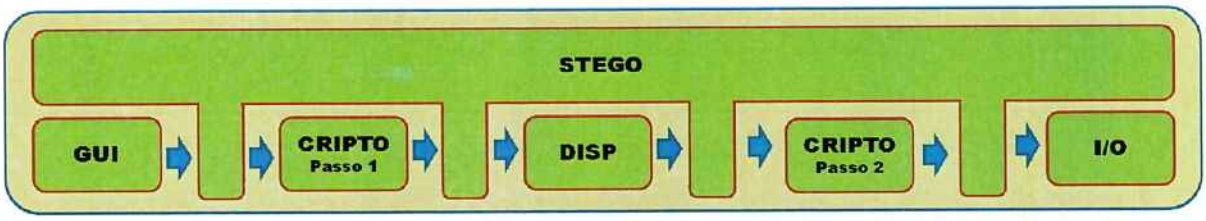


Figura 14 – Processo utilizado na abordagem do SEA

Inicialmente, o usuário seleciona o arquivo que deseja que seja armazenado no seu volume esteganográfico através da interface do SEA, representada pelo módulo GUI.

Essa requisição é passada para a camada Stego, que realiza o controle de todo o sistema, em especial quais dados são passados entre cada módulo, como pode ser observado na figura. A camada Stego, busca agrupar nesse ponto todas as informações necessárias do usuário e do volume esteganográfico em uso naquele momento. Além disso, inicia a leitura do arquivo do usuário no disco rígido.

Tendo todas as informações necessárias, o módulo Stego inicia, através das interfaces fornecidas pelo módulo Cripto, o primeiro passo de criptografia. A criptografia nesse ponto tem o intuito de proteger o conteúdo dos arquivos do usuário e evitar ataques possíveis caos fosse utilizado apenas do Passo 2 de criptografia da figura. O artigo mencionado traz maiores detalhes.

O módulo Stego recebe os dados do usuário criptografados e inicia-se então o processo de dispersão, através do módulo Disp. Nesse passo, devido à utilização de técnicas de redundância, a quantidade de dados inicial aumenta de tamanho. Os parâmetros passados para o módulo Disp determinam o quanto é esse aumento de tamanho. A escolha desses parâmetros é crucial para a confiabilidade do sistema, uma vez que existem diversos fatores que precisam ser balanceados, entre eles o tamanho do volume esteganográfico, o tamanho de cada bloco gravado e o tamanho médio dos arquivos. Uma análise minuciosa desses problemas pode ser encontrada na seção Testes e na seção Resultados.

Os dados recebidos do módulo de dispersão são novamente criptografados, no passo 2 de criptografia do SEA, e finalmente a camada Stego pode iniciar o armazenamento no disco rígido.

Logicamente, esse armazenamento é outro ponto crucial do sistema. É necessário que os dados sejam gravados em posições aleatórias no disco. No entanto, é também necessário que seja possível recuperar esses arquivos em um

segundo momento. Ai entra as funções do módulo de I/O. Esse módulo utiliza informações do usuário, tais como login, senha e informações do arquivo, tais como tamanho, nome e localização dentro da estrutura de dados do usuário para derivar uma seqüencia pseudo-aleatória de números. Esses números são transformados de forma a serem sempre menores que o número total de blocos do volume esteganográfico em uso. Os blocos recebidos do segundo passo de criptografia são então gravados em posições que obedecem a seqüencia criada.

5.3. RENOVAÇÃO DE DADOS

Uma das medidas de avaliação da confiabilidade do SEA é a meia-vida do arquivo armazenado. Uma vez que colisões provavelmente irão ocorrer, com o passar do tempo existe uma maior probabilidade de um arquivo ser corrompido dentro do sistema. Dessa forma, é interessante permitir que seja feito uma regravação dos blocos relativos a um mesmo arquivo, de forma a garantir que todos os dados estão lá. Isso é feito através de uma leitura periódica dos dados armazenados e sua posterior gravação. A esse processo dá-se o nome de renovação.

A renovação dos arquivos deve ser feita pelos usuários do sistema, uma vez que para verificar a integridade dos blocos armazenados é necessária a decriptografia desses dados, a qual depende da senha do usuário. Assim o usuário pode tanto renovar manualmente cada arquivo a partir da aplicação do SEA, como agendar uma tarefa a ser executada de tempos em tempos. Logicamente essa última opção requer o armazenamento seguro da senha do usuário, caso ele não esteja disponível para digitá-la no momento de disparar a renovação.

5.4. MODELO DE ARMAZENAMENTO DE DADOS

Os usuários do SEA podem criar estruturas próprias com os seus diretórios e arquivos nos volumes esteganográficos abertos. Em um sistema esteganográfico

completo, que tenha total controle sobre um dos discos rígido do computador, a esteganografia deveria ser completamente transparente para o usuário, da mesma forma que os sistemas que utilizam a criptografia do disco rígido o fazem atualmente. Assim sendo, o modelo de armazenamento de dados utilizado pelas aplicações poderia ser qualquer um, como por exemplo tabelas de alocação de arquivos ou inodes. Todos os dados pertencentes a essa estrutura seriam tratados pelo sistema de esteganografia da mesma forma que os dados de arquivos comuns.

De fato, em um sistema desse tipo, essa estrutura deveria ficar em segredo, uma vez que um sistema esteganográfico completo não pode relevar nenhuma informação sobre os dados armazenados do usuário. Em resumo, um atacante observando esse disco rígido em um sistema como esse não conseguiria afirmar sequer qual a estrutura de armazenamento utilizada.

Logicamente, devido aos problemas da colisão e a dificuldade em garantir 100% do funcionamento do sistema, fica muito difícil acreditar que esse sistema esteganográfico completo poderia ser instalado no disco rígido principal do computador, o qual abriga o sistema operacional. No entanto, seria perfeitamente viável a sua instalação em discos secundários, reservados ao armazenamento de dados de forma segura.

No entanto, o SEA não é um sistema completo de esteganografia. Ele funciona como uma aplicação sendo executada na máquina do usuário apoiada no sistema operacional e portanto possui certas limitações, sendo que a principal delas diz respeito ao armazenamento de dados.

A primeira limitação fica por conta da necessidade de alocar espaços separados dentro do espaço físico do disco rígido do usuário através da criação de arquivos físicos. Esses arquivos são a imagem dos volumes esteganográficos do SEA.

A segunda limitação, derivada da primeira, corresponde ao problema que não é viável em termos de usabilidade, exigir que os usuários implementem aplicações que criem uma estrutura de armazenamento própria dentro dos volumes esteganográficos do SEA. Além disso, para permitir esse tipo de uso, seria necessária a criação de interfaces e padrões de comunicação entre o SEA e as aplicações responsáveis pela estruturas criadas por cada usuário.

Sendo assim, decidiu-se que propor uma estrutura padronizada para todos os usuários. Essa estrutura segue a mesma idéia da FAT, utilizada pelos sistemas baseados no MS-DOS.

Um arquivo especial, chamado de FAT, é criado e gravado na primeira posição derivada pelo de módulo de I/O de um usuário. Caso o arquivo não caiba nessa posição, ele é gravado nas posições subseqüentes, da mesma forma que um arquivo comum.

O formato desse arquivo pode ser observado abaixo. Utilizou-se XML para definição de diretórios e arquivos.

```
<fat>
        <dir name="Teste">
              <file name="arquivo0.odt" fid="a1d5b55e826a80ed37b"/>
              <file name="arquivo1.odt" fid="12d5basd826asdesdfa"/>
        </dir>
</fat>
```

Neste arquivo a tag *fat* é a raiz do xml, a tag *dir* indica uma pasta que conterá os arquivos, sendo o seu atributo name utilizado para identificação na tela. A tag *file* representa um arquivo, e possui o nome e o identificador (FID). A tag dir pode conter outras tags dir ou tags file dentro dela, permitindo a criação de uma hierarquia.

Cada um dos arquivos é gravado segundo o procedimento descrito na seção anterior. A FAT é apenas um atalho dado ao usuário caso ele não lembre o nome ou caminho do arquivo que ele deseja recuperar.

5.5. INTERFACE GRÁFICA

O SEA é um sistema que lida com duas visões diferentes de dados: o disco real do usuário e o disco "virtual" do usuário, que corresponde à sua estrutura de diretórios. Nada mais normal que a aplicação funcione como uma janela para essas duas visões.

Dessa forma, a interface gráfica do SEA possui uma divisão em duas áreas. Um esquema geral da interface é apresentado na Figura 15 – Modelo da interface gráfica. Na esquerda da interface o usuário encontrará uma representação da estrutura de diretórios do seu disco rígido. Na direita, utilizando a mesma representação, a estrutura de diretórios existentes do volume esteganográfico em uso.

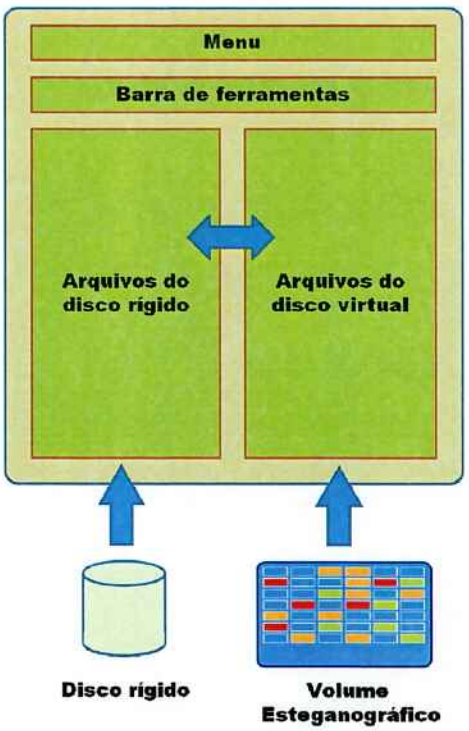


Figura 15 – Modelo da interface gráfica

Além disso a interface possui uma barra de ferramentas que permite ao usuário navegar dentro da cada estrutura de diretórios e modificá-la. Através da interface também é possível transferir arquivos de uma estrutura para a outra, removendo-os da estrutura original ou não. O último elemento da interface é o menu que agrega as funções de configuração do volume esteganográfico em uso.

O projeto da interface busca implementar uma interface com o usuário que siga os padrões gráficos conhecidos nos sistemas operacionais mais conhecidos do mercado.

6. IMPLEMENTAÇÃO

O SEA possui uma implementação funcional, feita utilizando a linguagem de programação Java. Essa seção busca apresentar todos os aspectos principais dessa implementação, em especial detalhes sobre algoritmos utilizados e situações inesperadas durante a fase de projeto que precisaram ser tratadas ou solucionadas.

6.1. DESCRIÇÃO GERAL

Ao abrir a interface gráfica do SEA um novo usuário deve primeiro criar um novo volume esteganográfico, através do menu File > New File System. O sistema irá então solicitar um local no seu sistema de arquivos convencional onde será armazenado o volume esteganográfico, o tamanho dos blocos e o tamanho total do espaço. Após isso, o SEA irá preencher o volume com dados aleatórios, calcular um sal e um vetor de inicialização que será usado nas camadas de criptografia.

O usuário então solicita a criação de uma nova FAT, através do menu FAT > New. Ele deverá então digitar o seu nome de usuário, uma senha e uma frase que servirá como FID da FAT. Nas próximas utilizações do sistema, basta que o usuário carregue a FAT recém criada fornecendo as mesmas informações no menu FAT > Open.

Agora, com o SEA devidamente configurado, o usuário pode armazenar os seus arquivos, assim ele possui algumas opções. A primeira é navegar no painel esquerdo, que representa o sistema de arquivos local, ou convencional, e selecionar o arquivo. A segunda é selecionar um arquivo através do Windows Explorer. Após selecionar o arquivo, o usuário pode utilizar comandos convencionais de copiar e colar (Ctrl+C e Ctrl+V) ou o recurso de drag-n-drop (arrastar e soltar) para colocar o arquivo no painel da direita, que representa o volume esteganográfico. No painel da direita, o usuário também pode criar pastas, para auxiliar a organização dos seus arquivos. Estas pastas existem somente na FAT, não influenciando no processo de

esteganografia, de forma que a movimentação de um arquivo entre pastas não requer a leitura do arquivo.

Para o acompanhamento dos próximos passos a Figura 16 – Descrição geral dos passos para armazenamento e recuperação de arquivos é apresentada.

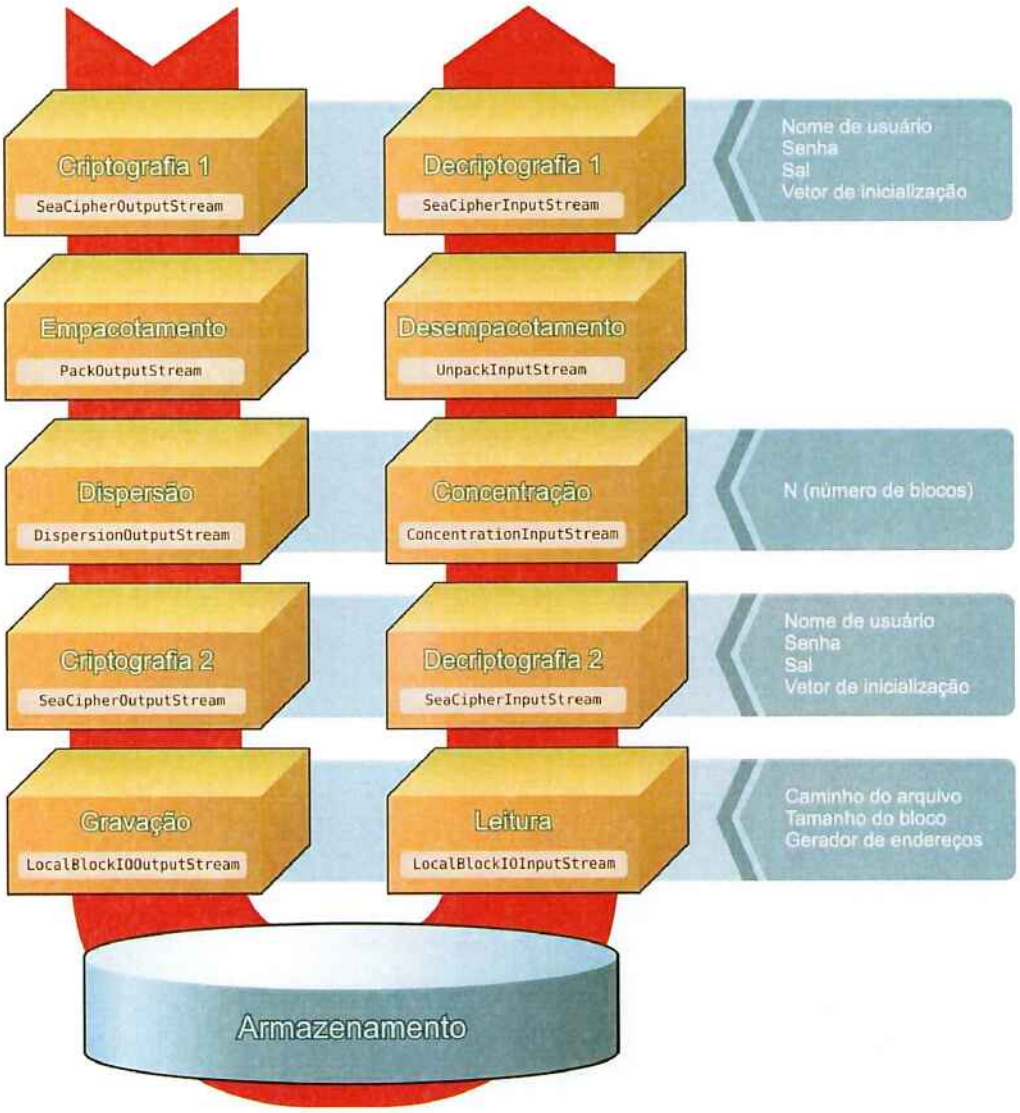


Figura 16 – Descrição geral dos passos para armazenamento e recuperação de arquivos

O SEA então dá início ao processo de esteganografia. O arquivo é criptografado inteiramente, na camada de criptografia 1. Em seguida ele é dividido em blocos na camada de empacotamento, onde cada bloco recebe um cabeçalho de identificação. Os blocos empacotados seguem para a camada de dispersão, que adiciona uma redundância aos dados, pois alguns destes podem ser perdidos por colisões no volume esteganográfico. Os blocos dispersados seguem para a camada

de criptografia 2, onde são cifrados individualmente cada um com uma chave diferente, derivada da chave principal. Os blocos, agora, estão prontos para serem gravados no volume.

A camada de gravação inicia o gerador de números pseudo-aleatórios com as informações do arquivo e do usuário, e então dá início à gravação dos blocos armazenando cada um em um endereço obtido a partir do gerador. Dessa forma o arquivo do usuário está agora "esteganografado" no volume.

Para efetuar a recuperação de um arquivo, o usuário deve carregar a FAT que contém a referência do arquivo, o SEA irá exibir todos os arquivos armazenados através desta FAT. O usuário seleciona o arquivo no painel do volume esteganográfico e o transporta para o painel do sistema de arquivos convencional ou diretamente para o Windows Explorer, tal como foi feito na gravação, porém na direção contrária. O SEA dará início ao processo de leitura do arquivo.

A primeira etapa do processo é, na camada de leitura, iniciar o gerador de números pseudo-aleatórios com as mesmas informações utilizadas para a gravação. Feito isto, a camada passa a ler os blocos no volume seguindo a mesma seqüência de gravação. Os blocos passam pela camada de decriptografia 2, e então são agrupados na camada de concentração.

A camada de concentração testa cada bloco até obter o número mínimo de blocos válidos para a recuperação da informação dispersada e a recupera. Em seguida a camada de desempacotamento verifica se a ordem dos blocos está correta, retira o cabeçalho e os agrupa, formando novamente a unidade do arquivo. A camada de decriptografia 1 decifra o arquivo inteiro obtendo a sua forma original. O SEA então grava o arquivo de volta no sistema de arquivos convencional, conforme especificado pelo usuário no início do processo.

6.2. RENOVAÇAO

O processo de renovação dos arquivos é feito seguindo praticamente as mesmas etapas apresentadas no item anterior. No entanto, dado que o propósito da renovação é meramente regravar os arquivos no volume esteganográfico de forma a diminuir o efeito de colisões, alguns passos podem ser eliminados, em especial os que são destinados a revelar o conteúdo real dos arquivos.

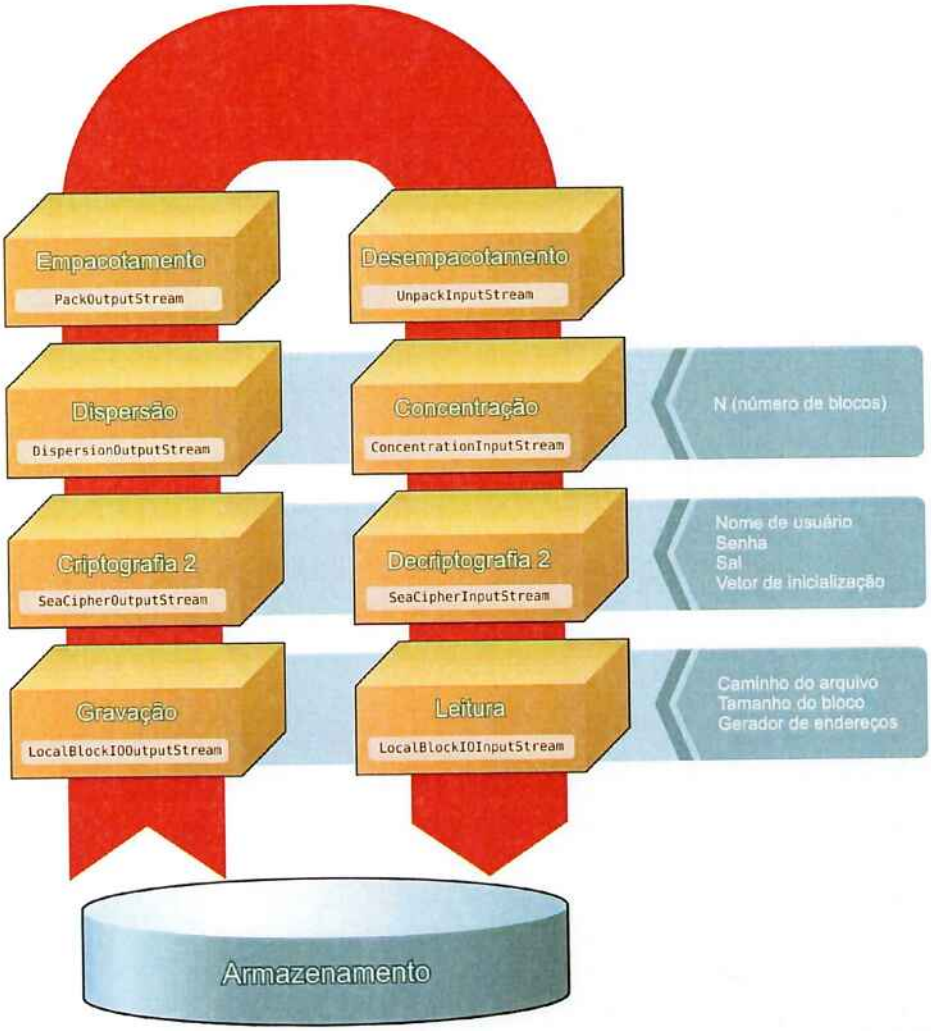


Figura 17 – Descrição geral dos passos para a renovação dos arquivos armazenados.

6.3. CRIPTOGRAFIA

Todas as funções de criptografia do sistema foram feitas utilizando a API conhecida como Java Cryptographic Extension, da Sun Microsystems. Essa API possui implementações de diversos algoritmos, entre eles o AES, o qual foi escolhido como algoritmo principal para a implementação do módulo Cripto do SEA.

Para a maior parte dos sistemas não faz sentido exigir que os usuários se lembrem dos valores das chaves de criptografia utilizadas, e não também não faz parte do escopo do SEA o uso de tokens em hardware ou qualquer outra forma de armazenamento seguro das chaves. Dessa forma, foi necessário utilizar um

algoritmo de derivação paras as chaves de criptografia, a partir das senhas utilizadas pelo usuário durante a esteganografia. O algoritmo escolhido foi o PBKDF2. Esse algoritmo faz parte do padrão conhecido como PKCS#5 – Password Based Cryptographic Standard, e utiliza funções de hash para derivar informação aleatória a partir de uma senha de usuário.

Foi sugerido no artigo gerado a partir do SEA que o segundo passo de criptografia não precisa ser executado com chaves tão fortes quanto o primeiro, uma vez que só visa acabar com as chances de um atacante descobrir os blocos pertencentes ao mesmo usuário dentro do volume esteganográfico. Sendo assim para o primeiro passo de criptografia utilizam-se chaves de 256 bits, enquanto que para o segundo passo, 128. A utilização de chaves de criptografia com menos bits para o segundo passo se baseia no ganho em performance que esse tipo de modificação possibilita.

O modo utilizado para a criptografia é o CFB (Cipher Feedback). O CFB é um modo de criptografia em cadeia, que tem, entre outras características, a que dados podem sejam criptografados no algoritmo escolhido sem que o resultado apresente um tamanho maior que os dados originais. Essa era uma características desejada durante a implementação, uma vez que o controle do tamanho dos dados de entrada e saída de cada módulo precisava ser feito. Além disso, outro motivo para a sua utilização foi que, a partir de certo ponto do projeto, decidiu-se transformar todas as classes que estavam trabalhando com consumo e produção de blocos de dados com tamanhos fixos e passar à utilização de cadeias (InputStream e OutputStream) da API do Java. O uso de criptografia em cadeia permitiu uma maior clareza no trabalho com essas classes, já que não havia a necessidade de se preocupar com dados de *padding*.

6.4. EMPACOTAMENTO

Uma vez que para a recuperação do arquivo só é necessária a seqüência de endereços utilizados para armazenar cada um dos blocos, foi necessária a criação de uma forma de identificação da quantidade de blocos vinculada a cada arquivo. A

maneira encontrar foi a inserção de uma camada secundária, chamada camada de empacotamento.

Nessa camada, cada um dos blocos do arquivo que serão usados com entrada na camada de dispersão, recebe um cabeçalho, que contém o índice do bloco e o tamanho total do arquivo. A figura a seguir mostra os campos de um bloco após o empacotamento.

Figura 18 – Bloco de dados após o empacotamento

No momento do empacotamento, a partir do tamanho do arquivo são calculados quantos blocos de 966 bytes serão necessários para armazenar o arquivo. A camada de dispersão trabalha com 5 blocos de entrada, isso significa que pode não haver dados suficientes para preencher os últimos blocos. Nesse caso a camada de empacotamento preenche os blocos com dados aleatórios, de forma que se tenha um número de blocos múltiplo de 5.

Quando for feito o desempacotamento, os blocos devolvidos pela camada de dispersão são verificados quanto à sua ordem. A seguir, a camada extrai os dados dos blocos recebidos e retorna apenas os bytes referentes ao tamanho do arquivo original. O tamanho do arquivo é armazenado em todos os blocos, pois caso ocorra perda de algum bloco, o restante do arquivo pode ser recuperado. Isso é uma característica importante do sistema, pois a perda de um bloco pode não significar a perda total da informação armazenada.

Dessa forma a camada de empacotamento pega conjuntos de 966 bytes do arquivo, e os transforma em blocos de 990 bytes, prontos para serem dispersados.

É importante que o empacotamento ocorra antes da segunda camada de criptografia pois os dados do tamanho de arquivo poderiam facilmente permitir a um atacante distinguir blocos vazio de blocos utilizados por este arquivo, criando uma séria falha de segurança na esteganografia. Por outro lado, o empacotamento deve ser feito após o arquivo passar pela primeira camada de criptografia para que possa ser feita a renovação do arquivo sem que os dados sejam expostos, conforme foi discutido no item 6.2. A camada de empacotamento foi então colocada logo antes da

camada de dispersão, uma vez que esta já gera blocos de 1 KB otimizados para o armazenamento.

6.5. DISPERSÃO DE DADOS

A dispersão de dados é um das principais técnicas utilizadas dentro do SEA. Como poderá ser observado nos testes e resultados finais, é a redundância criada por essa técnica que permite que os dados sejam recuperados, mesmo depois de diversas colisões tenham acontecido dentro do mesmo volume esteganográfico.

O algoritmo utilizado na dispersão de dados foi baseado no apresentado em (9), no entanto foi simplificado para sua utilização dentro do SEA. A explicação do algoritmo será feita com um exemplo simples e em seguida serão apresentadas as peculiaridades da implementação feita no SEA.

Suponha-se que se deseja aplicar o algoritmo para o texto "ABC". A idéia básica da dispersão é a visualização do conteúdo de entrada como um elemento geométrico, tal como uma reta ou um plano. No caso do exemplo, esse elemento é uma reta. Uma reta pode ser descrita por uma equação do tipo:

$$y = ax + b$$

Nessa equação, os coeficientes que determinam todas as características da reta são *a* e *b*. Dessa forma, para transformar o nosso texto "AB" em uma representação dessa reta, basta escolher um pedaço do texto para representar o coeficiente *a* e o resto para representar *b*. Se tomarmos por exemplo, a = "A" e b = "BC", temos a nova equação da reta.

$$y = "A"x + "BC"$$

Logicamente nesse caso, as partes do texto devem ser transformadas em números utilizando algum padrão conhecido, para a realização dos cálculos posteriores.

Uma reta pode ser sempre definida por dois pontos. Se escolhermos um *x* aleatório na equação acima, pode-se calcular o *y* resultante, e esses dois valores são as coordenadas de um ponto que está na reta descrita pela equação acima. Se continuarmos com esse processo, iremos obter uma série de pontos – $(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots (x_n, y_n)$, sendo n > m – todos pertencentes à reta e todos baseados nos valores de entrada. A partir de dois pontos quaisquer dessa lista pode-se recuperar a informação original, bastando para isso resolver o sistema formado pelas equações:

$$y_1 = r_1 x_1 + r_2$$
$$y_2 = r_1 x_2 + r_2$$

Sendo que os valores obtidos para r_1 e r_2 são "A" e "BC" respectivamente.

Dessa forma, o algoritmo da dispersão busca, a partir da informação de entrada, criar uma série de n pontos que possam originar equações desse tipo. Esses pontos são os dados de saída, que podem ser armazenados em um meio que oferece riscos a integridade dos dados, como por exemplo uma rede de dados, ou os volumes esteganográficos do SEA. Quanto mais pontos criados, maior a chance que o número mínimo deles (no caso do exemplo, 2) seja resgatado e a informação original recuperada.

No SEA, essa técnica é feita de maneira semelhante, só que com m (número de blocos de entrada) igual a 5 e o valor de n (número de pontos gerados) a escolher, sendo 10 o valor tipicamente utilizado. Dessa forma, dentro do módulo Disp, existem dois algoritmos principais implementados: o primeiro é utilizado durante o armazenamento dos dados no volume esteganográfico, chamado de algoritmo de dispersão, enquanto que o segundo é utilizando durante a recuperação desses dados, chamado algoritmo de recuperação.

O algoritmo de dispersão usado pelo SEA pode ser descrito da seguinte forma:

Entrada: 5 blocos de dados do usuário e n

Saída: n blocos de dados

Algoritmo de dispersão:

1. Sejam a_1, a_2, a_3, a_4, a_5 os 5 blocos de entrada. Seja $1 \leq k \leq n$.
2. Repita n vezes os seguintes passos.

a. Escolha c_{k1}, c_{k2}, c_{k3}, c_{k4}, c_{k5}.
b. Faça $y_k = c_{k1}a_1 + c_{k2}a_2 + c_{k3}a_3 + c_{k4}a_4 + c_{k5}a_5$
c. Repita os passos 2 e 3 acima n vezes, armazenando os valores dos coeficientes c_{ki} e do y_k resultante para cada iteração.
d. Gere um novo bloco de saída, concatenando os valores armazenados: (c_{k1} | c_{k2} | c_{k3} | c_{k4} | c_{k5} | y_k)

3. Retorne os n blocos gerados.

Esse algoritmo está esquematizado na figura a seguir.

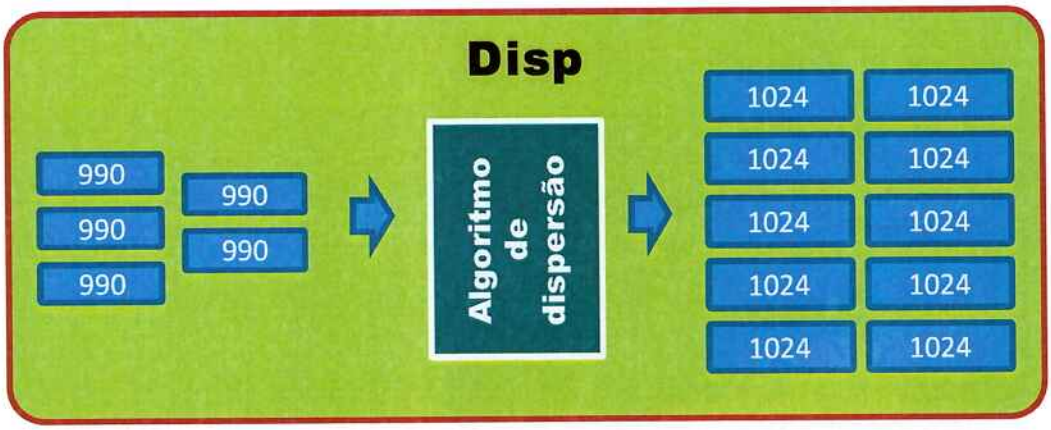


Figura 19 – Esquema geral do algoritmo de dispersão durante o armazenamento dos dados (assume-se n=10)

O algoritmo de recuperação é semelhante e executam o processo contrário, só que como entrada do algoritmo são necessários apenas alguns blocos resultantes do algoritmo de dispersão.

Entrada: 5 blocos de dados

Saída: 5 blocos de dados do usuário

Algoritmo de dispersão:

1. Sejam e_1, e_2, e_3, e_4, e_5 os 5 blocos de entrada. Seja $1 \leq k \leq n$.
2. Repita 5 vezes os seguintes passos
   a. Extraia os valores de c_{k1} | c_{k2} | c_{k3} | c_{k4} | c_{k5} | y_k
   b. Armazene os valores dos coeficientes c_{ki} na linha k de uma matriz C (5x5), e o valor de y_k na linha k de uma matriz Y (5x1).
3. Resolva o sistema C.r = Y:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix} x \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

4. Retorne o valor do vetor r.

Esse algoritmo está esquematizado na figura a seguir.

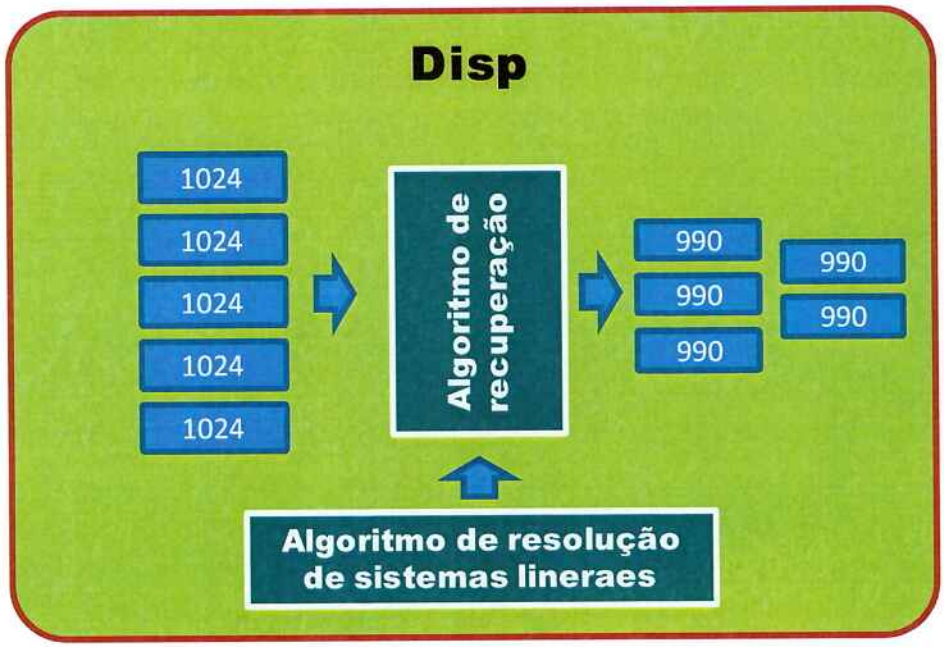


Figura 20 – Esquema geral do algoritmo de dispersão durante a recuperação dos dados

Apesar dos algoritmos serem bem simples, algumas restrições impostas pelo projeto tornou a suas implementações um pouco mais complicadas.

Quando o sistema precisa recuperar uma seqüencia de blocos de dados do usuário, é necessário resolver um sistema linear de 5 equações. Dado que os blocos de entrada não passam de números com um tamanho muito grande, não é viável confiar a resolução desse sistema a métodos que utilizam muitas divisões e arredondamentos.

Para resolver esse problema, o módulo de dispersão, durante a resolução do sistema linear trabalha com aritmética modular. Dessa forma, os valores obtidos na resolução do sistema no algoritmo de recuperação são exatamente os valores de entrada. Algumas manipulações são necessárias, como por exemplo a padronização de todos os valores aleatórios (dados do usuário e coeficientes obtidos dos blocos

dispersados) em números positivos. Além disso foi necessária a obtenção de um número primo com tamanho em bits maior que qualquer bloco de entrada do usuário, para realizar as operações modulares.

Um segundo detalhe é que, de forma a garantir a integridade dos dados e permitir testes do sistema mais simplificados foi adicionado um hash em cada bloco gerado na saída do módulo Disp, durante o processo de dispersão. O valor do hash é calculado com base no valor obtido de (c_{k1} | c_{k2} | c_{k3} | c_{k4} | c_{k5} | y_k) e o módulo de dispersão apresenta métodos para verificar a integridade de um bloco a partir do seu hash. Assim a validade de cada grupo de blocos resgatados do volume esteganográfico pode ser verificada antes que o processo recuperação seja iniciado. Dessa forma os campos de saída dos blocos gerados pelo algoritmo de dispersão são os exibidos na figura a seguir.

Figura 21 – Campos dos blocos de saída do algoritmo de Dispersão do módulo Disp.

Finalmente, devido a um desejo de viabilizar uma extensão relacionada a uma versão distribuída do SEA no futuro utilizando os recursos da Open DHT[1], decidiu-se que o tamanho máximo dos blocos de saída do módulo de dispersão deveria ser 1024 bytes. Assim era necessário escolher os tamanhos de cada campo de forma que não ultrapassem esse tamanho. Essa escolha ficou simplificada pelo uso da aritmética modular, uma vez que nenhum desses valores podia ser maior que o valor do primo escolhido.

Partiu-se do principio que, para aumentar a performance do sistema, seria interessante criar o menor número possível de blocos a partir dos dados do usuário. Dessa forma o valor dos blocos de entrada deveriam ser os maiores possíveis. No entanto, esse tamanho fica limitado pelo espaço disponível para Y dentro do bloco de saída, uma vez que o valor de Y é calculado com base nos dados de entrada.

Também foi escolhido o tamanho mínimo de 2 bytes para os coeficientes aleatórios criados nos passos do algoritmo de dispersão. Uma vez que o primo obtido tinha 991 bytes, e que Y não pode ultrapassar esse valor, o formato final dos

[1] Open DHT – Open Distributed Hash Tables – é uma rede de dados distribuída aberta que funciona no PlanetLAB, patrocinada por diversas empresas e instituições de tecnologia. Nela é possível armazenar dados relacionados à chaves de busca, sendo que o tamanho máximo permitido associado a cada chave é 1024 bytes.

blocos de saída do algoritmo de dispersão é o apresentado na figura abaixo. Os 3 bytes restantes podem ser utilizados no futuro para a adição de controles de seqüência, caso sejam necessários, ou para outras funcionalidades.

Figura 22 – Formato final do bloco de saída da dispersão dos dados (Total = 1024 bytes)

O byte mais significativo de Y é inserido nos blocos de saída com valor igual a 0. Isso porque é necessário manter o padrão de números positivos necessários para o trabalho com aritmética modular durante o algoritmo de recuperação. Assim, os dados de entrada são sempre blocos de 990 bytes.

6.6. ARMAZENAMENTO

A camada de armazenamento é responsável pela gravação e leitura dos dados em um meio físico. A implementação desta camada foi feita de modo a ser flexível suficiente para que o formato de gravação pudesse ser alterado afetando minimamente o resto do sistema, de modo que extensões como a descrita na seção 9 sejam facilmente acopladas ao SEA. Para se obter isso foi utilizado o design pattern conhecido como Abstract factory, como descrito em (10). O padrão foi ligeiramente alterado como se pode ver na figura a seguir.

Desta maneira, o processo de esteganografia utiliza somente as classes abstratas BlockIOInputStream e BlockIOOutputStream, sendo a implementação concreta bastante desacoplada do resto do código.

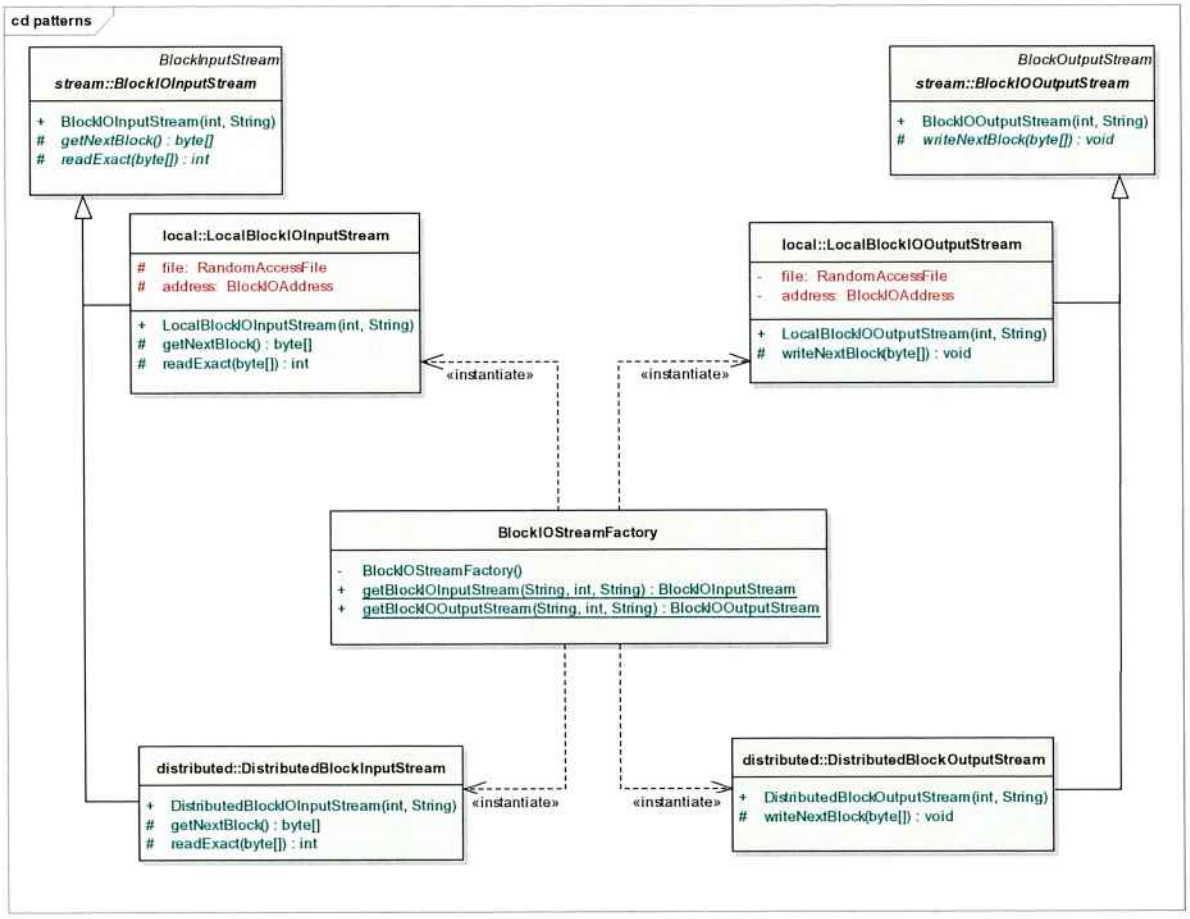


Figura 23 – Diagrama de classes final do módulo I/O
(exibindo um exemplo de criação de um segundo set de classes de armazenamento – não utilizado pela implementação atual do SEA)

Atualmente o armazenamento é feito em um arquivo convencional. A localização desse arquivo, bem como o seu tamanho é configurável no arquivo de propriedades do SEA. O usuário pode criar este arquivo através de um comando da GUI, onde ele especifica o caminho e o tamanho desejados. Assim que o usuário confirma os parâmetros o arquivo é criado e inicia-se o processo de gravação de bytes aleatórios por toda a extensão deste novo espaço esteganográfico. Ao fim dessa etapa, o volume está pronto para receber novos dados.

Quando um novo arquivo está para ser gravado no SEA é gerado um identificador único – FID – utilizando o algoritmo SHA-1 (11) que recebe como semente o nome de usuário, sua senha, o nome do arquivo a ser gravado, o sal e o tempo atual no formato Unix timestamp. O FID é de vital importância tanto para etapa de gravação como para a de leitura, pois ele é utilizado para definir em quais posições do espaço esteganográfico serão gravados os dados.

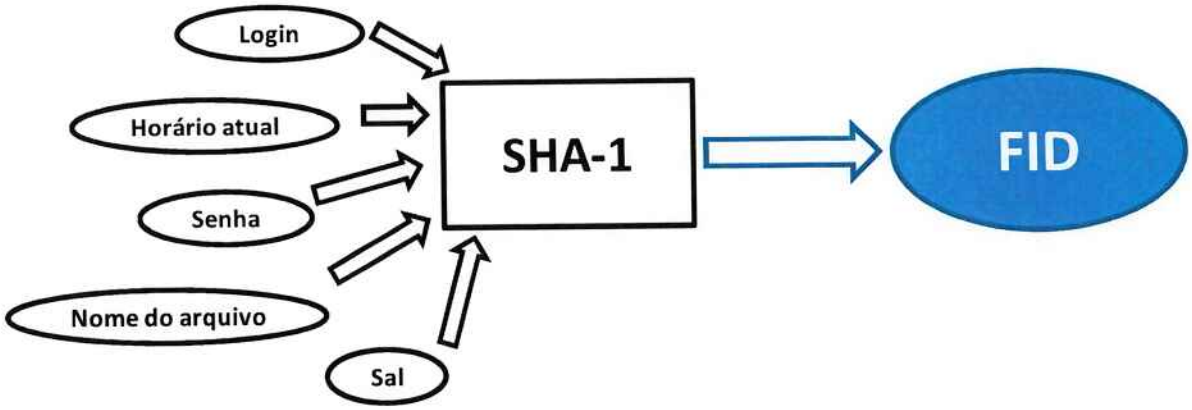


Figura 24 – Geração do FID para cada arquivo

6.6.1. GERAÇÃO DE ENDEREÇOS

O espaço esteganográfico é dividido em blocos de 1024 bytes. Cada um destes blocos recebe um endereço seqüencial que se inicia em zero com incrementos de 1.

Figura 25 – Esquema de numeração utilizado dentro do volume esteganográfico

Para cada bloco recebido da segunda camada de criptografia é necessário definir um endereço, que precisa ser pseudo-aleatório para que um atacante não possa identificar os blocos de dados a partir das posições de gravação. Optou-se por utilizar um gerador seqüencial de números aleatórios, que recebe como semente o FID. A cada iteração esta semente é modificada usando uma fórmula linear congruencial (12). É importante notar que os endereços dos blocos de cada arquivo dependem somente de seu FID, não sendo preciso armazenar as posições de cada bloco para posterior leitura. Basta alimentar o gerador com o mesmo FID que a seqüência produzida será exatamente igual. Isso permite que usuários que sabem todas as informações sobre o seus arquivos possam acessá-lo diretamente no

volume esteganográfico, se a necessidade de abrir a sua estrutura esteganográfica. Essa característica pode ser útil quando se deseja abrir um arquivo por exemplo, para um colega de trabalho sem revelar as demais informações armazenadas na estrutura de diretório. Afinal de contas, um dos motivos para se usar esse sistema é evitar que outros tenham conhecimento da existência das informações.

6.7. INTERFACE

A interface com o usuário foi planejada para ser semelhante ao de softwares gerenciadores de arquivos mais conhecidos do mercado, como o Microsoft® Windows® Explorer e o Norton® Commander®, de modo a facilitar o uso por usuários que já estejam familiarizados com o uso destes programas. Todos os comandos disponibilizados pela API do SEA podem ser acionados através de widgets gráficos, tornando o uso de comandos de texto desnecessários.

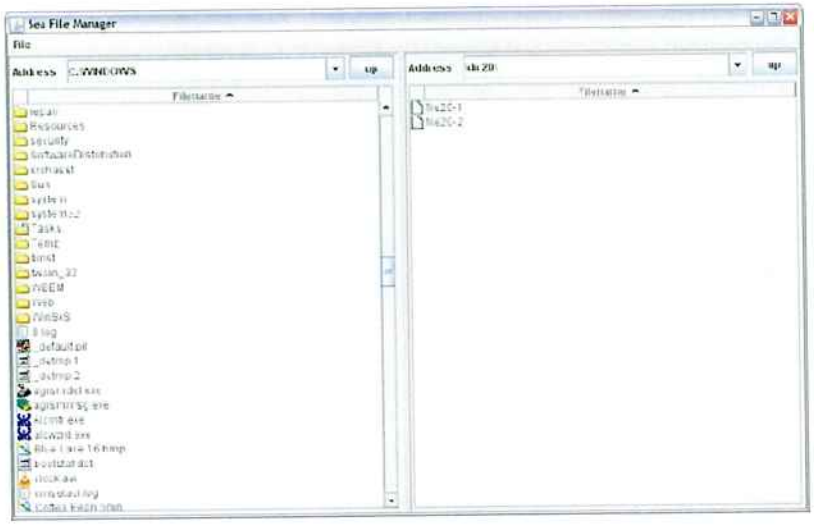


Figura 26 – Interface gráfica do SEA

Para a criação dos componentes gráficos foi utilizado o kit de ferramentas Swing, disponibilizado junto com a distribuição da plataforma Java SE da Sun. Ele provê um conjunto de funcionalidades que engloba completamente as necessidades

para a criação de interfaces gráficas com o usuário (GUI – graphical user interface) desejadas, sendo estas:

- **Componentes de GUI** – O Swing contém diversos componentes pré-fabricados prontos para o uso, indo desde os mais básicos, como botões, checkboxes e caixas de texto, até componentes avançados como caixas de diálogo padrão para abrir e salvar arquivos, campos de digitação de senha entre outros.

- **Look-and-feel configurável** – o Java, por ser definido como uma linguagem multi-plataforma, não possui acesso direto às widgets de interface gráfica dos sistemas operacionais nativos, utilizando por padrão um estilo gráfico próprio que não é necessariamente semelhante ao do sistema operacional do usuário. Desta maneira, as janelas, botões e demais componentes são exibidos com uma aparência diferente dos demais programas instalados, podendo causar desconforto para os usuários. No entanto é possível substituir este estilo padrão por outros à escolha do desenvolvedor. Na distribuição padrão da plataforma Java SE já existe um estilo que simula a aparência das interfaces gráficas do Microsoft® Windows®, e esta foi utilizada nesta implementação do SEA.

- **Transferência de dados** – A transferência de dados através de aplicações via operações de copiar, cortar e colar, além de gestos de mouse para selecionar e arrastar, são essenciais para o usuário executar os comandos para manipular os arquivos no SEA de maneira intuitiva. Tais operações são suportadas pelos componentes do Swing tanto para transferências dentro da mesma aplicação como entre o SEA e aplicações nativas.

O gerenciador de arquivos do SEA é composto por apenas três telas:

- **Login** – É a tela inicial do sistema, onde o usuário digita seu nome de usuário, senha e o identificador do sistema de arquivos (FAT – file allocation table). O significado destes campos são explicados em mais detalhe no

capítulo referente à camada de esteganografia. Após preencher estes campos e confirmar clicando em OK o usuário é direcionado à tela principal.

- **Tela principal** – Dividida verticalmente em dois painéis, a tela principal permite ao usuário navegar pelo sistema de arquivos nativo (fora do sistema esteganográfico) no painel esquerdo, examinando o conteúdo dos diretórios e realizando operações de mover e copiar arquivos sem precisar sair do gerenciador. Já o painel direito exibe os arquivos armazenados esteganograficamente na FAT definida na tela anterior, podendo realizar as mesmas operações que no painel esquerdo.

- **Configurações** – Nesta tela, acessível através da tela principal, o usuário tem a possibilidade de alterar as configurações do SEA, assim como personalizar suas preferências no gerenciador de arquivos.

7. TESTES E RESULTADOS

Durante as apresentações realizadas a respeito do SEA, ficou claro que a idéia da esteganografia em um sistema de arquivos não é exatamente clara. Menos aceita ainda é a idéia da utilização de um sistema que aceita colisões dos dados armazenados.

Os testes aqui descritos têm como objetivo esclarecer de forma concreta – através de resultados – porque essa visão é limitada. Essa seção apresenta um simulador criado com o propósito especifico de gerar esses resultados e mostrar que o sistema apresenta a garantia de integridade dos dados esperada. Além disso o simulador auxilia na calibração do sistema no que diz respeito a parâmetros internos, tais como número de replicações de dados após a dispersão, capacidade a ser oferecida no volume esteganográfico e tamanhos dos arquivos a serem utilizados.

7.1. PLATAFORMA DE TESTES

A plataforma de testes criada consiste em dois módulos separados. O primeiro deles, chamado *test* é responsável pela execução dos testes e é um simulador de algumas tarefas executadas pelo SEA. O segundo, chamado stat, é responsável pela leitura dos resultados obtidos e obtenção de dados estatísticos tais como média e desvio padrão.

Antes de detalharmos a arquitetura da plataforma, seguem alguns termos e variáveis utilizadas durante a explicação:

- **bloco-raw** – bloco de tamanho 990B, correspondente a um pedaço da informação do usuário
- **bloco-sto** – bloco de tamanho 1024B, correspondente a um pedaço da informação que é gravado no disco

- **bloco-disp** – referência ao conjunto dos dados que são dispersados na mesma passagem do algoritmo de dispersão. Antes da dispersão, correspondem a M blocos-raw e depois da dispersão a N blocos-sto
- **colisão Fatal** – ocorre quando mais de (N – M) colisões acontecem no mesmo bloco-disp após a dispersão, que inviabiliza a recuperação completa do arquivo
- **colisão** – ocorre quando um bloco-sto é sobrescrito por outro
- **tamanho do volume** – espaço em bytes ou em bloco-sto do volume esteganográfico
- **tamanho do arquivo** – espaço em bytes ou em bloco-sto dos arquivos de testes
- **M** – tamanho de um bloco-disp
- **N** – número de bloco-sto resultantes da dispersão de um bloco

O módulo test simula a existência de uma série de arquivos a serem gravados no volume esteganográfico. O volume esteganográfico também é simulado, no caso, por um vetor de blocos. Cada uma das posições desse vetor armazena a identificação do arquivo ao qual o bloco pertence, uma identificação do bloco-raw e uma identificação do bloco-disp. A cada gravação de um arquivo, todos os blocos desse vetor correspondentes a esse arquivo são identificados como pertencentes a este arquivo.

Para determinação de uma posição para gravação de um bloco pertencente a um dos arquivos foi utilizada a própria função de geração de posições pseudo-aleatórias presente no módulo *I/O* presente na implementação do SEA. No caso, todos os nomes dos arquivos, usuários e senhas foram gerados aleatoriamente, de forma a criar FIDs diferentes para cada arquivo. A figura a seguir apresenta de forma clara como é a relação do SEA com a sua plataforma de testes.

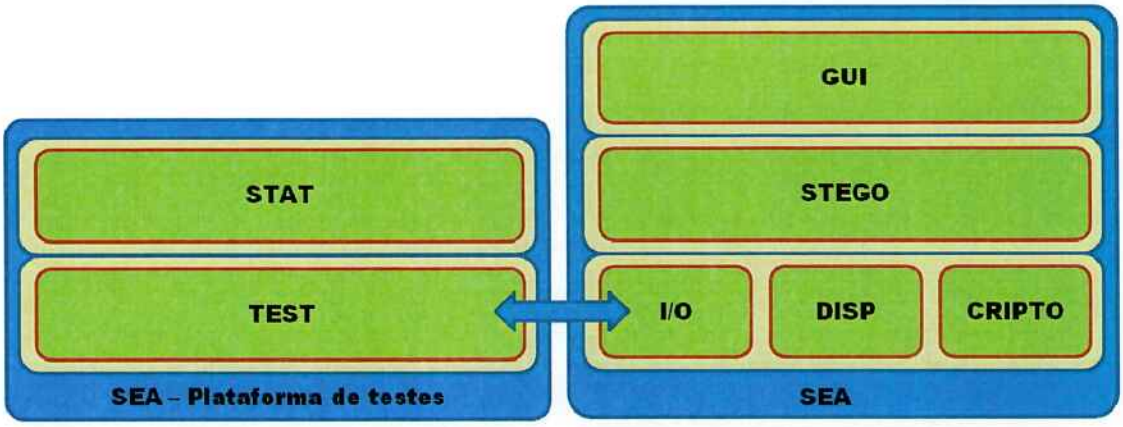


Figura 27 – Plataforma de testes utilizada no SEA

A detecção de colisões foi feita com uma busca no vetor de blocos dos que deveriam pertencer ao arquivo sendo testado. Uma colisão é identificada quando alguma das 3 informações armazenadas em um das posições verificadas do vetor de blocos foi alterada em relação aos dados gravados inicialmente nos blocos. Colisões fatais são contabilizadas apenas quando ocorrem colisões em mais de (N - M) posições pertencentes a um mesmo bloco-disp original.

Uma vez que existe uma grande quantidade de informações a serem armazenadas para cada um dos arquivos de teste, foi necessário aumentar o tamanho da memória disponível para o Java.

Todos os números gerados a partir do módulo *Test* são gravados em arquivos de texto. Cada um desses arquivos contém uma iteração dos testes realizados. Eles podem ser analisados individualmente ou, caso seja necessário, outras informações podem ser obtidas, através do módulo *Stat*. Esse módulo tem como entrada os resultados obtidos com o módulo *Test* e gera resultados seguindo o mesmo padrão de arquivo. Dados como médias de resultados de diversas iterações de desvios-padrão podem ser obtidos e analisados da mesma forma.

7.2. DESCRIÇÃO DOS TESTES

Todos os testes descritos a seguir foram executados utilizando-se um volume esteganográfico simulado de 2Mblocos-sto (2097152 blocos-sto). Considera-se esse

espaço suficiente dadas as características do sistema e os tamanhos dos arquivos simulados.

7.2.1. TESTE 1 – MESMO ARQUIVO

Nesse teste um arquivo de tamanho variável é gravado no espaço simulado. O objetivo desse teste é verificar qual valor de N que minimiza a colisão de um arquivo com ele mesmo, uma vez que quanto mais blocos são gravados, maior a probabilidade de um bloco de um arquivo colidir com um bloco desse mesmo arquivo.

Parâmetros de entrada:

- M
- Número de iterações a serem realizadas
- Quantidades de valores de N a serem testados
- Número de tamanhos de arquivos a serem testados, em números de blocos-disp
- Variação do tamanho do arquivo

Resultados obtidos a partir desse teste:

- Colisões – Blocos-sto substituídos x Tamanho do arquivo, para cada N
- Colisões fatais – Blocos-disp perdidos x Tamanho do arquivo, para cada N
- Número máximo de colisões por bloco-disp x Tamanho do arquivo, para cada N
- Média de colisões por bloco-disp x Tamanho do arquivo, para cada N

7.2.2. TESTE 2 – ENTRE ARQUIVOS

No teste entre arquivos, o objetivo é observar o tempo de vida de um arquivo no sistema. É gravado um arquivo de teste e guardados os seus blocos. Em seguida são realizadas gravações de outros arquivos, de acordo com os parâmetros de entrada, e calculadas as colisões no arquivo teste da mesma forma que a descrita anteriormente. Todos os arquivos têm o mesmo tamanho.

Parâmetros de entrada:

- M
- Número de iterações a serem realizadas
- Quantidades de valores de N a serem testados
- Quantidade de gravações a serem realizadas
- Passo da gravação
- Variação do tamanho do arquivo

Resultados obtidos a partir desse teste:

- Colisões – Blocos-sto substituídos x Gravações, para cada N
- Colisões fatais – Blocos-disp perdidos x Tamanho do arquivo, para cada N
- Número máximo de colisões por bloco-disp x Tamanho do arquivo, para cada N
- Média de colisões por bloco-disp x Tamanho do arquivo, para cada N

7.2.3. TESTE 3 – CAPACIDADE DO SISTEMA

O objetivo desse teste é avaliar a capacidade de armazenamento do volume esteganográfico para diversos tamanhos de arquivo e Ns. O número de colisões irá depender do valor de N e do número de blocos gravados, limitando a quantidade de arquivos íntegros no sistema até o momento que ocorre uma colisão fatal. Para realizar esse teste, para cada tamanho e N, foram gravados arquivos até que ocorresse pelo menos uma colisão fatal em algum dos já colocados. Nesse ponto, a

capacidade do sistema é considerada sua capacidade total, uma vez que uma colisão fatal foi encontrada.

Parâmetros de entrada:

- M
- Número de iterações a serem realizadas
- Quantidades de valores de N a serem testados
- Número de tamanhos de arquivos a serem testados, em números de blocos-disp
- Variação do tamanho do arquivo

Resultados obtidos a partir desse teste:

- Capacidade absoluta – Número de arquivos gravados x N, para cada tamanho de arquivo
- Capacidade relativa – % de blocos-sto ocupados x N, para cada tamanho de arquivo

7.3. RESULTADOS

Os arquivos resultantes de cada um dos testes realizados acima foram importados através do Microsoft Excel e os gráficos obtidos, assim como comentários a respeito dos resultados são os apresentados a seguir.

7.3.1. TESTE 1 – MESMO ARQUIVO

O primeiro objetivo desse teste é a avaliação de quantas colisões entre blocos do mesmo arquivo acontecem à medida que o tamanho desses arquivos aumenta. Essa análise é feita para cada um dos valores de N. Podemos notar que quanto maior o valor de N, maior o número de colisões. Isso é esperado, uma vez que o valor de N indica quantos blocos-sto extras serão gravados para garantir a

redundância de M blocos-raw do arquivo original. Quanto maior o número de gravações executadas, maior a chance de dois endereços aleatórios serem gerados e portanto, maior a chance de colisões.

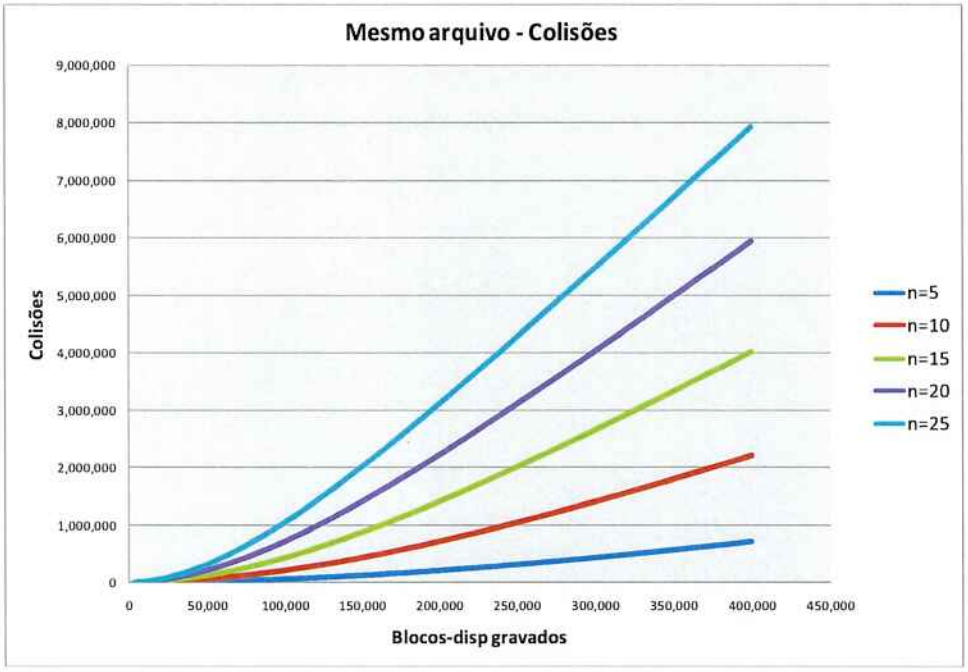


Figura 28 – Teste 1: Número de colisões entre blocos de um mesmo arquivo

Apesar do número grande de colisões ocorrendo para um número relativamente pequeno de gravações, a ocorrência de uma colisão dentro de um bloco-disp não significa que aquele bloco não poderá mais ser recuperado. Para que isso aconteça, é necessário que ocorram mais que (N – M) colisões dentro de um mesmo bloco-disp. Para tal, temos o gráficos da Figura 41. Uma versão desse gráfico com os resultados para valores acima de 160K pode ser encontrada no ANEXO A – RESULTADOS DE TESTES.

O primeiro ponto a ser notado nesses gráficos é o fato do menor valor de N não corresponder à curva com menor número de colisões fatais. Isso era de certa forma esperado, uma vez que apesar de aumentarem a redundância da informação, maiores valores de N correspondem também a um maior número de escritas no disco, o que causa mais colisões. Observe também que no caso de eliminação da redundância da informação (N = 5), qualquer colisão resulta em uma colisão fatal.

Nos gráficos podemos observar que as curvas com menor número de colisões fatais são as que têm N iguais a 10 e 15. Logicamente, não podemos afirmar que

esses N são ideais, pois uma colisão fatal inviabiliza a recuperação ideal do arquivo. No entanto, para efeito de configuração do sistema, sugere-se a escolha de N entre esses valores, uma vez que o SEA é configurado para recuperar todo o conteúdo ainda íntegro do arquivo, e um menor número de colisões fatais resulta portanto em mais conteúdo recuperado.

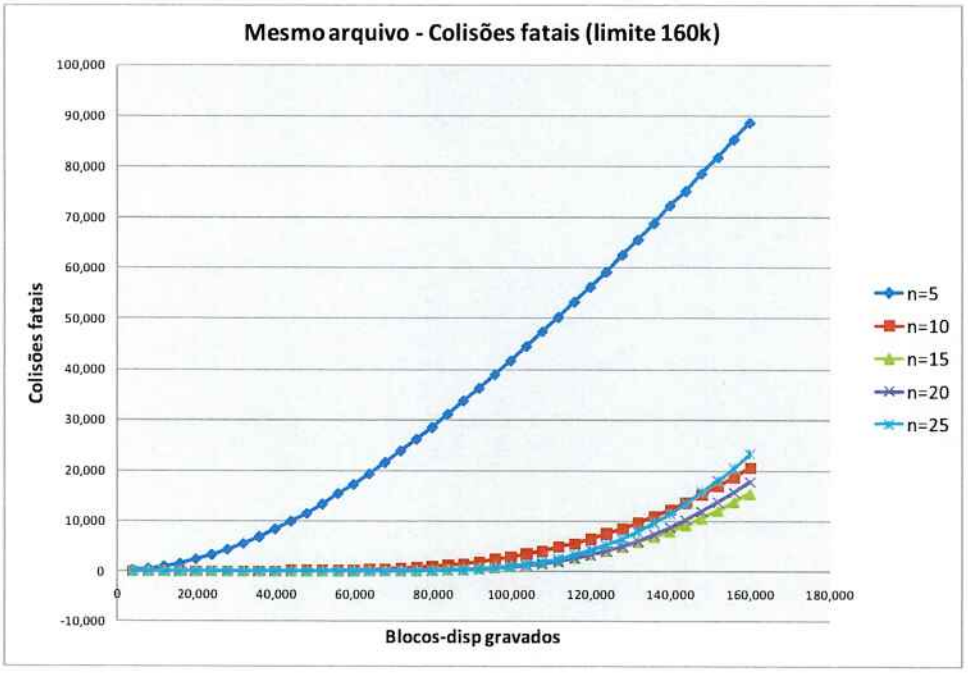


Figura 29 – Teste 1: Número de colisões fatais entre blocos de um mesmo arquivo (limite 160k blocos-disp)

Para um volume esteganográfico de 2Mblocos-sto (2GBytes), usado nos testes, não temos colisões fatais entre blocos para tamanhos de arquivo de até 80Kblocos-disp (~386MBytes). Esse tamanho corresponde a aproximadamente 20% do volume total do disco.

Dentre todos os bloco-disp do arquivo em cada tamanho, existe um cujo número de colisões internas foi máximo. O próximo gráfico indica qual é esse valor para cada um dos tamanhos de arquivo e valores de N. Logicamente, esse valor é limitado a N para cada caso, uma vez que N representa o número de blocos-sto dentro de cada bloco-disp.

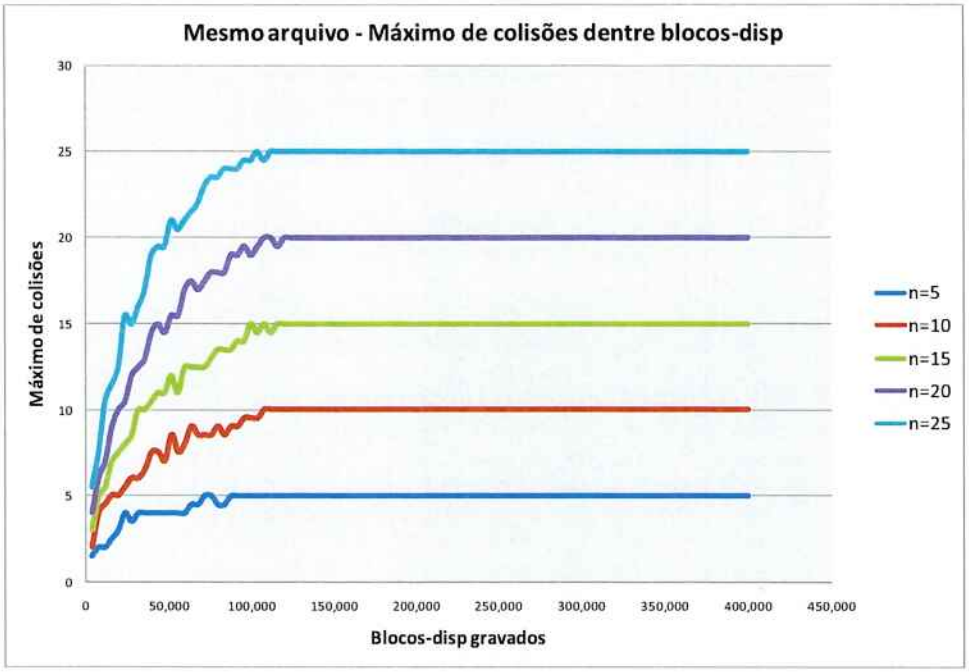


Figura 30 - Teste 1: Número máximo de colisões dentro de um mesmo bloco-disp

A média do número de colisões dentro de um mesmo bloco-disp, calculada dentro de um mesmo arquivo é apresentada no gráfico seguinte.

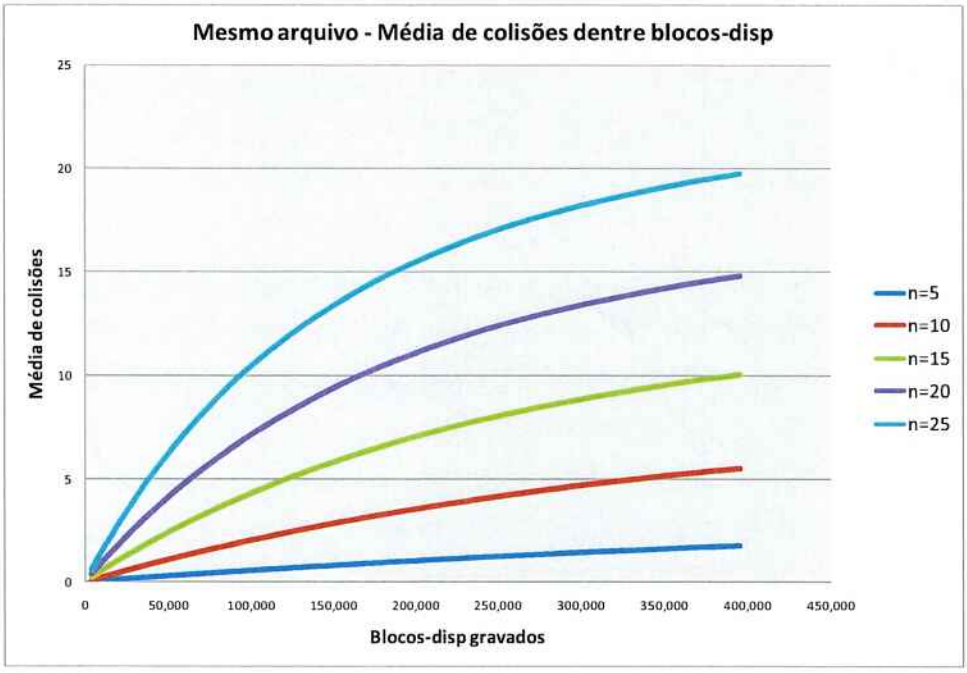


Figura 31 - Teste 1: Número médio de colisões dentro dos bloco-disp

7.3.2. TESTE 2 – ENTRE ARQUIVOS

O teste entre arquivos visa obter qual é a meia-vida de um arquivo no sistema. Esse conceito existe devido às colisões fatais que podem acontecer. Para obter os resultados desejados, é feita uma simulação de gravação de uma série de arquivos de 50 blocos-disp (250Kbytes). Os blocos-disp pertencentes a um arquivo de teste (com mesmo tamanho dos demais) é verificados por colisões e colisões fatais a medida que o novos arquivos eram gravados.

Observando o primeiro gráfico podemos notar resultados semelhantes aos obtidos para o teste 1. Quanto maior o valor de N, maior o número de blocos-sto gravados e portanto, maiores os valores de colisão.

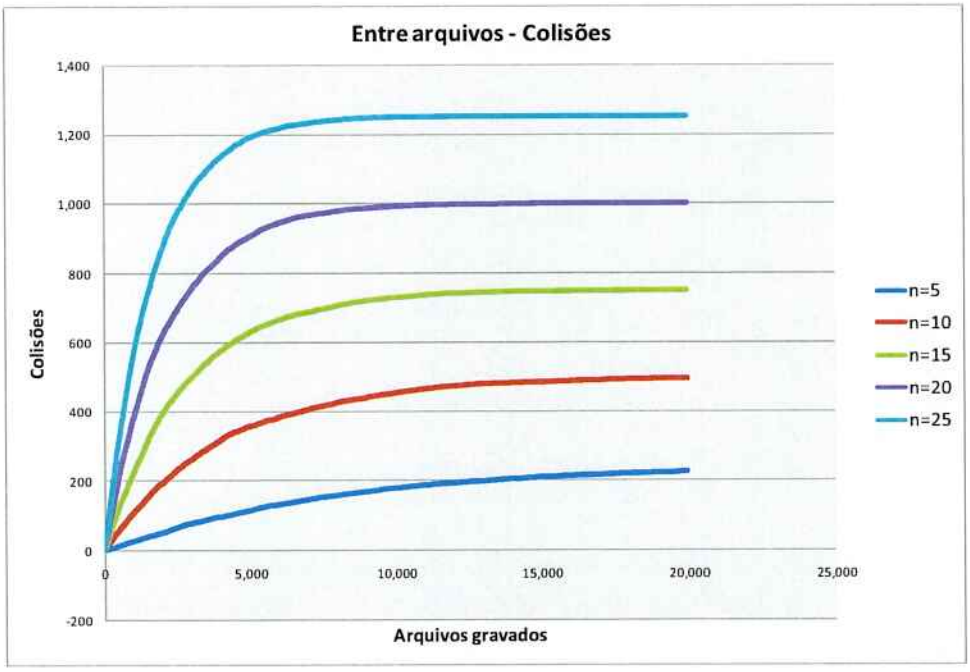


Figura 32 – Teste 2: Número de colisões no arquivo teste

Os gráficos relativos às colisões fatais indicam que um arquivo dentro do sistema consegue resistir sem problemas à gravação de até 1400 arquivos para N = 25, a opção a qual oferece a maior resiliência a novas gravações. No entanto, se o objetivo for obter o valor de N para o qual o sistema permite recuperar a maior quantidade de dados no caso de falhas, temos que N entre 15 e 20 são as melhores escolhas.

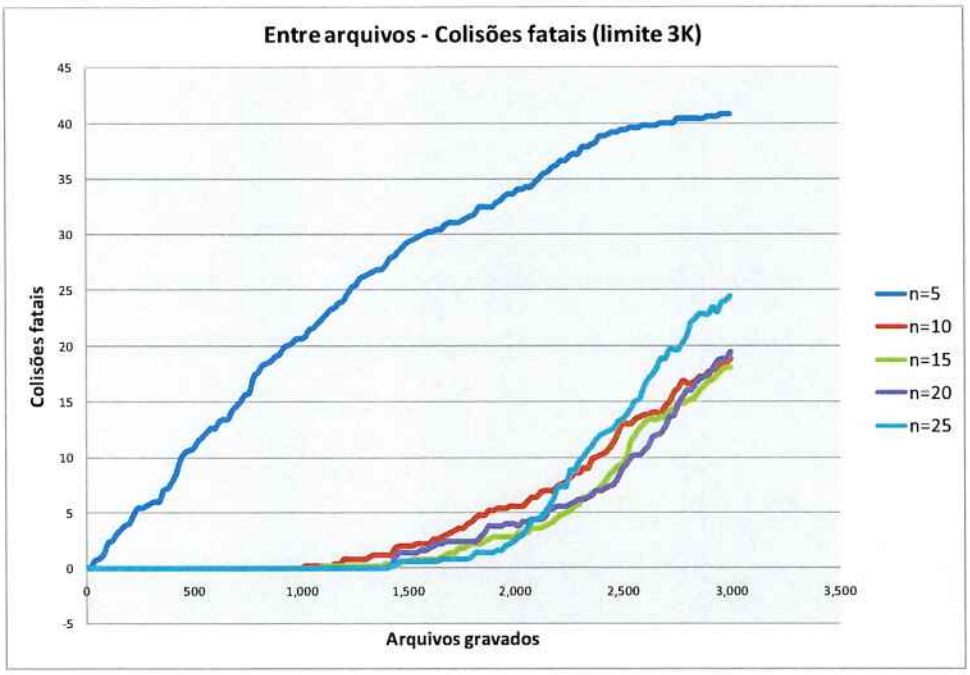


Figura 33 – Teste 2: Colisões fatais no arquivo teste (limite 3K arquivos gravados)

O número de arquivos gravados no sistema que causou a determinação da meia vida dos arquivos corresponde à 338MBytes. Esse valor é bem próximo do obtido com o teste 1, 386MBytes, correspondendo a 16% do total do volume esteganográfico. Esses valores começam a dar indicativos da capacidade efetiva do volume esteganográfico para o SEA.

Um gráfico correspondente ao anterior com limite até 25000 arquivos está disponível no ANEXO A – RESULTADOS DE TESTES. Lá também podem ser encontrados os gráficos relativos aos números máximos e médios de colisões por blocos-disp, os quais apresentam resultados semelhantes ao do Teste 1.

7.3.3. TESTE 3 – CAPACIDADE

Este teste tem como foco descobrir qual é a capacidade máxima do sistema, de forma a criar uma estimativa para o dimensionamento de sistemas reais. A capacidade do sistema é definida aqui como a porcentagem do espaço total disponível que podem ser ocupada sem que ocorram colisões fatais entre os

arquivos já armazenados. No sistema Mnemosyne (8), por exemplo, a capacidade máxima alcançada gira em torno de 50%.

Os gráficos a seguir mostram como varia a capacidade do sistema para diferentes valores de N, considerando apenas arquivos pequenos, de 50 blocos-disp (250Kbytes). Números de arquivos armazenados e capacidades relativas são apresentados.

Podemos observar que o pico de capacidade do sistema fica em torno de 14% do total de espaço do volume esteganográfico, o que equivale a 1200 arquivos. Esse valores estão de acordo com os obtidos no teste 2, e pequenas diferenças podem ser explicadas pela seqüência aleatória de posições utilizadas para o armazenamento em cada teste.

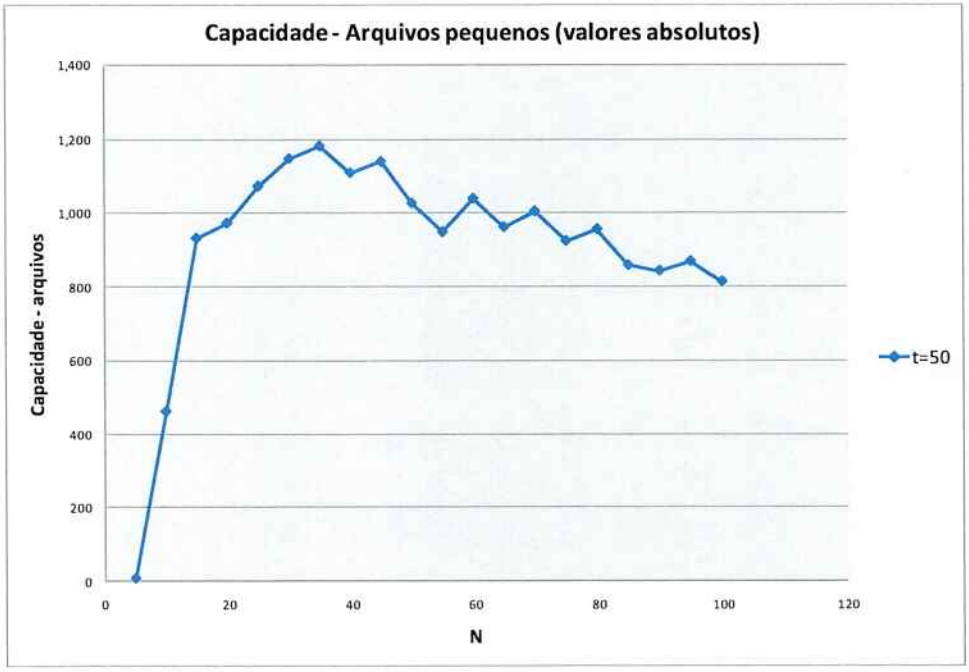


Figura 34 – Teste 3: Capacidade para arquivos pequenos

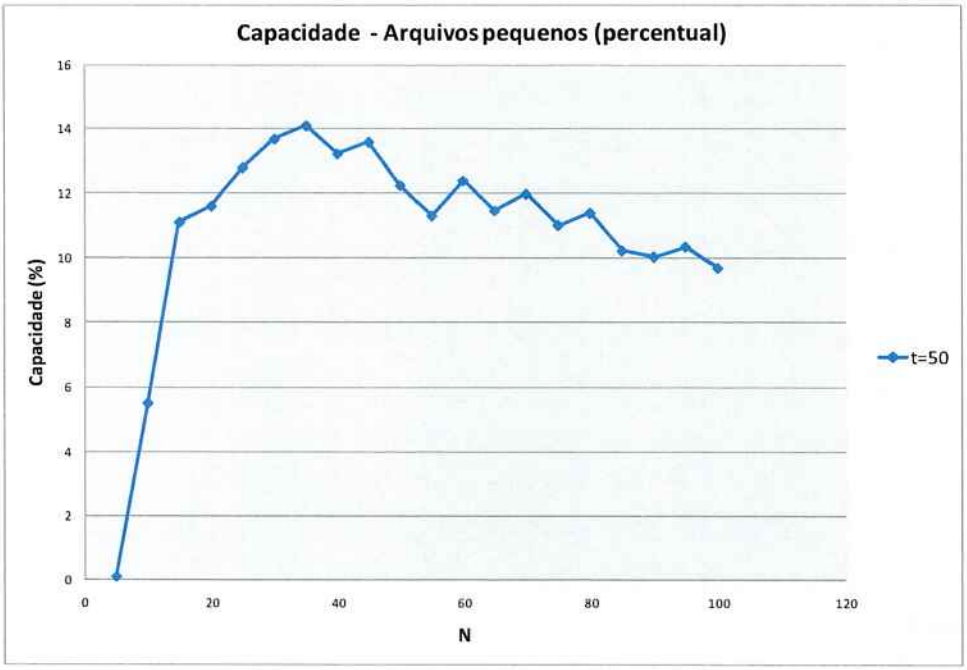


Figura 35 – Teste 3: Capacidade relativa para arquivos pequenos

A seguir é apresentada uma análise semelhante, com objetivo de a observar qual é o impacto da utilização de arquivos de tamanhos maiores na capacidade do sistema. Os resultados mostram que a capacidade do sistema se encontra na faixa de 12 a 15%.

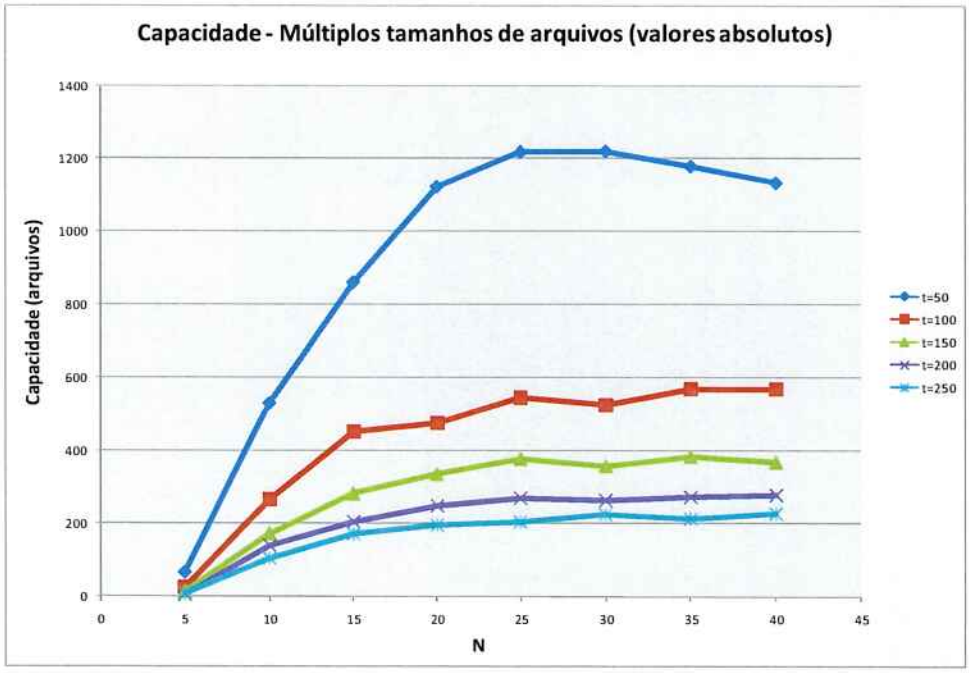


Figura 36 – Teste 3: Capacidade para arquivos de tamanhos diferentes

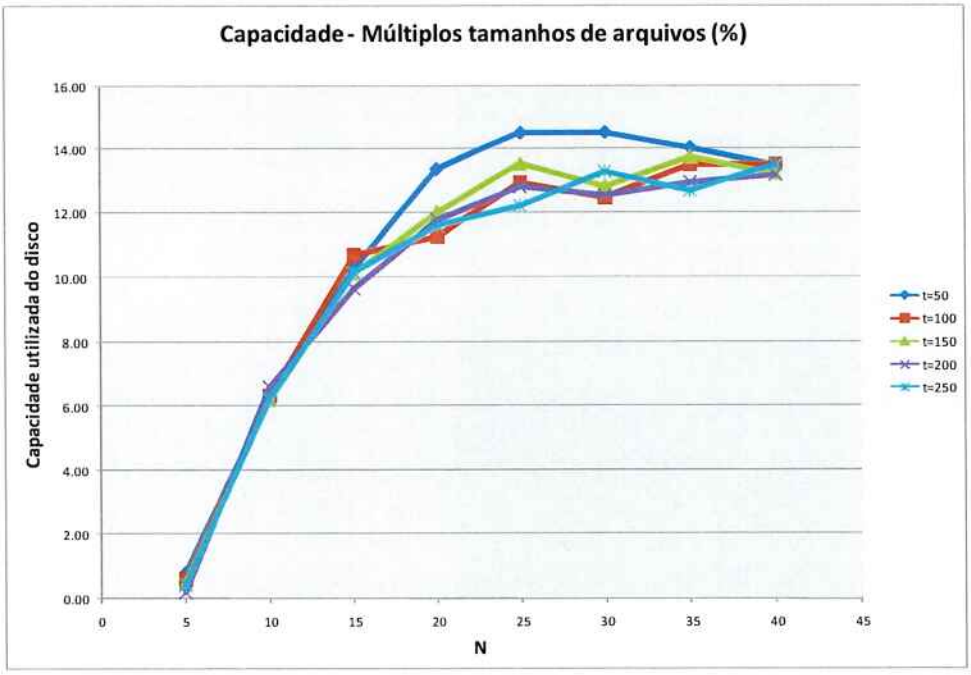


Figura 37 – Teste 3: Capacidade relativa para arquivos de tamanhos diferentes

Era esperado que arquivos de tamanho pequeno conseguissem otimizar o uso do espaço em disco, uma vez que possuem menos blocos e portanto menores chances de colisão, permitindo assim armazenar um maior número de arquivos sem que muitos sejam perdidos. Os resultados obtidos confirmam essa teoria.

Infelizmente o valor de capacidade obtido pelo Mnemosyne não é critério de comparação para o SEA, uma vez que não são apresentados no artigo daquele sistema os tamanhos dos arquivos utilizados. No entanto, sabe-se que o total do volume esteganográfico utilizado é de 4MBlocos-sto, e portanto, o dobro do tamanho utilizado no SEA. Acredita-se que com uma configuração mais adequada do sistema, através da criação de volumes maiores de dados,

8. APLICAÇÕES

A principal vantagem de um sistema esteganográfico como o SEA é a possibilidade dos usuários negarem, se a possibilidade de contestação, a existência de uma informação. Vamos com essa seção mostrar diversas situações onde o SEA pode ser utilizado para atender as necessidades de um usuário

8.1. ESFERA INDIVIDUAL

8.1.1. Exemplo 1

Uma família possui apenas um computador em casa. Esse computador é utilizado por todos os membros da família, que armazenam todos seus arquivos pessoais e de trabalhos. Dentre os filhos da família, existe um que possui conhecimento avançado sobre computação e normalmente é a pessoa que resolve todos os problemas relativos àquele computador.

Um dia, o pai e mãe decidem comprar, em segredo um apartamento na praia, que poderá ser aproveitado por toda a família, e que já é um desejo dos filhos durante muito tempo. A idéia é fazer uma surpresa durante o Natal. Durante meses os pais pesquisam opções na Internet e armazenam fotos de imóveis, listas de preços e contatos de imobiliárias. Ao invés de utilizar o armazenamento em uma pasta qualquer, eles decidem se precaver e armazenar os arquivos dentro de um volume esteganográfico.

Apesar dos filhos notarem o tempo diferente que os pais passam na frente do computador, e de desconfiarem que existe alguma coisa de diferente acontecendo, não podem afirmar que nada fora do usual está acontecendo, pois nem o filho com conhecimentos avançados sobre computadores consegue identificar a existência de qualquer novo arquivo no computador. Dado que o volume de informações armazenadas é muito pequeno perto do conteúdo total do disco, a diferença de espaço livre também passa desapercebida. Desconfiado sobre a nova aplicação que

os pais instalaram na máquina, o filho decide pedir informações sobre o seu funcionamento.

Durante a apresentação, os pais dizem uma senha especial para os filhos que revela apenas a existência de um diretório contendo alguns e-mails trocados com amigos de trabalho que contém algumas informações delicadas. Esse diretório foi especialmente criado dentro do mesmo volume esteganográfico que contém as informações da compra do apartamento, com o único objetivo de mascarar esse fato.

Esse é um exemplo típico de utilização de um sistema como o SEA para a segurança de informações privativas. O objetivo de um sistema como esse vai muito além da proteção contra pessoas maliciosas. Ele busca proteger a privacidade das informações dos usuários, independente se o conteúdo dessas informações é ou não algo que pode ser usado contra aquela pessoa.

8.1.2. Exemplo 2

Uma empresa de software acabou de criar um aplicativo muito complexo, cujo desenvolvimento custou milhões de dólares. Essa empresa, justamente por se tratar de uma empresa moderna, permite o uso de computadores e dispositivos pessoais dentro do ambiente de trabalho. Esses dispositivos passam por uma validação e análise de conformidade com alguns requisitos, de acordo as regras e certificações de mercado de entidades internacionais.

Um dos principais desenvolvedores, acostumado a trabalhar no seu notebook pessoal, decidiu levar uma versão beta da aplicação para sua casa. Essa aplicação, de acordo com análises de mercado feitas internamente, tem um grande potencial para desbancar todos os concorrentes e fazer a empresa entrar no mercado já como líder. Devido a isso, a existência do produto é mantida em sigilo absoluto.

Devido a esse detalhe, o desenvolvedor resolve se precaver, e cria um volume esteganográfico onde armazena todos os arquivos referentes àquela aplicação, não deixando nenhuma evidência do seu desenvolvimento fora desse volume. Como esse notebook é utilizado também para atividades pessoais, e por sua mulher, ele decide criar uma segunda estrutura de diretórios, para proteger alguns arquivos pessoais e e-mails de familiares.

Durante uma conferencia de desenvolvedores, esse funcionário deixa o notebook em uma mesa por alguns minutos e quando volta, o computador não está lá. Mais tarde, os funcionários do hotel dizem ter encontrado o notebook completamente destruído. Ao averiguar os danos, o desenvolvedor descobre que o disco rígido do notebook foi retirado. No entanto, ele sabe que os arquivos relativos à aplicação estão a salvo mesmo que os ladrões tenham capacidade de analisar todos os bits daquele disco, pois provavelmente não será possível nem detectar que existia alguma referência aquele projeto, devido à utilização da esteganografia.

Vale lembrar que a esteganografia, assim como a criptografia e qualquer outra ferramenta voltada à segurança não pode ser utilizada sem seguir certo número de regras. Da mesma forma que para proteger a confidencialidade de uma informação criptografada é necessária a utilização de chaves derivadas de senhas fortes, no caso da esteganografia, além de senhas fortes, é necessária a consciência de que a existência de informações fora do volume esteganográfico não vai ser protegida. Dessa forma é importante saber o que proteger e como proteger.

8.2. ESFERA ORGANIZACIONAL

8.2.1. Exemplo 1

Um datacenter, tentando atrair novos segmentos de mercado, decide oferecer serviços relativos à segurança para os seus clientes. Entre as opções disponíveis, está a disponibilização de servidores com criptografia de disco rígido, servidores dentro de redes seguras e a utilização de sistemas de arquivos esteganográficos.

Esta última é uma nova opção, adotada em caráter de teste de forma a avaliar a viabilidade de um sistema como esse. Notou-se que existe um segmento de mercado voltado a armazenamento de arquivos pequenos, mas altamente confidenciais, ao invés de grandes arquivos de dados.

Um dos grandes problemas do datacenter ao trabalhar com esse segmento de mercado é o receio que alguns de seus clientes decidam utilizar os serviços de segurança oferecidos como um banco suíço, armazenando todo tipo de arquivos

ilegais, resultados de espionagem industrial ou dados pessoais de funcionários que não deveriam estar sendo armazenados. Mais tarde, a empresa pode sofrer processos judiciais devido a sua incapacidade de lidar com esse tipo de problema.

Ao analisar as características do sistema esteganográfico de arquivos a ser utilizado, nota que ele é uma solução para esse tipo de problema. Uma vez que não existe forma de observar a existência de arquivos dentro de volume esteganográfico, quanto menos o conteúdo desses arquivos, fica muito difícil da empresa ser comprovadamente acusada de armazenar esses arquivos.

No pior dos casos, devido aos canais de auditoria disponíveis no sistema, é possível oferecer para qualquer entidade que necessite, dados relativos á utilização do disco esteganográfico. No entanto como não é possível distinguir que os dados armazenados são verdadeiros ou falsos.

Além disso, a administração do sistema é muito simples, uma vez que todo o cuidado relativo à manutenção da integridade dos arquivos é passado para a responsabilidade do cliente, uma vez que o datacenter não tem nenhum controle sobre o espaço ocupado nos discos rígidos do servidor ou sobre a existência dos dados. A única restrição – se é que pode ser chamada assim – a ser obedecida pelo datacenter é não disponibilizar ou armazenar nenhuma forma de backup dos volumes esteganográficos, uma vez que os resultados de uma análise da diferença entre duas versões do backup podem revelar as posições onde o usuário gravou seus dados. Esse tipo de informação é exatamente a obtida através do canal de auditoria, no entanto, de uma forma muito mais controlada.

Os sistemas esteganográficos apresentam uma série de vantagens para empresas como datacentersd ou mesmo corporações que gostariam de oferecer espaços de armazenamento para seus funcionários. Uma série de problemas que existem com os formatos de armazenamento atual são eliminados pelo fato da entidade que possui a infra-estrutura de hospedagem não ter conhecimento sobre o quanto do espaço físico está ocupado.

9. EXTENSÕES

Durante o projeto do SEA surgiram algumas idéias focadas em melhorar as funcionalidades oferecidas pelo sistema, mas que fugiam do escopo inicial do projeto e não puderam ser implementadas. No entanto, algumas das características do sistema atual foram projetadas de forma que possíveis extensões do projeto fossem facilitadas, o que tornou o SEA um sistema relativamente flexível, com diversas aberturas para novas funcionalidades. O objetivo dessa seção é descrever as duas principais extensões possíveis e apresentar os motivos pelos quais essas seriam interessantes.

9.1. SISTEMA DISTRIBUÍDO

O SEA é um sistema esteganográfico baseado em grande parte nos sistemas desenvolvido por (7) e na versão local do sistema apresentado por (8). O último chamado Mnemosyne, é um sistema esteganográfico distribuído que utiliza a rede Tapestry(13) como base para o armazenamento dos dados.

Seria interessante no futuro transformar o SEA de forma que ele se torne um sistema desse tipo. Entre os motivos que tornam um sistema esteganográfico distribuído de arquivos interessante estão a facilidade de manutenção, a segurança de alocar arquivos confidenciais em sistemas externos, protegidos por um arquitetura de segurança mais robusta e a criação de mais uma oferta a ser oferecida por centros de armazenamento de dados.

Cogitou-se a utilização da rede OpenDHT para essa extensão. Essa rede é composta por diversos nós interligados os quais, através de um cliente de acesso, permitem que sejam armazenados dados em tabelas indexadas através de *hashes*. A estrutura da rede recebe a requisição de cada cliente e se preocupa em descobrir em qual dos nós está objetivo. Nela existem duas operações básicas:

- **put(key,data)** – adiciona uma nova informação associada a chave especificada.

- **get(key)** – recupera a informação associada à chave especificada.

Durante o projeto do SEA pensava-se em deixar a transição de um sistema local para um sistema distribuído bem simples. Esse foi um dos motivos da criação do módulo I/O. Esse módulo realiza basicamente as duas operações acima, só que o destino final da versão atual é o volume esteganográfico em uso no momento. Sugere-se que, para a utilização da OpenDHT esse módulo seja substituído por um cliente dessa rede, e passe a gerenciar as operações de put() e get().

Além dessa tarefa, o novo módulo de I/O deve ser responsável pelo gerenciamento das chaves utilizadas por aquele volume esteganográfico, da mesma forma que atualmente é responsável pelo gerenciamento das posições dentro do volume onde são gravadas as informações do usuário.

A forma com que se dá a inicialização do volume esteganográfico também deve ser alterada. Uma vez que não existem limitações para o número de chaves usadas por um cliente da OpenDHT, seria necessário criar um limite virtual. Para tal, por exemplo, poderiam ser escolhidas chaves seqüenciais de forma que exista um mapeamento entre essas chaves e posições em um volume esteganográfico atual.

Outro ponto a ser observado é o fato de que a OpenDHT não realiza substituição automática do conteúdo de uma chave quando uma nova requisição de put() é feita sobre essa chave. Os novos dados são simplesmente adicionados a uma lista de resultados relativos àquela chave. Assim, uma operação de get() pode retornar mais de um resultado. Logicamente existe o aspecto positivo de esse esquema evitar colisões entre arquivos. No entanto esse fato possui uma implicação muito pior dentro do contexto do SEA, uma vez que analisando quais chaves possuem mais de uma entrada seria possível determinar onde existe e onde não existem dados do usuário. Felizmente, a rede OpenDHT permite a remoção de conteúdo associado a uma chave e portanto, é possível remover os dados já existe antes de adicionar uma nova informação. Observe que essa função não precisa ser necessariamente executada pelo novo módulo I/O.

Existe um último ponto interessante de discussão. Devido a utilização de uma rede distribuída, o sistema fica sujeito a ataques de análise de trafego entre os nós da rede DHT e cada um dos clientes. Para solucionar esse problema, pode ser estabelecido que clientes podem ler e escrever dados randômicos para o volume

esteganográfico distribuído. Dessa forma, não é possível com a análise de trafego, distinguir que blocos de dados são reais e quais são informações aleatórias.

A rede OpenDHT, como o nome sugere, é aberta a qualquer pessoa. Dada que essa é uma aplicação de segurança e privacidade, é desejável que exista algum controle de acesso de usuários. Para tal, sugere-se a criação de uma rede separada da atualmente em funcionamento no PlanetLAB, além da transformação do SEA atual em uma aplicação cliente, parte de uma estrutura composta por mais duas outras aplicações, como se segue:

- **Aplicação de Administração e Autenticação (AAA):** A AAA seria responsável pela configuração do funcionamento geral do sistema, centralizando as operações de gestão e controle. Fará por exemplo o controle do cadastro de usuários, e será a interface pela qual os administradores manipularão o sistema.

- **Aplicação de Transferência (AT):** A AT seria responsável por controlar o acesso aos nós da rede DHT fazendo a autenticação dos usuários e limitando o uso da DHT de acordo com as configurações definidas na AAA. A AT funcionará como uma camada de acesso adicional sobre a rede DHT, pois esta não possui as funções de controle de acesso.

- **Aplicação Cliente (AC):** A AC seria uma extensão da versão local do projeto. Ela é a interface com a qual o usuário final interage com o sistema, e apresenta algumas funcionalidades a mais que a permitem dialogar com as demais aplicações para permitir o acesso ao sistema de arquivos distribuídos.

Um diagrama da comunicação entre essas entidades está na figura a seguir. Como pode ser observado, a AT funciona como um firewall instalado na frente de cada nó da rede DHT. A AAA é um elemento de confiança na rede. Ela foi criada especialmente com esse objetivo. Dado os requisitos de segurança do sistema não pode ser admissível que ataques como personificação de uma AT ou de uma AC sejam viáveis. Assim sugere-se que uma das funções da AAA seja o funcionamento como uma autoridade certificadora, que emite certificados para os demais elementos.

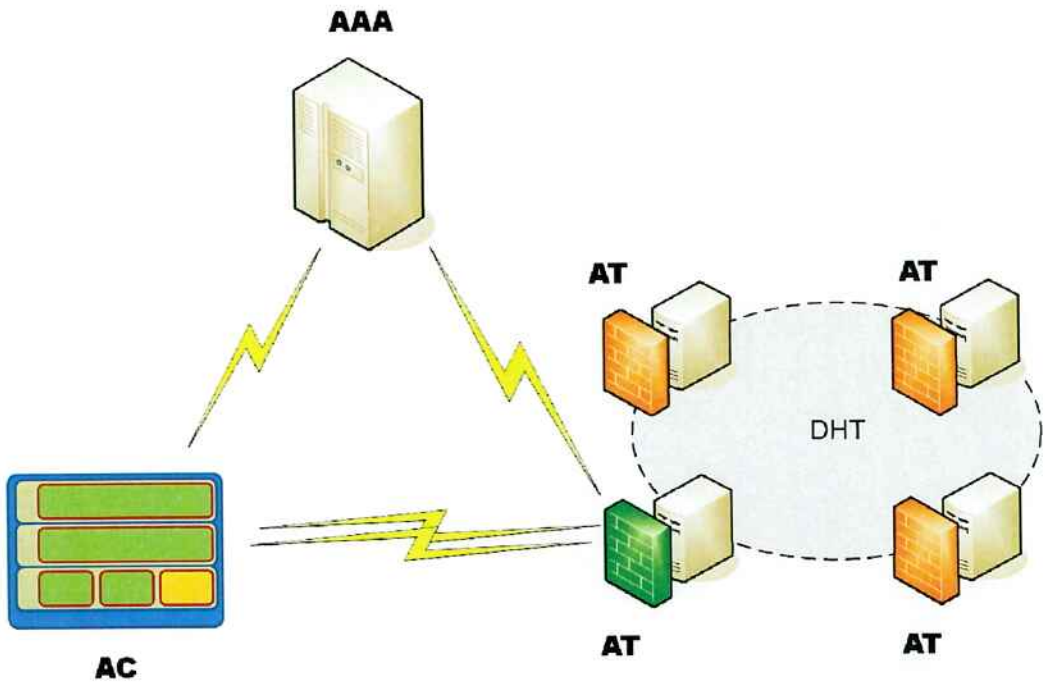


Figura 38 – Arquitetura básica de uma futura versão distribuída do SEA

Umas das aplicações desse tipo de sistema é o oferecimento de serviços para pessoas ou empresas que desejam manter seus dados privativos, tanto para terceiros como para a própria empresa os armazenando. Dessa forma, outro ponto de interesse na versão distribuída do sistema é como seria realizada a cobrança por uso. Uma vez que esse tipo de sistema seria oferecido como um serviço, é importante prever formas de cobrança e de atendimento de níveis de serviços (SLA – Service Level Agreement). Em (14) são oferecidas algumas sugestões e análises, com foco no sistema Mnemosyne. No entanto, dadas as características do SEA, o sistema de cobrança provavelmente será diferente, possuindo alguma associação com o nó primário da rede DHT onde o usuário realiza seu registro, por exemplo. Esse não é um problema simples, devido ao fato que a aplicação AC pode se conectar a qualquer um dos nós da rede.

Acredita-se que os requisitos a seguir sumarizam todas as funcionalidades necessárias em uma versão distribuída do SEA, incluindo as observações feitas até o momento.

REQUISITOS NÃO-FUNCIONAIS

1. AAA – Gerenciamento de usuários

A AAA irá apresentar um cadastro básico de usuários que tem a permissão de utilizar o serviço. Este cadastro deve informar o limite de uso de cada usuário. Com estas informações são gerados certificados, que são enviados aos clientes.

Cada certificado possui uma validade, e ao expirar o usuário perde a capacidade de acessar o sistema. Para que isto não ocorra um novo certificado deve ser gerado e enviado ao usuário.

2. AC, AT – Autenticação por certificado digital

Uma entidade certificadora de confiança gerará um certificado para cada cliente do sistema, que ficará armazenada no computador do usuário acessível pela AC.Ao conectar-se, a AT checará as credenciais da AC através deste certificado.

3. AC, AAA, AT – Limite de utilização e espaço disponível

Para garantir a possibilidade de recuperação de dados nesse ambiente multiusuário e distribuído, é necessário que o espaço total disponível para todos os clientes seja significativamente maior que o total de dados que cada usuário pode inserir na rede em um determinado período de tempo. A proporção ideal entre a quantidade de dados e o espaço disponível será objeto de estudo na fase de testes do sistema, não havendo no momento um valor definido.

O limite de transferência de cada usuário estará registrado em seu respectivo certificado, e as AT farão o controle das escritas a partir desta informação.

4. AC – Organização dos arquivos

Será fornecida dentro da AC uma forma para o cliente criar uma estrutura de arquivos, com diretórios, a fim de organizar os seus dados.

5. AC – Método de renovação

A AC realizará periodicamente a renovação dos dados gravados no sistema de arquivos para aumentar a probabilidade do arquivo não estar danificado quando for preciso recuperá-lo. O período entre cada renovação é definido pelo próprio usuário.

6. AAA, AT – Cobrança

Como cada usuário tem um limite máximo de utilização por nó, e a cobrança é feita baseada nesse valor. O sistema ira impedir o uso indevido e o valor cobrado é fixo mesmo se o usuário não utiliza o limite disponível completamente.

7. AAA – Envio de credenciais

A AAA é responsável pelo envio de todas as credenciais geradas para as respectivas ACs e ATs do sistema.

8. AC - Proteção contra sobrescrita

O sistema deve garantir que toda informação possa ser recuperada mesmo que uma pequena quantidade de blocos tenha se perdido. Os blocos podem ser perdidos por sobrescrita causada por outro cliente ou por outro arquivo do mesmo cliente. Para isso deve ser utilizada redundância dos dados, de maneira que mesmo com a perda de alguns blocos ainda seja possível recuperar o arquivo integralmente. Ainda, deve ser utilizado um mecanismo de renovação periódica, no qual de tempos em tempos o cliente lê os blocos corretos remanescentes e grava o arquivo novamente no espaço esteganográfico, restaurando a redundância perdida com o tempo.

9. AT – Remoção antes da inserção

A aplicação AT será responsável por sempre e sem exceção realizar a remoção de um conjunto de dados relacionado a uma chave da rede DHT antes da inserção de novos dados. Isso irá evitar que chaves duplicadas sejam armazenadas e indiquem a localização dos dados.

10. AAA, AT – Refutabilidade dos dados dos clientes

Não deve ser possível para ninguém exceto o proprietário dos dados verificar o conteúdo, o tamanho ou a própria existência dos seus arquivos armazenados. Também dado conjunto de dados armazenados, não deve ser possível verificar a quem pertence a informação gravada, ou mesmo diferenciar dados de clientes de um conjunto de bytes aleatórios.

Para que isto seja possível, o espaço esteganográfico precisa estar inicialmente preenchido em sua totalidade com bytes aleatórios.

Além disso, para que não seja possível identificar o proprietário dos blocos, as AC's devem, durante a operação de escrita, escrever também blocos de bytes aleatórios espalhados entre os blocos de dados reais. Desta maneira não é possível que um indivíduo que intercepte as comunicações entre a AC e a AT possa diferenciar dados reais de dados aleatórios.

11. AC - Proteção em profundidade

Mesmo que um atacante tenha acesso físico a todas as máquinas da rede DHT e ao conjunto de chaves onde a informação esta armazenada, os dados estão criptografados.

A criptografia será feita em dois níveis: no primeiro, o arquivo é criptografado integralmente. A partir dos dados criptografados, é criado um novo arquivo contendo redundâncias, que será dividido em blocos de tamanho definido pelo sistema. No segundo nível, cada bloco é então criptografado novamente antes de ser enviado.

REQUISITOS NÃO-FUNCIONAIS

1. AAA, AT - Canal de auditoria

As ATs e AAA deverão apresentar formas de, através de uma escolha dos administradores e certas checagens a serem especificadas posteriormente, registrar as informações necessárias para verificar se os limites de acesso estão sendo

aplicados corretamente, e em caso de falha do controle de acesso ser possível descobrir quando, quem e o que causou a falha.

2. AC, AAA, AT – Cálculos de variáveis de utilização do sistema

Todos os cálculos do sistema devem assumir que o limite máximo de usuários para aquele sistema de armazenamento de arquivos já foi atingido. O espaço total disponível, quantidade máxima de utilização de cada usuário e taxa de redundância devem ser calculados de forma a permitir uma utilização adequada do sistema no pior caso (ou seja, com o máximo de usuários).

3. AAA, AT, AC – Interface Gráfica

O sistema deve prover uma interface gráfica para a execução das funções de responsabilidade dos clientes ou administradores.

4. AAA, AT – Aleatoriedade dos dados

Os dados da unidade lógica devem apresentar distribuição aleatória com ou sem a presença de arquivos.

5. AAA, AT, AC – Estrutura de dados

Somente o cliente poderá conhecer a estrutura de dados armazenada (nomes de arquivos, organização da árvore de diretórios etc.). Nenhuma informação de como e quais arquivos estão armazenados estará disponível a qualquer entidade exceto o proprietário dos dados.

9.2. SEPARAÇÃO DO MODELO DE ARMAZENAMENTO DE DADOS

Um ponto de interesse futuro no sistema é a separação entre o modelo de armazenamento de dados utilizado e o controle sobre os volumes esteganográficos.

Conforme explicitado na seção 5.4 - MODELO DE ARMAZENAMENTO DE DADOS, a atual implementação assume um esquema fixo para que os usuários criem a sua estrutura de diretórios.

A proposta envolve um trabalho de separação de funções do SEA e a criação de novas interfaces de acesso ao volume esteganográfico através de modificações estruturais do módulo Stego, como pode ser observado nas figuras a seguir.

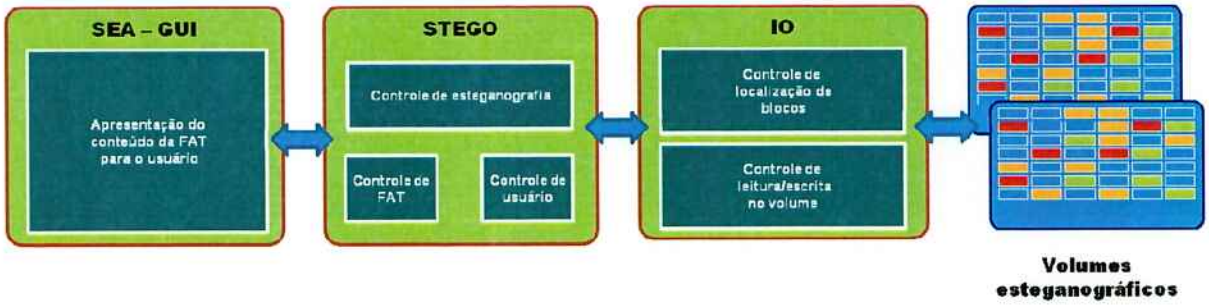


Figura 39 – Esquema geral de controle dos volumes esteganográficos atual

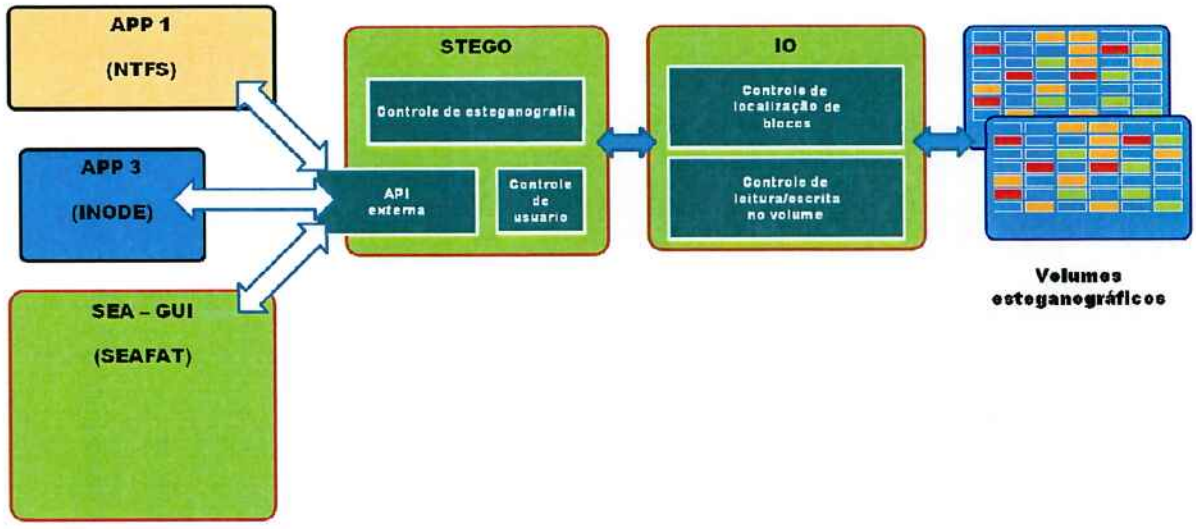


Figura 40 – Esquema proposto para separação de funções

Essa modificação irá permitir uma maior liberdade para diversas aplicações utilizarem as vantagens da esteganografia através do SEA. Em linhas gerais o objetivo é criar uma arquitetura parecida com a que existem hoje em dia nos computadores: qualquer sistema operacional pode ser instalado no disco rígido, juntamente com suas aplicações. Cada sistema operacional decide qual é o melhor método de organizar os seus arquivos no espaço físico, e cuida de garantir a integridade desses dados.

Logicamente, no caso do SEA não estaríamos falando de sistemas operacionais, mas sim de aplicações que desejem armazenar os seus dados de formas diferentes da adotada na implementação atual. Por exemplo, pode-se imaginar uma aplicação que não deseja a existência de uma estrutura de diretórios e sim apenas arquivos espalhados pelo disco, ou até aplicações que realmente desejem trabalhar com estruturas já conhecidas (INODE, NTFS).

Existem dois obstáculos principais a serem vencidos nessa estrutura.

O primeiro diz respeito à criação das interfaces disponíveis para que as aplicações realizem suas ações dentro do volume esteganográfico. Esse padrão de comunicação deve ser claro o suficiente para oferecer toda a gama de serviços que aplicações de segurança possam desejar. Em especial, esse problema envolve as mudanças necessárias dentro do SEA para permitir que as aplicações consigam recuperar dados específicos sem ter conhecimento das posições onde esses dados foram armazenados dentro do volume esteganográfico.

O outro obstáculo está relacionado com a necessidade de criação, por parte das aplicações que utilizarão os serviços de esteganografia do SEA, o controle sobre qual serão as requisições feitas ao módulo Stego para obter ou gravar os dados desejados. A dificuldade está no fato que essas aplicações terão de, baseadas na interface de acesso oferecida pelo SEA, conseguir criar requisições que resultem nos dados desejados sem o conhecimento da posição onde eles estão armazenados. Uma idéia é a utilização de identificadores para cada stream de informação armazenada, ao invés de para cada arquivo, como é feito no SEA atualmente.

10. CONCLUSÃO

O SEA é um sistema que cria, com sucesso, uma nova camada de segurança para os seus usuários. Ele vem atender a uma necessidade básica de todos as pessoas que utilizam computadores atualmente: maior privacidade. Privacidade aqui diz respeito a qualquer informação do usuário, independente se esse usuário tem algo a esconder ou não.

A principal qualidade do sistema é justamente essa: qualquer um pode guardar no mesmo lugar, com um alto nível de segurança as informações sobre a próxima viagem da sua família, as informações sobre o novo projeto secreto da sua empresa e os seus últimos e-mails. Cada um desses conjuntos de informações é protegido não só pela esteganografia implementada pelo sistema, mas também pela própria existência dos demais. Caso o usuário se sinta ameaçado a ponto de ter de afirmar que possui dados no disco, ele pode simplesmente revelar a informação de menor importância, mantendo a confidencialidade das demais.

Os testes realizados sobre o protótipo criado mostraram que o sistema é viável do ponto de vista técnico. No entanto, ainda existe espaço para melhorias significativas no que diz respeito à performance, a qual não foi foco principal da implementação. O protótipo criado também mostrou que é possível utilizar uma técnica complicada como a esteganografia de forma bem transparente para o usuário, e até mais simples que programas que utilizam somente criptografia.

O índice máximo de aproveitamento do disco rígido foi baixo se comparado à trabalhos anteriores, no entanto acredita-se que isso é devido basicamente à dificuldade de realização de testes com espaços maiores dos volumes esteganográficos. Também se espera que outros algoritmos de redundância possam ser implementados, melhorando a resiliência do sistema a colisões. Vale a pena mencionar que o custo por espaço vem caindo há décadas e o fator espaço tende assim se tornar um problema menor no futuro.

As aplicações do sistema são vantajosas tanto para empresas de hospedagem de arquivos como para usuários domésticos, o que cria um forte potencial para aproveitamento comercial de produtos desse tipo.

11. REFERÊNCIAS

1. REEDS, JIM. **Solved: The Ciphers in Book III of Trithemius's Steganographia**. Cryptologia. Outubro 1998.
2. IDA, GAMALL WEDNESDAY. **Hiding Glyph: Bytewise Image Steganography**. Gamall Wednesday Ida's Forum. Disponível em: <http://www.gamall-ida.com/f/viewtopic.php?f=3&t=192>. Acessado em 16 nov. 2007.
3. HAMLIM, SEAN. **Revelation – Steganography made easy**. Disponível em < http://revelation.atspace.biz/steganography.html>. Acessado em 16 nov. 2007.
4. COLE, ERIC. **Hiding in Plain Sight: Steganography and the Art of Covert Communication**. Indianapolis, Indiana – USA: Wiley Publishing Inc., 2003.
5. PETITCOLAS, FABIEN A. P. **The image downgrading problem.** Disponível em <http://petitcolas.net/fabien/steganography/image_downgrading/index.html>. Acessado em 16 nov. 2007.
6. PETITCOLAS, FABIEN A. P. **MP3Stego**. Disponível em < http://www.petitcolas.net/fabien/steganography/mp3stego/index.html>. Acessado em 16 nov. 2007.
7. ANDERSON, ROSS; NEEDHAN, ROGER; SHAMIR, ADI. **The Steganographic File System**. In: IWIH: International Workshop on Information Hiding. 1998.
8. HAND, STEVEN; ROSCOE, TIMOTHY. **Mnemosyne: Peer-to-peer Steganographic Storage**. In: Proceedings of the 1st International Workshop on Peer-to-peer Systems (IPTPS'02). Boston, MA – USA: 2002.
9. RABIN, MICHAEL O. Efficient dispersal of information for security, load balancing and fault tolerance. **Journal of the ACM**, p.335, abr. 1989.
10. GAMMA, ERIC; HELM, RICHARD; JOHNSON, RALPH; VLISSIDES, JOHN M. Design Patterns: Elements of reusable Object-Oriented Software. Addison-Wesley, 2000.
11. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Secure Hash Signature Standard**. Federal Information Processing Standards Publication 180-2. 01 ago. 2002.
12. KNUTH, DONALD. **The Art of Computer Programming**. Vol. 3. MA – USA: Addison-Wesley, 1998.
13. ZHAO, B. Y.; KUBIATOWICZ, J. D.; JOSEPH, A. D. **Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing**. Berkeley – CA – USA : 2000.
14. ROSCOE, TIMOTHY; HAND, STEVEN. **Transaction-based Charging in Mnemosyne: a Peer-to-peer Steganographic Storage System**. In: International Workshop on Peer-to-peer Computing at Networking. Pisa, Italia: 2002.
15. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Advanced Ecnryption Standard (AES).** Federal Information Processing Standards Publication 197. 26 nov. 2001.

12. ANEXOS

12.1. ANEXO A – RESULTADOS DE TESTES

TESTE 1 – MESMO ARQUIVO

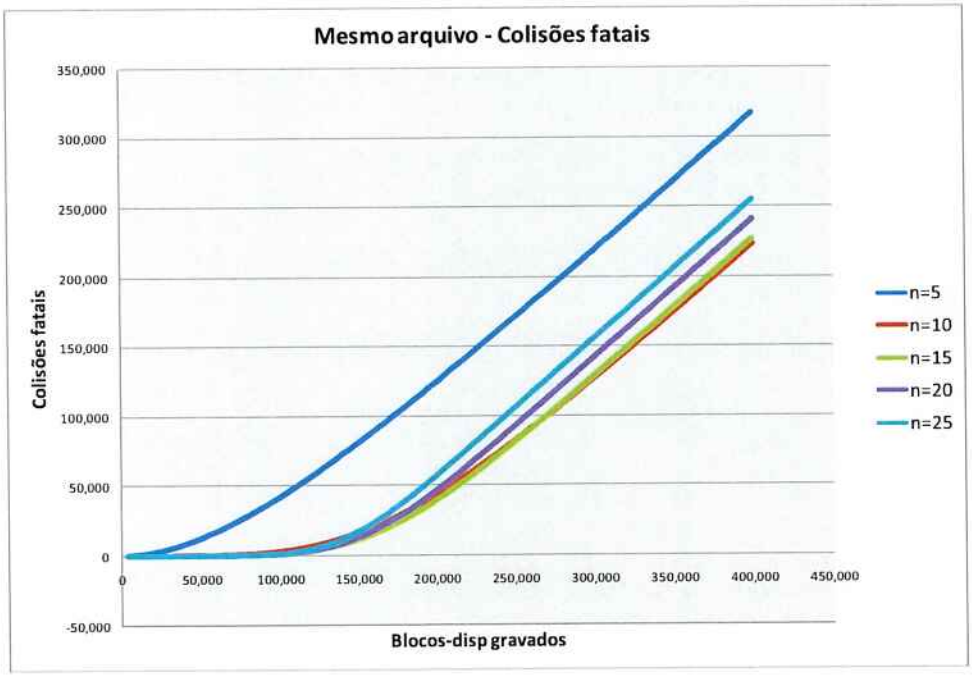


Figura 41 – Teste 1: Número de colisões fatais entre blocos de um mesmo arquivo

TESTE 2 – ENTRE ARQUIVOS

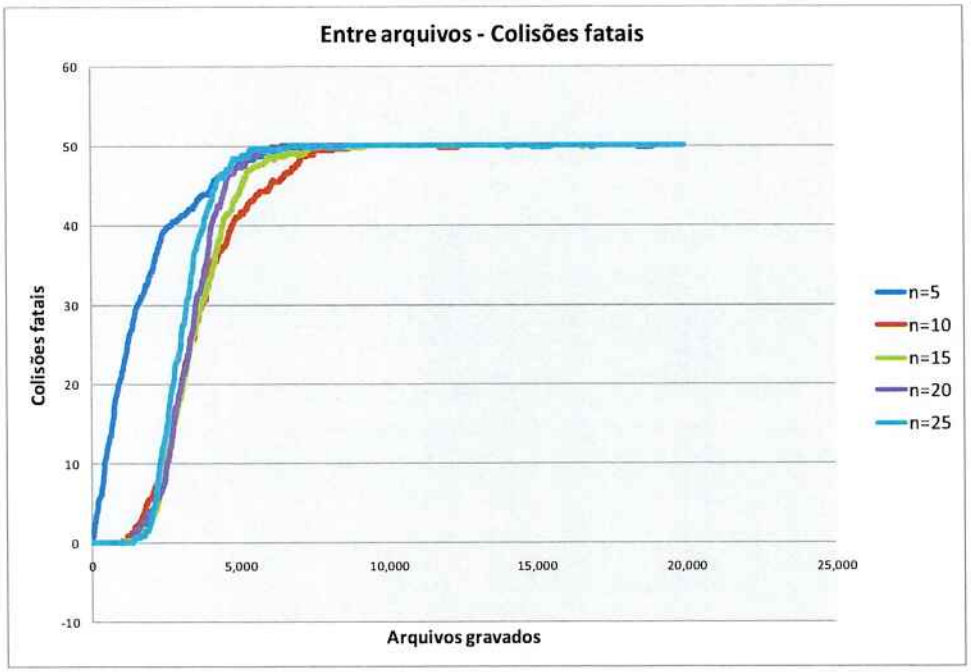


Figura 42 – Teste 2: Colisões fatais no arquivo teste

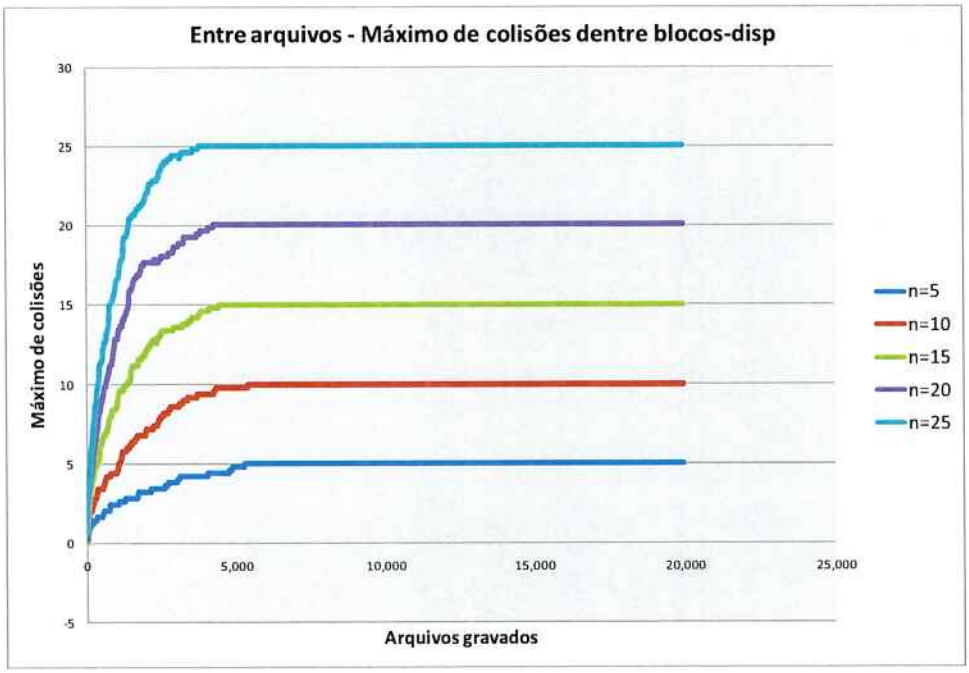


Figura 43 – Teste 2: Número máximo de colisões dentro de um mesmo bloco-disp

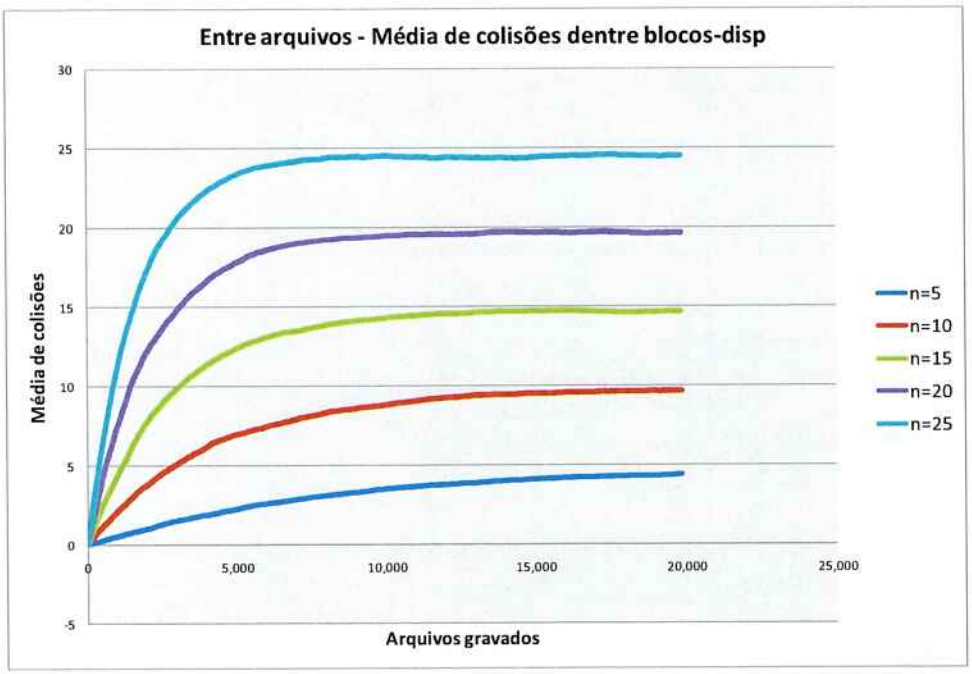


Figura 44 – Teste 2: Número médio de colisões dentro de um mesmo bloco-disp

12.2. ANEXO B – MANUAL DO USUÁRIO

O SEA é um sistema para armazenamento de arquivos de forma esteganográfica. Os arquivos são armazenados em um volume esteganográfico. Existem três operações com arquivos no SEA: Gravação, Recuperação e Renovação. As duas primeiras são operações convencionais de escrita e leitura no SEA. A Renovação é utilizada para aumentar a vida útil do arquivo no sistema. A contagem de argumentos neste manual será feita a partir do nome do arquivo Jar, uma vez que todos os comandos são precedidos pela execução da máquina virtual Java.

Configuração do volume esteganográfico

Para dar início à utilização do sistema, é necessário que seja criado um volume esteganográfico, acompanhado do seu arquivo de configuração. Esta tarefa pode ser realizada através do aplicativo seainit. Neste manual, faremos a criação de um volume de 2GB de tamanho, que será armazenado fisicamente no arquivo blocos.txt. Execute o comando a seguir para configurar o volume, sendo o primeiro argumento a localização do arquivo e o segundo a quantidade de blocos a ser criada (cada bloco pode armazenar 1KB).

java –jar seainit.jar blocos.txt 2097152

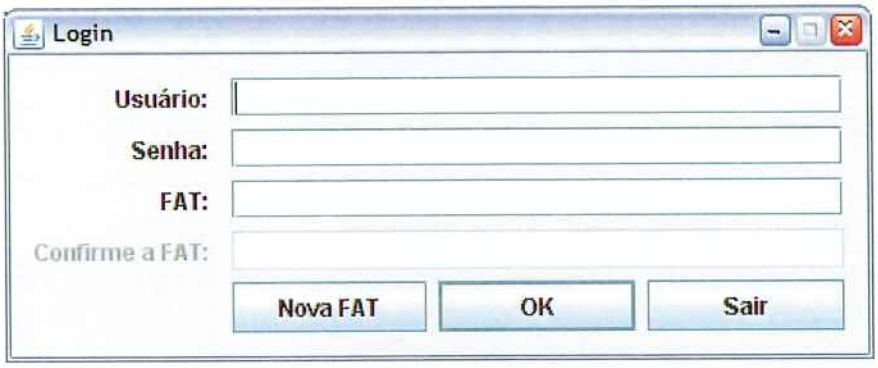
Passo 1: Executando a interface gráfica

O SEA possui uma interface gráfica de fácil utilização para auxiliar as operações com arquivos. Para executar a interface gráfica do SEA, digite:

java –jar sea

Passo 2: Identificação do usuário

No SEA é possível utilizar uma estrutura de diretórios, chamada de FAT, para facilitar o gerenciamento dos arquivos. Assim, quando o sistema é iniciado, a seguinte tela é exibida:



Você deve digitar o seu nome de usuário, senha e a estrutura de diretórios (FAT) que deseja utilizar e clicar em OK. Para criar uma nova estrutura, basta clicar no botão Nova FAT.

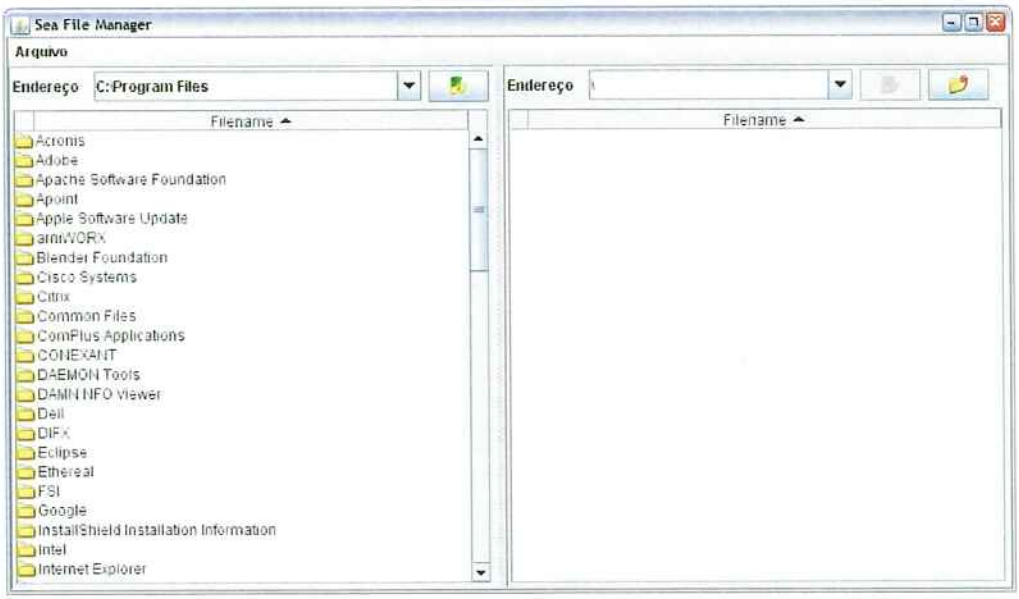
Passo 2: Gerenciamento de arquivos

A tela que aparece em seguida é dividida em duas partes. O lado esquerdo apresenta a estrutura de diretórios real do seu computador, enquanto o lado direito representa os diretórios da FAT carregada no passo anterior.

Neste ponto, você pode criar e remover pastas da sua estrutura. Para criar uma nova pasta, basta clicar no botão verde no canto superior direito da tela. Digite o nome da nova pasta na tela a seguir:

Para remover uma pasta, basta pressionar a tecla delete ou o botão vermelho, estando a pasta selecionada.

A figura abaixo exibe a tela principal do aplicativo SEA:



Caso o usuário deseje criar ou utilizar outras estruturas de diretórios dentro do volume esteganográfico, basta selecionar a opção Arquivo > Fat do menu.

Para efetuar a gravação de um arquivo selecione-o no lado esquerdo da janela e arraste-o para o lado direito. Para realizar a sua recuperação, basta arrasta-lo de volta para o local no disco real onde ele será gravado.

A renovação é uma operação importante para garantir a permanência do arquivo dentro do volume esteganográfico. Para realizar a renovação de um arquivo, selecione-o e vá ao menu Arquivo > Renovar. Para renovar um diretório inteiro, basta executar a mesma operação selecionando o diretório desejado.

Passo 3: Finalização

Para finalizar da utilização do sistema, basta fechar a aplicação.

12.3. ANEXO C – PITFALLS IN STEGANOGRAPHIC FILE SYSTEMS

Pitfalls in Steganographic File Systems

Fernando de Oliveira Gil[1] , **Roberto Hideo Morigaki**[1] ,
Leandro José Malandrin[1] , **Paulo S. L. M. Barreto**[1]

[1] Departamento de Engenharia de Computação e Sistemas Digitais (PCS),
Escola Politécnica, Universidade de São Paulo, Brazil.

Abstract. *In this paper we describe certain pitfalls that might lead to attacks when using some sort of information dispersal algorithms together with steganographic file systems. A simple solution to the problems is proposed, making used of cryptography to mitigate the possible attacks.*

Keywords: steganography.

1. Introduction

Steganographic file systems have been created to give users a new level of privacy to information, hiding its existence from attackers and offering plausible deniability to users. Without the secret the attacker cannot find out if the information is available or not, and therefore, can not force a user to reveal its contents.

The fact that it is impossible for any entity, including the system storing the information itself, discover where and if the information is stored, makes a requirement the existence of some kind of data redundancy, reducing the probability of an user corrupt blocks previously written. Several methods can be used to prevent this from happening, and among them are the basic replication of the data blocks inside the logical space and the use of information dispersal algorithms.

These algorithms split the information into m different blocks, but only $n < m$ need be successfully retrieved in order to reconstruct the original information. Using this method together with steganographic file systems described earlier might lead to certain pitfalls. Our contribution in this paper is a description of these problems and the proposal of a solution to them.

This paper is organized as follows. In section 2., we present a brief explanation on steganographic file systems and in section 3 we present the concept behind the information dispersal algorithms used in these systems. In Section 3. we describe two dispersal schemes used in such systems that might lead to problems. Section 4. introduces a simple solution to these problems. Section 5. concludes the paper.

2. Preliminaries

The algorithms mentioned earlier are used to distribute information across different locations in a way that it can be recovered even if some of the locations are no longer available.

One way to do that is transform the data into a geometrical object, for example a line or a plane. To describe a plane, all you need are 3 points in space. If you suppose these points are (X_n, Y_n, Z_n), where $n = 1, 2, 3$, you can create a linear system with equations of the type $Y_n = aX_n + bY_n + cY_n$. The coefficients a, b, c in this case describe the plane,

and that is the information which needs to be retrieved without loss from the system (the information that will be stored).

Now suppose it is necessary to have this information stored with the requirement of integrity even if one of the locations is down or corrupted. All it is needed is pick m points in the plane and distribute its coordinates instead of the equation coefficients themselves. As long as you are able to retrieve 3 of those m points, you still can obtain the original data by solving a linear system.

Off course, there is no need to be restricted to 3 dimensions and the number of coordinates for each point can be increased. However, increasing the size of the equation system to be solved can decrease the overall performance to store and retrieve information.

This kind of technique is very helpful when dealing with steganography file systems like [Hand and Roscoe 2002]. In these systems users can overwrite data blocks containing information previously written, since there is no way (or is wanted to be one) for the system to decide if there is any information stored in any location.

3. Potentially flawed schemes

We now present some flawed schemes using the information dispersal algorithm together with the steganographic file systems. If one utilizes the second construction utilized in [R. Anderson and Shamir 1998], it is almost mandatory that some sort of encryption is used to prevent a person from distinguishing between random data and actual encrypted data. However, the order between encryption step and dispersal step might lead to some problems, as we will see. It worth to note that no matter which method is used, if the basic idea is still the same, the problems described will have to be considered.

3.1. Scheme 1: Encryption step before the Information Dispersal step

Imagine a attacker trying to find a way to disrupt the access of users of a steganographic file system and assume that this attacker has read and write access to this system (for example, another user of the same system). In this scheme, a regular user of the system would gather a certain amount of data, split it into k blocks, encrypt each block with the cipher of its preference and try to disperse the k encrypted blocks along the logical storage space of the file system. Using the process described earlier this user will end up with m blocks from which only n must be retrieved to get the information back.

An attacker in this situation might want to find out, inside the steganographic storage, which blocks belongs to the same user or not. To do that, he can pick combinations of n blocks of the system and find out the data it produces, as a regular user would do, with the exception that this person does not know how to evaluate if the contents of the data are valid or not. However, by picking different combinations of n blocks from all the blocks available and storing the values obtained, he can select which combinations of blocks led to the same resulting data, obtaining this way the collection of blocks which correspond to the same user, among all the blocks inside the storage space.

This result leads to two different attacks:

1. The attacker might want to simply deny the service to the user, and all he has to do is write random information to the set of blocks found out to belong to the

same user (denial of service, or DoS, attack). This way the user will not have the minimum number of blocks to retrieve the information.

2. The attacker, if allowed to analyze traffic sent to the service provider, can find out which user are reading that set of blocks and therefore determine the identity of the user. The user can even try to include random writes and reads to random locations among the valid ones, and the attacker will still be able to determine its identity, as long the majority of writes and reads are directed at valid blocks.

3.2. Scheme 2: Information Dispersal step before Encryption step

The requirement of a data refresh procedure on some steganography systems as in [Hand and Roscoe 2002, R. Anderson and Shamir 1998] is ideally implemented as a scheduled task, making transparent to the user the half-life problem of the steganographic file system. However, if the information dispersal is done before any kind of encryption, that means the process which implements the refresh procedure will require not only knowing where the information is stored but will also need to know the key used to cipher that information. Therefore a process who has the only purpose of rewriting information will reveal its contents. That constitutes another vulnerability in the system.

4. New proposal

We propose the use of a second step of encryption together with the original encryption and dispersal, combining the schemes 1 and 2 presented above. Therefore, after gathering the data which is to be stored into the steganographic system, the process handling the storage will cipher its contents, apply the distribution algorithm, creating as many M extra blocks as specified and then a second step will be applied to each block created, with another key. The resulting blocks are the ones stored into the system. There is no need for the use of keys of the same strength in both encryption steps, since the second step is applied only to mitigate the denial-of-service attack and identity attacks presented in Scheme 1.

5. Conclusions

Steganographic file systems are privacy data storage. These systems require data to be encrypted and dispersed. In this paper, we presented two pitfalls in such file systems. If the system did encryption and then dispersal (scheme 1) an attacker could find files in the steganographic space. If the system did dispersal and then encryption (scheme 2), the data will be unprotected during the refresh process. To solve these problems, we propose to do encryption both before and after the dispersal, combining the two schemes together.

References

Hand, S. and Roscoe, T. (2002). Mnemosyne: Peer-to-peer steganographic storage. Available from `http://www.cl.cam.ac.uk/research/srg/netos/papers/2002-mnemosyne-iptps.pdf`.

R. Anderson, R. N. and Shamir, A. (1998). The steganographic file system. IWIH: International Workshop on Information Hiding.