FELIPE JOSÉ GOMES RIBEIRO

EASYSHARE - SISTEMA DISTRIBUÍDO DE PROTEÇÃO DE INFORMAÇÃO UTILIZANDO PARTILHA DE SENHAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS - BRASIL
2006

FELIPE JOSÉ GOMES RIBEIRO

EASYSHARE - SISTEMA DISTRIBUÍDO DE PROTEÇÃO DE INFORMAÇÃO UTILIZANDO PARTILHA DE SENHAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Criptografia

Orientador:
Prof. Ricardo Martins de Abreu Silva

Co-Orientador
Prof. Lucas Monteiro Chaves

LAVRAS
MINAS GERAIS - BRASIL
2006

.

Ficha Catalográfica preparada pela Divisão de Processos Técnico da Biblioteca Central da UFLA

Ribeiro, Felipe José Gomes

Easyshare: Sistema distribuído de proteção de informação utilizando partilha de senhas. / Felipe José Gomes Ribeiro. Lavras - Minas Gerais, 2006. 74p : il.

Monografia de Graduação - Universidade Federal de Lavras. Departamento de Ciência da Computação.

1.Informática. 2. Criptografia. 3. Partilha de senhas. I. Ribeiro, F.J.G. II. Universidade Federal de Lavras. III. Easyshare: Sistema distribuído de proteção de informação utilizando partilha de senhas.

FELIPE JOSÉ GOMES RIBEIRO

EASYSHARE - SISTEMA DISTRIBUÍDO DE PROTEÇÃO DE INFORMAÇÃO UTILIZANDO PARTILHA DE SENHAS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em *26 de Abril de 2006*

Prof. Heitor Augustus Xavier Costa

Prof. Rudini Menezes Sampaio

Prof. Ricardo Martins de Abreu Silva
(Orientador)

Prof. Lucas Monteiro Chaves
(Co-Orientador)

LAVRAS
MINAS GERAIS - BRASIL

Aos meus pais, Mary e Wilson, por garantirem que esse sonho fosse concluído. Dedico também à minha família e minha namorada Dani.

Agradecimentos

Gostaria de agradecer a ajuda que recebi de muitas pessoas, pois sem o apoio destas a realização deste trabalho não seria possível. Assim, meus agradecimentos vão para meus pais, que sempre estiveram ao meu lado nessa batalha para vencer mais uma fase da vida, minha namorada Dani, minha família, meu amigos e, principalmente aos mestres e amigos que tive a oportunidade de conviver no decorrer da minha graduação. Um obrigado aos meus orientadores Ricardo e Lucas, que contribuiram significativamente com esse trabalho.

Resumo

Este trabalho tem por objetivo utilizar a partilha de senhas através do método de polinômio de Lagrange no desenvolvimento de um sistema distribuído de proteção de informação denominado *Easyshare*

Abstract

The objective of this work consists in applying Lagrange polynom in the development of a distributed system for protecting information called Easyshare

Sumário

1 INTRODUÇÃO **1**
1.1 Motivação . . . 1
1.2 Escopo e objetivos . . . 2
1.3 Cronograma e metodologia . . . 2

2 FUNDAMENTOS E TERMINOLOGIA DA CRIPTOGRAFIA **4**
2.1 Considerações iniciais . . . 4
2.2 Algoritmos e chaves . . . 4
2.3 Algoritmos de criptografia baseados em chave . . . 5
2.3.1 Algoritmo de chave privada . . . 6
2.3.2 Algoritmo de chave pública . . . 6

3 ANÁLISE HISTÓRICA **8**
3.1 Considerações iniciais . . . 8
3.2 Aspectos históricos . . . 8

4 ANÁLISE SOCIAL **14**
4.1 Considerações iniciais . . . 14
4.2 Aspectos sociais . . . 14

5 TEORIA DOS NÚMEROS **16**
5.1 Considerações iniciais . . . 16
5.2 Algoritmos fundamentais . . . 16
5.2.1 Algoritmo euclidiano . . . 16
5.2.2 Algoritmo euclidiano estendido . . . 17
5.3 Fatoração única . . . 18
5.3.1 Teorema da fatoração única . . . 18
5.4 Aritmética modular . . . 19
5.4.1 Relações de equivalência . . . 19
5.4.2 Inteiros módulo n . . . 20
5.4.3 Aritmética modular . . . 22
5.4.4 Divisão Modular . . . 23
5.5 Sistemas de congruência . . . 25
5.5.1 Algoritmo chinês do resto . . . 26

6 O MÉTODO DA PARTILHA DE SENHAS **28**
6.1 Considerações iniciais . . . 28
6.2 Apresentação do problema . . . 28
6.3 Algoritmos de partilha de senhas . . . 31
6.3.1 Polinômio interpolador de Lagrange . . . 31
6.3.2 Esquema dos planos . . . 32
6.3.3 Karnin-Greene-Hellman . . . 32
6.3.4 Aplicação do algoritmo chinês do resto . . . 33

7 RESULTADOS E DISCUSSÕES **36**
7.1 Considerações iniciais 36
7.2 Apresentando o *EasyShare* 36
7.2.1 Características 36
7.2.2 Instalação e configuração 38
7.3 A ferramenta em execução 39
7.3.1 Controle de Usuários 41
7.3.2 Bloqueando e Desbloqueando Arquivos e Pastas 44
7.3.3 Gerência de senhas 51
7.4 Modelagem UML 52
7.4.1 Diagrama de casos de uso 52
7.4.2 Diagrama de classes dos subsistemas 53

8 CONSIDERAÇÕES FINAIS **60**
8.1 Conclusões 60
8.2 Trabalhos Futuros 60

A MODELAGEM DO BANCO DE DADOS **64**

B DETALHAMENTO DOS CASOS DE USO **65**

Lista de Figuras

2.1 Criptografar e descriptografar utilizando uma chave 5
2.2 Criptografar e descriptografar utilizando duas chaves 5
2.3 Criptografia de chave pública 6

3.1 Cifra dos Templários 9
3.2 A palavra CRIPTOGRAFIA, cifrada pela cifra dos templários 9
3.3 Pedra de Roseta 10
3.4 Cilindro de Jefferson 11
3.5 Máquina de criptografia ENIGMA 12

6.1 Esquema $(2, n)$ de partilha 29
6.2 Esquema $(3, n)$ de partilha 30
6.3 Esquema de Blakley 33

7.1 Arquitetura cliente-servidor aplicada ao *EasyShare* com n computadores 37
7.2 Estrutura de diretórios do *EasyShare* 38
7.3 Conteúdo do arquivo *conec.txt* 39
7.4 Tela inicial do *EasyShare* 39
7.5 Janela para a realização do login no sistema 40
7.6 Funcionalidades inativas até que se realize login no sistema 40
7.7 Tela que gera string criptografada 41
7.8 Exemplo de informação criptografada 42
7.9 Usuário comum não tem permissões para gerenciar usuários. 42
7.10 Usuário administrador com o controle de usuários liberado. 43
7.11 Janela de cadastro de novos usuários. 43
7.12 Listagem dos usuários cadastrados. 44
7.13 Janela onde se altera informações ou remove usuário. 44
7.14 Janela para realizar o bloqueio (aba Opções gerais). 45
7.15 Janela para realizar o bloqueio (aba Usuários). 46
7.16 Tarefa 70% completa. 46
7.17 Tarefa já completa. 47
7.18 Mensagem de nova parte de senha mostrada ao usuário. 48
7.19 Janela para desbloqueio de informação. 49
7.20 Depois de selecionar o arquivo, informações sobre a proteção são exibidas. 49
7.21 Procurando por usuários disponíveis. 50
7.22 Requisitando partes aos usuários. 50
7.23 Janela que requisita a senha ao usuário. 51
7.24 Fim do processo de liberação de informação. 52
7.25 Janela que requisita a senha ao usuário. 52
7.26 Janela que requisita a senha ao usuário. 53
7.27 Diagrama de casos de uso do *EasyShare*. 54
7.28 Susbsistemas do *EasyShare*. 54
7.29 Diagrama de classes do subsistema IHM. 55

7.30 Diagrama das classes Details. 56
7.31 Diagrama das classes Handler 57
7.32 Diagrama das classes DAO 58
7.33 Relacionamento entre as classes. 59

A.1 Modelagem entidade relacionamento do banco de dados. 64

Lista de Tabelas

5.1 Divisões do algoritmo euclidiano estendido 18
5.2 Alguns elementos da classe $\overline{3}$ para $\mathbb{Z}_6$ 22

7.1 Tempo gasto para bloquear uma pasta de tamanho 11,9 MB 47
7.2 Tempo gasto para bloquear um arquivo de tamanho 136 bytes 48

Capítulo 1

INTRODUÇÃO

1.1 Motivação

Nos últimos anos observou-se que um dos problemas mais árduos que a indústria de computadores enfrenta é o da segurança de dados. O assunto é ainda mais complicado porque a segurança dos dados está relacionada ao problema social ainda mais amplo do sigilo, que frequentemente se torna um tópico emocional. Isso [Júnior, 1977] mostra através do fato de que as organizações sempre reuniram informações de vários tipos, mas o problema da segurança veio ao conhecimento público em anos recentes com a difusão do uso de computadores, sistemas de informação e facilidades de telecomunicações, provocando assim um avanço extraordinário na disseminação e na popularização de serviços que utilizam uma rede de computadores, sendo que a Internet ocupa a posição de destaque. Por outro lado, com a Internet, grandes questões que vêm preocupando tanto especialistas quanto usuários comuns estão relacionadas à segurança. Isso se dá ao fato de que a proteção da privacidade de pessoas que tem seus dados, ou segredos, circulando pela rede deve ser extremamente forte, pois os usuários estão sujeitos à fraudes, falsificações eletrônicas e até mesmo disseminação de vírus de computadores. Pode-se observar a gravidade desses problemas analisando a tendência atual da Internet, que promove o advento das "lojas virtuais" utilizadas para o comércio eletrônico e dos *home-banking*, que possibilitam aos clientes de um determinado banco realizarem transações através da internet. Além desses serviços também pode-se citar a troca de correspondências através de *e-mails* e até mesmo a troca de mensagens instantâneas entre usuários localizados em diferentes locais do mundo, fazendo com que distâncias sejam encurtadas e fronteiras perdidas. Deve-se salientar que um computador que não esteja conectado a nenhum outro, ou até mesmo à internet, também está vulnerável à problemas de segurança, pois se uma pessoa mal intencionada estiver no comando da máquina ela poderá ter acesso a qualquer tipo de informação ali presente. Assim, verifica-se que todos esses serviços necessitam de técnicas de segurança que façam com que a pessoa se sinta protegida ao acessar, gravar, enviar ou receber qualquer tipo de informação através de seu computador. A chave para resolver tecnologicamente estes problemas é a *Criptografia*, que tem como objetivo, basicamente, "esconder" informações sigilosas de qualquer pessoa *desautorizada* a lê-las.

Segundo [Schneier, 1996], durante os últimos vinte anos, a pesquisa acadêmica na área tem crescido

constantemente. Mas a criptografia computadorizada foi, durante muito tempo, um domínio exclusivo das forças militares americanas e o início se deu a partir da época da Segunda Guerra Mundial.

Hoje, praticamente todo o estado da arte da criptografia computadorizada está localizado fora dos muros de proteção das agências militares, permitindo ao cidadão comum ter acesso à toda essa segurança que irá protege-lo de possíveis ameaças futuras.

1.2 Escopo e objetivos

Como a criptografia é uma área bastante abrangente, este trabalho é voltado para uma subárea denominada *partilha de senhas*. Esse método, como o próprio nome já diz, permite o compartilhamento de um segredo (uma senha, ou uma mensagem) entre várias pessoas. Isso tem se tornado cada vez mais necessário, pois hoje existem segredos que são tão importantes que não podem ser confiados à uma única pessoa, e [Buholzer, 1999] assume que a melhor coisa a ser feita é distribuí-lo entre várias pessoas. Assim, a partilha de senhas permite que se tenha essa confiança nos segredos que lhe são submetidos sem que ocorra uma exposição da segurança ou um aumento de risco sobre o sistema. Portanto, uma corporação que possua muitos funcionários realizando operações críticas pode utilizar a partilha de senhas. Basta distribuir uma senha entre n funcionários, sendo que quaisquer $k(k \leq n)$ destes funcionários podem iniciar as operações críticas (como por exemplo, abrir conta em bancos, assinar cheques corporativos, acessar um determinado arquivo em um servidor ou até mesmo lançar mísseis nucleares).

O objetivo principal desse trabalho foi desenvolver um sistema que implemente o método de Lagrange para resolver o problema da partilha de senhas e tenha um controle sobre a utilização das informações de arquivos e pastas de uma rede de computadores.

1.3 Cronograma e metodologia

O trabalho foi desenvolvido utilizando os materiais e métodos descritos a seguir:

- foi realizado um levantamento bibliográfico, na internet e em bibliotecas, de livros, artigos científicos clássicos e atuais relacionados ao tema. Em paralelo, foi realizado um estudo do que seria o problema da partilha de senhas, propriamente dito. Isto foi feito através de uma análise detalhada do material coletado. Quando o estudo bibliográfico foi concluído, fez-se um estudo das técnicas da partilha de senhas. Este foi o estado da arte;

- após estas atividades preliminares, foi realizado um estudo de como seriam implementadas, computacionalmente, tais técnicas que servem como uma ferramenta de segurança de informações, ou até mesmo como uma ferramenta didática a futuros interessados;

- uma vez que as formas da implementação foram definidas, foi implementado um ambiente em Java utilizando banco de dados MySQL [MySQL, 2005] que utiliza algoritmos de partilha de senhas para

proteger informações distribuídas em uma rede de computadores;

- terminada a implementação, foi necessário passar à etapa de testes em laboratório com o uso de exemplos práticos. Foram usados para ambiente de testes o laboratório II do Departamento de Ciência da Computação da Ufla e uma rede doméstica com 2 computadores;
- a última fase foi a realização de uma apresentação do trabalho que visou mostrar os resultados obtidos e a maneira como a pesquisa foi desenvolvida.

Capítulo 2

FUNDAMENTOS E TERMINOLOGIA DA CRIPTOGRAFIA

2.1 Considerações iniciais

A seguir será apresentado um estudo sobre os vários termos e métodos utilizados na área de criptografia, pois vários deles podem tornar-se confusos, dificultando assim uma melhor compreensão do tema proposto.

2.2 Algoritmos e chaves

Segundo [Schneier, 1996] um *algoritmo criptográfico* é uma função matemática usada para criptografar e decriptografar. Criptografar uma mensagem significa torná-la ilegível para qualquer pessoa não autorizada. E, decriptografar é o processo inverso que transforma uma mensagem ílegivel em legível. Mas deve-se observar que, geralmente, existem duas funções que se relacionam: uma para criptografar e outra para decriptografar.

Se a segurança de um algoritmo é baseada na maneira de como é seu funcionamento, então dá-se o nome de *algoritmo restrito*. Esse tipo de algoritmo possuiu somente interesses históricos, pois são totalmente inadequados para os padrões atuais. Pode-se imaginar um grupo muito grande de pessoas que queira utilizar um algoritmo como esse, mas a todo momento existem pessoas que entram e saem do grupo. Se alguém, acidentamente, revelar o segredo, todo o grupo deverá mudar o algoritmo e isso mostra como esse tipo de sistema é falho. E, sendo mais condenatório, algoritmos restritos não possuem controle de qualidade ou padronização, fazendo com que cada grupo de usuários deva possuir seu próprio algoritmo implementado. Se ninguém do grupo possuir os conhecimentos básicos e necessários para compreender a criptografia, a segurança do sistema nunca será comprovada. Assim, esses algoritmos são popularmente conhecidos pela sua baixa segurança.

A criptografia moderna resolve esse problema com uma *chave*, denotada de K. E essa chave pode ser qualquer valor dentro de uma larga escala de valores. Assim, de acordo com a figura 2.1, ambas as funções de criptografar e descriptografar utilizam essa chave K, isto é, elas dependem dessa chave

para realizar suas funções. Agora, pode-se escrever as funções de criptografar(E) e descriptografar(D) da seguinte maneira (onde M é a mensagem original e C é a mensagem cifrada):

$$E_K(M) = C$$

$$D_K(C) = M$$

Essas funções tem a seguinte propriedade:

$$D_K(E_K(M)) = M$$

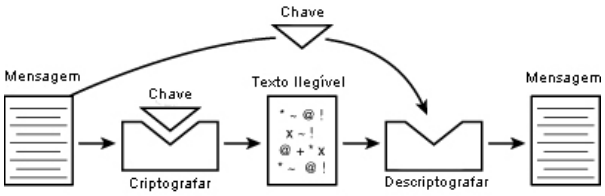

Figura 2.1: Criptografar e descriptografar utilizando uma chave

Alguns algoritmos usam chaves diferentes para criptografar e descriptografar, como pode ser visto na figura 2.2. Portanto, utiliza-se a chave K_1 para criptografar e a chave K_2, diferente de K_1, para descriptografar. Então, nesse caso:

$$E_{K1}(M) = C$$

$$D_{K2}(C) = M$$

$$D_{K2}(E_{K1}(M)) = M$$

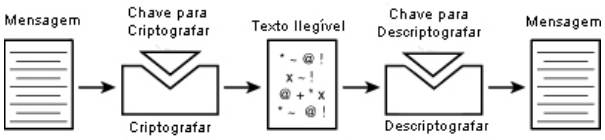

Figura 2.2: Criptografar e descriptografar utilizando duas chaves

2.3 Algoritmos de criptografia baseados em chave

Segundo [Schneier, 1996], existem dois tipos gerais de algoritmos baseados em chaves: *algoritmos de chave privada* e *algoritmos de chave pública*.

2.3.1 Algoritmo de chave privada

Os algoritmos de chave privada também são conhecidos como algoritmos simétricos e são caracterizados pelo fato de que a chave de criptografar é calculada a partir da chave para descriptografar e vice-versa, sendo que na maioria das vezes essas duas chaves são iguais. Nesses sistemas de criptografia, o remetente e o destinatário devem concordar previamente em utilizar uma determinada chave para que a comunição entre eles ocorra de maneira segura. Portanto, a segurança de um algoritmo de chave privada é toda baseada na chave, pois caso a chave seja divulgada qualquer um poderá criptografar ou descriptografar mensagens. Assim, enquanto a chave permanecer segura, a comunicação entre remetente e destinatário também irá permanecer segura.

Criptografar e descriptografar utilizando um algoritmo simétrico é denotado por:

$$E_K(M) = C$$

$$D_K(C) = M$$

2.3.2 Algoritmo de chave pública

Está é uma idéia introduzida em 1976 por W. Diffie e M.E. Hellman na Universidade de Standford e, independentemente, por R.C. Merkle da Universidade da Califórnia. A criptografia de chave pública também é conhecida como criptografia assimétrica, e ela se baseia no fato de que a chave usada para criptografar é diferente da chave utilizada para descriptografar, como visto na figura 2.3. E, diferentemente da criptografia simétrica, a chave para descriptografar não poderá ser calculada a partir da chave usada para criptografar. A partir daí verifica-se que o *algoritmo de chave pública* tem esse nome porque a chave que criptografa pode se tornar pública ou, em outras palavras, um estranho poderá criptografar uma mensagem, mas somente uma determinada pessoa terá acesso à mensagem ali presente. No algoritmo de chave pública a chave para criptografar é chamada de *chave pública* e a chave para descriptografar e chamada de *chave privada*.

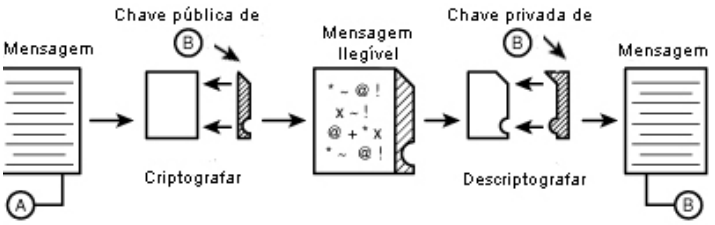

Figura 2.3: Criptografia de chave pública

Criptografar com chave pública implica que:

$$E_K(M) = C$$

Como a chave pública é diferente da chave privada, a descriptografia com a chave privada correspondente é denotada por:

$$D_K(C) = M$$

Capítulo 3

ANÁLISE HISTÓRICA

3.1 Considerações iniciais

Como dito anteriormente, nos últimos anos observou-se o grande avanço da internet e das redes de computadores. Mas, juntamente com esse avanço, chega a crescente ameaça que se sofre diariamente quando se usa serviços que utilizam de dados pessoais ou sigilosos no tráfego de informações. Assim surgiu a criptografia, no intuito de "esconder" dados e informações pessoais de pessoas desautorizadas. A seguir, apresenta-se uma análise histórica do surgimento e da evolução das técnicas de *criptografia*.

3.2 Aspectos históricos

Inicialmente, pode-se dizer que *criptologia* é a disciplina científica que reúne e estuda os conhecimentos (matemáticos, computacionais, psicológicos, filológicos, etc.) e técnicas necessários à criptografia. Assim, será mostrado um pouco da criptologia para que se possa entender a evolução da criptografia e alguns de seus conceitos básicos.

A história acontece numa vila egípcia perto do rio Nilo chamada Menet Khufu. Khnumhotep II era um arquiteto do faraó Amenemhet II. Ele construiu alguns monumentos para o faraó, os quais precisavam ser documentados. E estas informações, escritas em tabletes de argila, não eram para cair no domínio público.

O escriba de Khnumhotep II teve a idéia de substituir algumas palavras ou trechos de texto destes tabletes. Caso o documento fosse roubado, o ladrão não encontraria o caminho que o levaria ao tesouro - morreria de fome, perdido nas catacumbas da pirâmide.

[Kahn, 1996] considera isto como o primeiro exemplo documentado da escrita cifrada.

Segundo [Gomes, 2000], *cryptos*, em grego, significa secreto, oculto, e *grapho* significa escrita. Assim, diz-se que a criptografia é a arte da "escrita secreta" ou, para ser mais abrangente, a arte dos "códigos secretos". Sua utilização, como foi mostrado por [Kahn, 1996], é datada de muito tempo atrás, e sua evolução divida em três fases distintas: a *criptografia manual*, a *criptografia por máquinas* e a *criptografia em rede*.

A criptografia manual se baseia na utilização de um dos códigos mais simples, que consiste em substituir uma determinada letra pela letra seguinte do alfabeto, ou seja, transladar o alfabeto uma casa para diante. Júlio César usou este tipo de cifragem para esconder mensagens governamentais e também para poder se comunicar com suas legiões em combate. Para compor seu texto cifrado, César alterou letras desviando-as em três posições; A se tornava D, B se tornava E, etc. Às vezes, César reforçava sua cifragem substituindo letras latinas por gregas.

O código de César é o único da antiguidade que é usado até hoje, apesar de representar um retrocesso em relação à criptografia existente na época. Atualmente denomina-se qualquer cifra baseada na substituição cíclica do alfabeto de código de César.

De acordo com [Loewenguth, 2004], um outro exemplo de cifra que foi bastante utilizada provem da ordem do Templo, que atuou internacionalmente de 1119 até 1312, e utilizava uma cifra própria. Os templários cifravam as letras de crédito que eles mantinham em circulação entre seus nove mil postos de comando. Desta forma, as letras de crédito, que evitavam o transporte de riquezas, circulavam protegidas e "autenticadas".

O cifrante foi extraído da cruz chamada "das oito beatitudes" e que constituía o emblema da ordem (3.1)

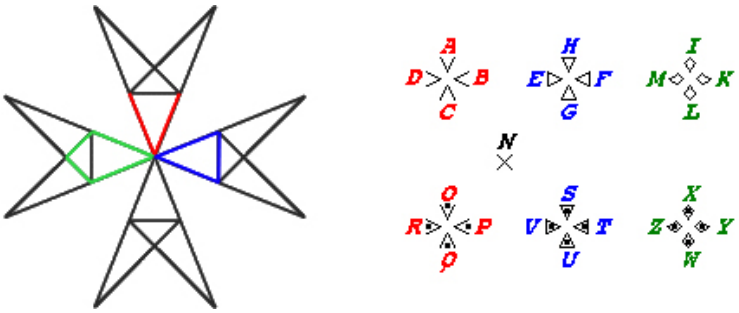

Figura 3.1: O cifrante da ordem do Templo

De acordo com o diagrama, cada letra do alfabeto é substituída pelas linhas indicadas em vermelho, azul e verde.

Assim, a palavra

CRIPTOGRAFIA

seria escrita da forma mostrada na figura 3.2.

Figura 3.2: A palavra CRIPTOGRAFIA, cifrada pela cifra dos templários

Outros exemplos de criptografia manual são a *cítara espartana* e o *cifrário de Francis Bacon*, mas não há a necessidade de se entrar em maiores detalhes sobre esses métodos, pois todos eles se baseiam

na criptografia mais simples, onde a codificação de mensagens era feito através de algum processo pré determinado, como as substituições de certas letras por outras, ou a susbtituição de letras por símbolos, como já foi mostrado anteriormente.

Contudo, esses métodos de criptografia possuem um grande defeito: são muito fáceis de serem decifrados. E, de acordo com [Coutinho, 2003], qualquer código que envolva substituir cada letra sistematicamente por outro símbolo qualquer sofre do mesmo problema, sendo que [Terada, 2000] nos leva a compreender que isso se deve ao fato de que a freqüência média com que cada letra é usada em uma língua é mais ou menos constante. Assim, pode-se citar o exemplo da língua portuguesa, onde as vogais aparecem mais vezes, sendo que a vogal mais freqüente é a letra A, com 13,5% de freqüencia, seguida por E com 12,5%, I com 6,0%, O com 5,5% e U com 4,5%. Outras características da língua portuguesa são:

- se um monossílabo tem uma única letra, então essa letra é uma vogal;
- a consoante mais freqüente é P, com 11,5% de freqüencia, seguida de T com 9% e S com 8%.

Assim, apenas contando a freqüencia de cada símbolo no texto, pode-se descobrir a que letra correspondem os símbolos mais freqüentes. Isto geralmente é suficiente para decifrar toda a mensagem. Mas, deve-se observar que este método só funciona se a mensagem for longa.

O método de *contagem de freqüencia* de caracteres foi usado para decifrar inscrições antigas, sendo que o exemplo mais famoso é o da decifração dos hieróglifos egípcios por J.F. Champollion em 1822. A chave para a decifração foi a descoberta da *Pedra de Roseta*, mostrada abaixo na figura 3.3, que é um bloco de basalto negro que contém uma mesma inscrição escrita em hieróglifos, demótico e grego.

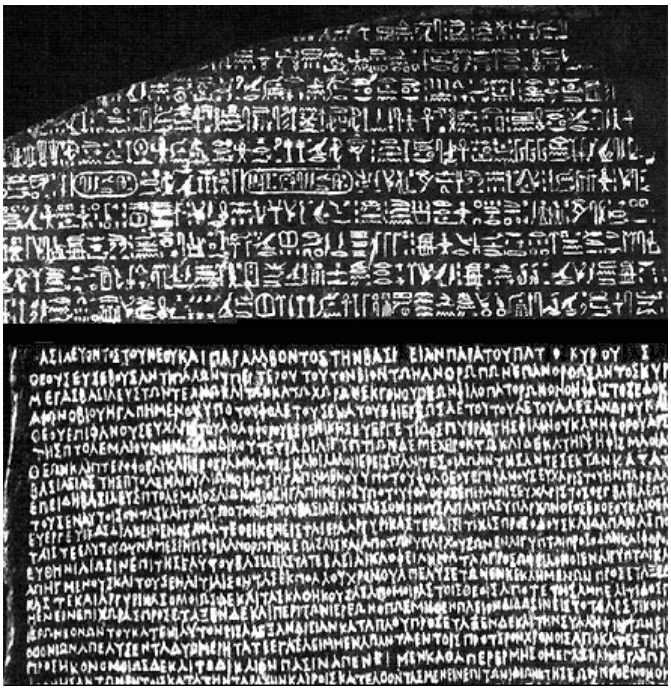

Figura 3.3: Fragmentos da pedra de Roseta. Hieróglifos(acima) e Grego(abaixo)

Portanto, para decifrar os símbolos egípcios, Champollion se baseou na inscrição em grego, e a partir

daí comparou a freqüência das letras gregas com a freqüencia dos hieróglifos até encontrar um determinado padrão.

Agora, com os recursos que se possui nos dias de hoje, fica claro que decifrar uma mensagem por contagem de freqüencia é ainda mais simples se existe um computador. Supondo que a língua seja conhecida, a maior parte do processo pode ser automatizado, tornado assim totalmente inviáveis todos os códigos que envolvem substituição de letras.

A segunda fase da evolução da criptografia é chamada de *criptografia por máquinas*, onde basicamente, existia uma tabela predeterminada que era usada em conjunto com uma máquina, onde o operador desta, usando a tabela e manipulando a máquina podia enviar uma mensagem criptografada. Um exemplo deste tipo de criptografia surgiu por volta do ano de 1800 e foi idealizada por Thomas Jefferson, que era secretário de estado de George Washington e futuro presidente dos Estados Unidos.

O cilindro de Jefferson (*Jefferson's wheel cipher* em Inglês), na sua forma original, é composto por 26 discos de madeira que giram livremente ao redor de um eixo central de metal. As vinte e seis letras do alfabeto são inscritas aleatoriamente na superfície mais externa de cada disco de modo que, cada um deles, possua uma sequência diferente de letras. Girando-se os discos pode-se obter as mensagens.

Observe na figura 3.4 que, numa das linhas, é possível ler "THOMAS JEFFERSON WHEEL CIPHER". Esta seria a mensagem clara. O remetente, no entanto, escolhe qualquer outra linha e a envia ao destinatário. Tomemos como exemplo a linha imediatamente acima da mensagem clara. Neste caso, a mensagem cifrada enviada seria:

MZNCSKYONSLKTRFAJQQBRTXYUK

O destinatário, que possui um cilindro com a mesma sequência de discos, transfere a mensagem recebida para o seu cilindro e procura uma linha que possua texto que faça sentido. No nosso exemplo, ele encontra a mensagem decifrada na linha imediatamente inferior à da mensagem cifrada.

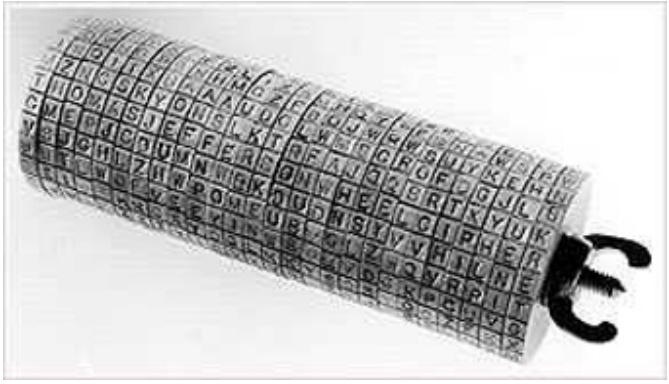

Figura 3.4: Cilindro idealizado por Thomas Jefferson

Outro exemplo de criptografia com máquinas foi muito utilizado pelos alemães na Segunda Guerra Mundial, através da famosa máquina de criptografia ENIGMA, que codificava e decodificava os códigos

secretos que eram repassados durante toda a guerra. A sua chave era definida através da posição de rotores na parte superior da máquina, como indicados na figura 3.5.

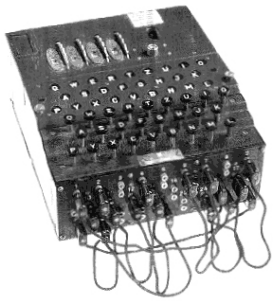

Figura 3.5: ENIGMA: utilizada pelos alemães em 1944

Segundo [Kahn, 1996], a Segunda Guerra Mundial também contribuiu indiretamente para o surgimento da terceira fase da evolução da criptografia: a *criptografia em rede*. Essa contribuição indireta aconteceu porque foi nesse período que surgiram os primeiros computadores, inicialmente com o objetivo principal de se decifrar os códigos que eram repassados entre os inimigos. Sendo que posteriormente essas máquinas teriam papel fundamental para o estudo e o desenvolvimento de técnicas de criptografia. Assim, a criptografia em rede destaca-se pela utilização de algoritmos responsáveis pela execução dos métodos criptográficos, tornando-os mais rápidos e seguros.

De acordo com [Terada, 2000], até o fim da década de 70, a maioria dos algoritmos criptográficos eram secretos, principalmente aqueles utilizados pela diplomacia e pelas forças armadas de cada país. Assim, a criptografia evoluiu como ciência a partir desse cenário inicial, conhecido como esfera militar, onde os programas e informações importantes eram protegidos de uma forma mais autoritária, o que favorecia a chamada contra-espionagem. Mas com o início da década de 80, o cenário da criptografia iniciou um processo de modificação, fazendo com que as técnicas utilizadas nesses métodos de segurança se tornassem mais acessíveis ao público e ao setor comercial, que também pretendia adotar políticas seguras. Assim, os programas de criptografia que possuiam algoritmos fechados, ou seja, só o autor conhecia e testava, não passaram pelo crivo evolutivo da competição, pois a informatização da sociedade trouxe a necessidade e o uso da criptografia a outros cenários. Nos dias de hoje a segurança dos sistemas criptográficos são baseadas no conhecimento da chave privada, pois para se comprovar a segurança de um algoritmo criptográfico, basta publicá-lo e tê-lo analisado pelos criptoanalistas[1]. Se o algoritmo passa por tal análise, a industria aceita-o como seguro em relação ao estado da arte.

No final da década de 70, mais precisamente no ano de 1979, que o israelense Adi Shamir [Shamir, 1979] apresentou um novo método de criptografia, que é baseado na idéia de um trabalho em grupo; ou seja, a segurança dos dados dependia do trabalho conjunto de determinados indivíduos de um grupo qualquer. Esse

[1]Pessoa com conhecimento avançado em criptografia. É responsável por aplicar esses conhecimentos para descobrir alguma vulnerabilidade do algoritmo criptográfico.

método é denominado de *partilha de senhas* (ou *secret sharing*, em inglês) e vem sendo muito utilizado nos dias de hoje, com o intuito de assegurar que determinados dados não sejam usados indevidamente e até mesmo para impedir o uso destes para benefícios pessoais. Essa idéia foi apresentada por Shamir em seu artigo entitulado *"How to Share a Secret"* de 1979 e que rendeu uma valorosa conquista no campo da criptografia.

Capítulo 4

ANÁLISE SOCIAL

4.1 Considerações iniciais

Este capítulo tem por finalidade mostrar os principais aspectos sociais que estão presentes no contexto da criptografia e da partilha de senhas.

4.2 Aspectos sociais

Desde que o mundo foi criado que o ser humano tem sentido a necessidade de guardar segredos. Sejam segredos familiares, segredos sentimentais, segredos pessoais, segredos religiosos, ou segredos militares e governamentais. Tão forte quanto a necessidade nata da espécie humana de guardar segredo sobre determinados assuntos é a vontade dos mesmos humanos de desvendar esses segredos. Seja por dinheiro, poder, vingança, curiosidade, arrogância, ou qualquer outro sentimento, essa tem sido uma batalha que, ao longo dos anos vem sendo travada entre aqueles que querem guardar segredos e os que querem desvendar esses segredos.

Desta maneira, pode-se observar que a confiança entre pessoas é um assunto que é tratado com muito cuidado, pois os muitos fatores que foram citados anteriormente são facilmente detectados devido à essa natureza humana de querer desvendar segredos. Portanto, quando se tem muito poder e dinheiro envolvidos em determinada situação, deve-se avaliar com muita clareza e precisão quais os riscos que podem estar sendo expostos e, principalmente, quais os danos que uma possível falha poderá acarretar.

[Júnior, 1977] afirma que o sigilo é um assunto controvertido, emocional e interessante. Dessa forma, [Westin, 1967] citado por [Júnior, 1977] define esse aspecto sigiloso como: "*Sigilo é o direito de indivíduos, grupos ou instituições de determinar para eles mesmos quando, como e até que ponto informações sobre eles podem ser comunicadas a outros.*"

[Júnior, 1977] defende a idéia de que a razão pela qual o sigilo está relacionado de modo significativo aos sistemas de segurança de dados vem do fato de que o sigilo é uma parte integrante da sociedade e afeta o comportamento de seus membros. Isso mostra que uma organização requer também sigilo para alcançar seus objetivos básicos. Agências governamentais, companhias privadas, partidos políticos, todos precisam estar livres da revelação dos produtos competitivos, processos de tomadas de decisões, proce-

dimentos operacionais internos e informações prejudiciais sobre os membros da organização. Em muitos casos, o sucesso de uma organização é baseado em sua “imagem” ou “aparência externa”, de modo que a revelação dos negócios internos privados seria prejudicial ao sucesso da organização. Em outras palavras, a oportunidade e as comunicações privadas definem a diferença entre o sucesso e o fracasso.

Obviamente, segundo [Júnior, 1977], surgiram questões legais e morais que estão além do objetivo da segurança de dados em um sistema de informações computadorizado. Mesmo assim, embora os sistemas computacionais não possam controlar diretamente o comportamento dos indivíduos, pode-se desenvolver medidas efetivas, de modo que a proteção contra o acesso não autorizado aos arquivos de informações possa ser assegurada.

Para lidar com fatos como estes a criptografia se torna extremamente útil, pois faz com que informações importantes se tornem ilegíveis para pessoas desautorizadas a acessá-las. Mas não basta que se tenha apenas um sistema seguro que protege dados importantes de pessoas desautorizadas; é necessário que a pessoa responsável por esses dados seja de extrema confiaça, pois se este não for o caso, todo o sistema adotado perde sua aplicação. Este tipo de problema pode ser facilmente observado nos dias de hoje, pois muitos funcionários de variadas empresas “vendem” informações que são de extrema importância, ou até mesmo por falta de satisfação no ambiente de trabalho acabam liberando esses preciosos dados.

Outro problema, citado por [Coutinho, 2003], mostra que, hoje em dia, é relativamente fácil interceptar mensagens enviadas por linha telefônica, o que faz com que seja necessário codificá-las sempre que contenham informações sensíveis. Isto inclui transações bancárias, comerciais, ou até mesmo uma compra feita com cartão de crédito.

Estes problemas mostram que a comunicação entre computadores pela Internet vem criando novos desafios para a criptografia moderna, pois a necessidade de proteção de informações nos dias de hoje é um fator decisivo para que empresas possam continuar no mercado, ou até mesmo para garantir a privacidade de um cidadão diante dos riscos que a comunicação globalizada pode lhe oferecer.

Foi a partir da junção da necessidade de um sistema seguro e a falta de confiança entre as pessoas que surgiu o método proposto por [Shamir, 1979], que força uma cooperação entre as partes para que uma informação possa se manter segura, e até mesmo evita que uma única pessoa possa se beneficiar com a utilização indevida e egoísta daquilo que está em suas mãos.

Capítulo 5

TEORIA DOS NÚMEROS

5.1 Considerações iniciais

A descrição mais detalhada e completa do método da *partliha de senhas* será dada posteriormente, pois verifica-se a necessidade de um vasto desenvolvimento de muitas idéias e técnicas matemáticas. Assim, o propósito deste capítulo é mostrar a fundamentação da área da matemática denominada de *teoria dos números*, que é necessária para o estudo do problema proposto, assim como suas demonstrações e aplicações.

5.2 Algoritmos fundamentais

5.2.1 Algoritmo euclidiano

O objetivo deste algoritmo é calcular o máximo divisor comum entre dois números inteiros. Mas antes de iniciar o estudo do algoritmo euclidiano, é recomendado rever o *teorema da divisão* que se aprende no primário, mas com algumas novidades.

Supondo que se queira dividir a por b, tem-se:

Teorema da divisão: *Sejam a e b inteiros positivos. Existem números inteiros q e r tais que*

$$a = bq + r \qquad \text{e} \qquad 0 \leq r < b$$

Além disso, os valores de q e r satisfazendo as relações acima são únicos.

No teorema da divisão pode-se definir q e r como sendo quociente e resto.

Agora, retornando ao algoritmo euclidiano, diz-se que para calcular o máximo divisor comum entre a e b, é feita a seguinte sequência de divisões:

$$\begin{aligned}
a &= bq_1 + r_1 \quad \text{e} \quad 0 \leq r_1 < b \\
b &= r_1q_2 + r_2 \quad \text{e} \quad 0 \leq r_2 < r_1 \\
r_1 &= r_2q_3 + r_3 \quad \text{e} \quad 0 \leq r_3 < r_2 \\
r_2 &= r_3q_4 + r_4 \quad \text{e} \quad 0 \leq r_4 < r_3 \\
&\vdots \qquad\qquad\quad \vdots
\end{aligned}$$

Analisando a sequência de restos, observa-se que o *seguinte é sempre menor que o anterior* mas *todos são maiores ou iguais a zero*.

Escrevendo essas sequências de forma seguida, obtêm-se

$$b > r_1 > r_2 > r_3 > r_4 \ldots \geq 0$$

Como entre b e 0 há apenas uma quantidade finita de números inteiros, esta sequência não pode continuar indefinidamente. Ou, seja, ela chega ao final quando um dos restos é zero, e isto garante que o algoritmo sempre pára.

5.2.2 Algoritmo euclidiano estendido

Ainda não foi possível extrair toda a informação que será usada posteriormente sobre o máximo divisor comum. Então, sejam a e b inteiros positivos, e d o máximo divisor comum. Portanto, é possível encontrar inteiros α e β tais que

$$\alpha \cdot a + \beta \cdot b = d.$$

A não ser em casos triviais, ou α ou β é um inteiro negativo. O algoritmo euclidiano modificado que calcula α e β simultaneamente é denominado *algoritmo euclidiano estendido*.

A idéia, onde a implementação foi dada por [Knuth, 1981], é que para efetuar os cálculos correspondentes a uma certa divisão, basta guardar os dados referentes às duas divisões imediatamente anteriores. Assim, a sequência de divisões pode ser reescrita da seguinte forma:

$$\begin{array}{ccc}
a = bq_1 + r_1 & \text{e} & r_1 = ax_1 + by_1 \\
b = r_1q_2 + r_2 & \text{e} & r_2 = ax_2 + by_2 \\
r_1 = r_2q_3 + r_3 & \text{e} & r_3 = ax_3 + by_3 \\
r_2 = r_3q_4 + r_4 & \text{e} & r_4 = ax_4 + by_4 \\
\vdots & & \vdots \\
r_{n-3} = r_{n-2}q_{n-1} + r_{n-1} & \text{e} & r_{n-1} = ax_{n-1} + by_{n-1} \\
r_{n-2} = r_{n-1}q_n & \text{e} & r_n = 0.
\end{array}$$

Os números $x_1, \ldots, x_{n-1}$ e $y_1, \ldots, y_{n-1}$, são os inteiros que serão determinados, portanto essas informações podem ser condensadas na tabela 5.1

A primeira e a segunda linha são os 'casos base' e por isso são denotados como linhas -1 e 0. Assim adota-se $x_{-1} = 1, y_{-1} = 0, x_0 = 0, y_0 = 1$. Agora, o algoritmo se preocupa em preencher a tabela até encontrar a condição de parada, fazendo com que os últimos valores encontrados para x e y sejam α e β. Ou melhor

$$\alpha = x_{n-1} \qquad \text{e} \qquad \beta = y_{n-1}$$

restos	**quocientes**	x	y
a	*	x_{-1}	y_{-1}
b	*	x_0	y_0
r_1	q_1	x_1	y_1
r_2	q_2	x_2	y_2
r_3	q_3	x_3	y_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_{n-2}	q_{n-2}	x_{n-2}	y_{n-2}
r_{n-1}	q_{n-1}	x_{n-1}	y_{n-1}

Tabela 5.1: Divisões do algoritmo euclidiano estendido

A condição de parada sempre é encontrada, pois este algoritmo é uma extensão do algoritmo euclidiano, e como já foi visto, o algoritmo euclidiano possui uma condição de parada bem definida.

5.3 Fatoração única

Segundo [Coutinho, 2003], dividir para consquistar é uma estratégia muito útil. Por exemplo, qualquer substância pode ser dividida em partes menores, os átomos. Estudando as propriedades de cada tipo de átomo, pode-se desvendar muitas das propriedades das substâncias que deles são constituídas.

Um fenômeno semelhante ocorre com os números inteiros. Os átomos neste caso são os números *primos* e cada inteiro pode ser escrito como um produto de primos. Mas, achar a decomposição de um número inteiro em números primos pode ser um processo muito lento e laborioso.

5.3.1 Teorema da fatoração única

É conveniente, antes de mais nada, recordar a definição de um *número primo*. De acordo com [Knuth et al., 1990] um número inteiro p é primo se $p \neq \pm 1$ e os únicos divisores de p são ± 1 e $\pm p$. Portanto $2, 3, 5, -7$ são primos, mas $45 = 5 \cdot 9$ não é primo. Um número inteiro, diferente de ± 1, que não é primo é chamado de *composto*. Logo 45 é composto. Assim, todo número maior que 1 é um número primo ou é um número composto, mas nunca os dois. Logo será visto porque ± 1 não são números primos.

Teorema da fatoração única. *Dado um inteiro positivo $n \geq 2$ pode-se sempre escrevê-lo, de modo único, na forma*

$$n = p_1^{e_1} \dots p_k^{e_k},$$

onde $1 < p_1 < p_2 < p_3 < \ldots < p_k$ são números primos e $e_1, \ldots, e_k$ são inteiros positivos.

Os expoentes $e_1, \ldots, e_k$ no teorema são chamados de *multiplicidades*. Assim a multiplicidade de p_1 na fatoração de n é e_1. Em outras palavras, a multiplicidade de p_1 é o maior expoente e_1 tal que $p_1^{e_1}$ divide n. Observa-se também que n tem k fatores primos *distintos*, mas que a quantidade total de fatores primos (distintos ou não) é a soma das multiplicidades $e_1 + \ldots + e_k$.

[Coutinho, 2003] afirma que este teorema diz duas coisas. Em primeiro lugar, todo inteiro pode ser escrito como um produto de primos. Em segundo lugar, só há uma escolha possível de primos e expoentes para a fatoração de um inteiro dado.

Depois da análise do enunciado do teorema da fatoração única, fica mais fácil entender o porquê da exclusão de ± 1 da definição de números primos. Segundo [Coutinho, 2003], se isso não fosse feito, não se poderia falar da unicidade da fatoração no teorema acima. Por exemplo, se 1 fosse primo, então 2 e $1^2 \cdot 2$ seriam duas *fatorações distintas* do número 2. Usando a mesma idéia de multiplicar o número por uma potência de 1 (ou de -1) têm-se uma infinidade de fatorações distintas para cada inteiro. Para excluir este tipo de fatoração trivial, diz-se que ± 1 não são números primos.

5.4 Aritmética modular

A *aritmética modular* pode ser caracterizada como um fenômeno cíclico. Um exemplo concreto pode ser dado quando se fala de horas, pois o relógio mostra que $13 + 18$ dá 7, ou seja, se é 1 da tarde, então daqui a 18 horas serão 7 da manhã. Mas isto não acontece só com as horas, mas sim com todos os fenômenos que são cíclicos. Portanto, nessa seção será mostrado como somar, multiplicar, dividir e até mesmo resolver equações utilizando essa aritmética.

5.4.1 Relações de equivalência

[Coutinho, 2003] afirma que para se introduzir os conceitos de aritmética modular as relações de equivalência deverão ser apresentadas em primeiro lugar, pois estas irão desempenhar um papel muito importante para se compreender o assunto.

Supondo que X é um conjunto; que pode ser finito ou infinito. Assim, uma relação em X é definida em como comparar dois elementos deste conjunto. Mas, para definir a relação, é necessário dizer quem é o conjunto subjacente, ou seja, o conjunto cujos elementos estão sendo comparados.

Pode-se citar, como exemplo prático, o conjunto dos números inteiros, onde existem duas relações naturais que são as de igualdade e a de desigualdade. Outro exemplo, bastante concreto, seria adotar um conjunto de bolas coloridas, onde uma possível relação seria *bolas de uma mesma cor*.

Agora, voltando à situação geral, X é um conjunto onde está definida uma relação, que será denotada por $\sim$. Assim, esta será uma *relação de equivalência* se, quaisquer que sejam os elementos $x, y, z \in X$, as seguintes propriedades serão satisfeitas:

1. $x \sim x$.

2. Se $x \sim y \quad$ então $\quad y \sim x$.

3. Se $x \sim y \quad$ e $\quad y \sim z \quad$ então $\quad x \sim z$.

A primeira propriedade é denominada *propriedade reflexiva* e indica que se a relação é de equivalência, então um elemento pode ser comparado a si mesmo. Um exemplo é o da igualdade no conjunto dos números inteiros. A segunda propriedade chama-se *propriedade simétrica* e a terceira *propriedade transitiva*.

As relações de equivalência são usadas para classificar elementos de um conjunto em subconjuntos com propriedades semelhantes. As subdivisões de um conjunto produzidas por uma relação de equivalência são conhecidas como *classes de equivalência*. Assim, a formalidade seria: seja um conjunto X e $\sim$ uma relação de equivalência definida em X. Se $x \in X$ então a *classe de equivalência* de x é o conjunto dos elementos de X que são equivalentes a x por $\sim$. Portanto, denotando por $\overline{x}$ a classe de equivalência de x, têm-se em símbolos:

$$\overline{x} = \{y \in X : y \sim x\}.$$

Existe uma propriedade muito importante das classes de equivalência que é denotada por: *qualquer elemento de uma classe de equivalência é um representante de toda a classe*. Isto é, ao se conhecer um elemento da classe toda a classe poderá ser reconstruída. Assim, retornando ao conjunto X com sua relação de equivalência $\sim$, a propriedade afirma que se y é um elemento da classe de x então as classes de x e y são iguais, ou seja:

$$\text{se } x \in X \quad \text{e} \quad y \in \overline{x} \quad \text{então} \quad \overline{x} = \overline{y}.$$

De acordo com [Coutinho, 2003] isso pode ser provado através das propriedades das relações de equivalência. Se $y \in \overline{x}$, então, por definição, $y \sim x$; e pela simétrica, $x \sim y$. Mas se $z \in \overline{x}$ então também temos $z \sim x$. Logo pela propriedade transitiva $z \sim y$. Portanto $z \in \overline{y}$. Assim, mostrou-se que $\overline{x} \subseteq \overline{y}$, e um argumento análogo mostra que $\overline{y} \subseteq \overline{x}$

Deve-se considerar também, as seguintes propriedades do conjunto X com a relação de equivalência $\sim$:

1. X é a união de todas as classes de equivalência.

2. Duas classes de equivalência distintas não podem ter um elemento em comum.

O conjunto das classes de equivalência de $\sim$ em X é chamado de *conjunto quociente* de X por $\sim$. Portanto, o conjunto quociente não é um subconjunto de X, mas sim do conjunto das partes de X. E, consequentemente, de acordo com a propriedade citada acima, o conjunto das partes de X é aquele cujos elementos são todos os subconjuntos de X.

5.4.2 Inteiros módulo n

O conjunto dos inteiros vem com uma marco inicial natural, o número zero. Escolhendo um número inteiro positivo, e chamando esse número de n, ele será o *módulo* ou *período* da construção de uma

relação de equivalência no conjunto dos inteiros. Então, pode-se dizer que, pulando de n em n, todos os inteiros são equivalentes; ou ainda, dois inteiros cuja diferença seja um múltiplo de n são equivalentes. Assim, uma definição mais formal diria que dois inteiros a e b são *congruentes módulo* n se $a - b$ é um múltiplo de n. Se a e b são congruentes módulo n, escreve-se

$$a \equiv b \text{ (mod } n).$$

Tomando alguns exemplos numéricos e $n = 5$ como módulo, pode-se verificar que:

$$10 \equiv 0 \text{ (mod } 5) \qquad \text{e} \qquad 14 \equiv 24 \text{ (mod } 5).$$

Mas, sempre deve-se observar que quem é congruente a quem depende do módulo que foi escolhido.

Segundo [Coutinho, 2003], pode-se verificar também que a congruência módulo n é uma relação de equivalência. Primeiro, a propriedade reflexiva. Seja a um inteiro. Para mostrar que $a \equiv a$ (mod n), verifica-se, por definição, que a diferença $a - a$ é um múltiplo de n, pois 0 é múltiplo de qualquer número inteiro. Agora, analisando a propriedade simétrica, verifica-se que se $a \equiv b$ (mod n), então $a - b$ é um múltiplo de n. Mas $b - a = -(a - b)$; logo $b - a$ também é múltiplo de n. Portanto $b \equiv a$ (mod n).

Para a propriedade transitiva, supõe-se que $a \equiv b$ (mod n) e $b \equiv c$ (mod n); onde a, b e c são inteiros. A primeira congruência diz que $a - b$ é múltiplo de n; a segunda que $b - c$ é múltiplo de n. Somando múltiplos de n obtem-se de volta múltiplos de n; logo $(a - b) + (b - c) = (a - c)$ é um múltiplo de n. Assim, $a \equiv c$ (mod n). Portanto, após a verificação das três propriedades conclui-se que a congruência módulo n é uma relação de equivalência.

O conjunto que de fato importa é o conjunto quociente de $\mathbb{Z}$ pela relação de congruência módulo n. Assim, este conjunto tem uma notação própria, $\mathbb{Z}_n$; e seu nome é *conjunto dos inteiros módulo* n. Agora, para identificar os elementos de $\mathbb{Z}_n$, sabe-se por definição que são subconjuntos de $\mathbb{Z}$: as classes de equivalência da congruência módulo n. Seja $a \in \mathbb{Z}$. A classe de a é formada pelos $b \in \mathbb{Z}$ que satisfaçam $b - a$ múltiplo de n; isto é $b - a = kn$, para algum $k \in \mathbb{Z}$. Então, a classe de a pode ser descrita da forma:

$$\overline{a} = \{a + kn : k \in \mathbb{Z}\}.$$

Em particular $\overline{0}$ é o conjunto dos múltiplos de n, fazendo com que se $a \in \mathbb{Z}$, então podemos dividi-lo por n, obtendo q e r inteiros, tais que

$$a = nq + r \qquad \text{e} \qquad 0 \leq r \leq n - 1.$$

Logo $a - r = nq$ é um múltiplo de n. Portanto $a \equiv r$ (mod n), que quer dizer que o conjunto quociente $\mathbb{Z}_n$ é formado pelas classes $\overline{0}, \overline{1}, \ldots, \overline{n-1}$. Além disso, duas dessas classes não podem ser iguais. Resumindo:

$$\mathbb{Z}_n = \{\overline{0}, \overline{1}, \ldots, \overline{n-1}\}.$$

Um exemplo numérico pode ser definido da seguinte maneira:

$$\text{sendo } n = 6, \text{ temos } \mathbb{Z}_6 = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}\}$$

Supondo que se queira encontrar alguns elementos da classe $\overline{3}$, na equação $a = nq + r \quad$ e $\quad 0 \leq r \leq n - 1$ basta adotar $r = 3$ e a partir daí atribuir valores a q para que se encontre o elemento a da classe $\overline{3}$, como descrito na tabela 5.2.

Valores de q	**Valores de** a
0	3
1	9
2	15
3	21
-1	-3

Tabela 5.2: Alguns elementos da classe $\overline{3}$ para $\mathbb{Z}_6$

Portanto, alguns elementos que fazem parte da classe $\overline{3}$ para $\mathbb{Z}_6$ são:

$$\overline{3} = \{-3, 3, 9, 15, 21\}.$$

5.4.3 Aritmética modular

A definição da soma de determinadas classes de $\mathbb{Z}_n$ é dada por: Sejam $\overline{a}$ e $\overline{b}$ as classe de $\mathbb{Z}_n$ que deseja-se somar. A fórmula para a operação é a seguinte:

$$\overline{a} + \overline{b} = \overline{a + b}.$$

Mas, esta fórmula deve ser interpretada da forma correta. À esquerda se tem a soma de duas classes, e à direita a classe da soma de dois números inteiros. Agora, utilizando o exemplo em $\mathbb{Z}_6$ citado anteriormente. De acordo com essa fórmula, para somar $\overline{4}$ a $\overline{3}$ basta somar os inteiros 4 e 3, obtendo o resultado 7; logo $\overline{4} + \overline{3} = \overline{7}$. Este era o resultado esperado? Sim, pois como se tem 7 - 1 = 6, então 7 e 1 estão na mesma classe de equivalência módulo 6; ou seja, $\overline{7} = \overline{1}$ Mas, $\overline{4} = \overline{22}$ e $\overline{3} = \overline{21}$. Já que esta se somando classes, o resultado tem que dar $\overline{1}$. Então, de acordo com a fórmula obtem-se:

$$\overline{22} + \overline{21} = \overline{43}.$$

Como 43 - 1 = 42 é divisível por 6, então $\overline{43} = \overline{1}$, e assim o mesmo resultado foi obtido.

A diferença entre duas classes é definida de modo análogo ao da adição, assim não é necessária sua demonstração. E, passando à multiplicação, pode-se copiar a definição matemática dada à adição e aplicá-la também na multiplicação. Assim:

$$\overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

Assim, como nas operações em números inteiros, as operações com classes possuem propriedades correspondentes. As propriedades da adição são:

$$(\overline{a} + \overline{b}) + \overline{c} = \overline{a} + (\overline{b} + \overline{c})$$
$$\overline{a} + \overline{b} = \overline{b} + \overline{a}$$
$$\overline{a} + \overline{0} = \overline{a}$$
$$\overline{a} + \overline{-a} = \overline{0}$$

E, as propriedades multiplicativas são:

$$(\overline{a} \cdot \overline{b}) \cdot \overline{c} = \overline{a} \cdot (\overline{b} \cdot \overline{c})$$
$$\overline{a} \cdot \overline{b} = \overline{b} \cdot \overline{a}$$
$$\overline{a} \cdot \overline{1} = \overline{a}$$

Há também a propriedade que relaciona duas operações, a *distributividade*:

$$\overline{a} \cdot (\overline{b} + \overline{c}) = \overline{a} \cdot \overline{b} + \overline{a} \cdot \overline{c}$$

Um exemplo em $\mathbb{Z}_6$ para a multiplicação seria:

$$\overline{2} \cdot \overline{3} = \overline{6} = \overline{0}$$

A divisão em $\mathbb{Z}_n$ possuiu uma aritmética mais complexa, portanto será descrita a seguir com mais cuidado.

5.4.4 Divisão Modular

Supõe-se que a e b são números reais, e se quer 'dividir a por b'. Isto é o mesmo que 'multiplicar a por $1/b$'.Assim, o número real $1/b$ é conhecido como o *inverso* de b, que é caracterizado pela equação $b \cdot 1/b = 1$. De acordo com [Coutinho, 2003], trabalhar com o inverso ao invés da divisão diretamente tem uma vantagem conceitual, que será vista logo. Mas, observa-se o fato de que $b \neq 0$ para as afirmações acima terem sentido. Agora, transpõe-se tudo para $\mathbb{Z}_n$.

Supondo $\overline{a} \in \mathbb{Z}_n$, e que $\overline{\alpha} \in \mathbb{Z}_n$ seja seu inverso, a equação $\overline{a} \cdot \overline{\alpha} = \overline{1}$ deve ser verificada em $\mathbb{Z}_n$. Se $\overline{a} = \overline{0}$, então $\overline{a}$ não possuiu inverso. Mas, em $\mathbb{Z}_n$ pode haver outros elementos sem inverso além dos pertencentes à classe $\overline{0}$.

Se $\overline{a} \in \mathbb{Z}_n$ tem inverso $\overline{\alpha}$, pode-se dizer que a equação

$$\overline{a} \cdot \overline{\alpha} = \overline{1}$$

corresponde a dizer que $a\alpha - 1$ é divísivel por n. Isto é

$$a\alpha + kn = 1$$

para algum inteiro k. Assim, verfica-se que mdc[1](a, n) = 1. Assim, pode-se dizer que se $\overline{a}$ tem inverso em $\mathbb{Z}_n$ então mdc(a, n) = 1. Mas, o contrário também é verdadeiro?

Supõe-se um inteiro a, e que mdc(a, n) = 1. Assim, se for aplicado o algoritmo euclidiano estendido tal que

$$a\alpha + n\beta = 1.$$

Pode-se verificar que esta equação é equivalente a

$$\overline{a} \cdot \overline{\alpha} = \overline{1}$$

em $\mathbb{Z}_n$. Logo a classe $\overline{\alpha}$ calculada pelo algoritmo euclidiano estendido é o inverso de a em $\mathbb{Z}_n$. Assim, conclui-se que se mdc(a, n) = 1 então $\overline{a}$ tem iverso em $\mathbb{Z}_n$.

Teorema da inversão. *A classe $\overline{a}$ tem inverso em $\mathbb{Z}_n$ se, somente se, a e n são primos entre si.*

Verifica-se que o melhor a fazer é utilizar o algoritmo euclidiano estendido, pois este descobre se o inverso realmente *existe* e *qual é* ao mesmo tempo. Agora, um exemplo numérico: qual é o inverso de $\overline{3}$ em $\mathbb{Z}_{32}$?

De início, verifica-se que mdc$(3, 32)$ = 1, então o inverso existe. Agora, aplicando o algoritmo estendido, temos

$$3 \cdot 11 + 32 \cdot (-1) = 1,$$

que é equivalente a $\overline{3} \cdot \overline{11} = \overline{1}$ em $\mathbb{Z}_{32}$. Logo, $\overline{11}$ é o inverso de $\overline{3}$ em $\mathbb{Z}_{32}$.

O conjunto dos elementos em $\mathbb{Z}_n$ que possuem inverso é muito importante, por isso deve ser definido. Ele vai ser denotado por $\mathcal{U}(n)$. Assim:

$$\mathcal{U}(n) = \{\overline{a} \in \mathbb{Z}_n : \text{mdc}(a, n) = 1\}.$$

[Coutinho, 2003] afirma que é fácil calcular $\mathcal{U}(p)$ quando p é um número primo. Neste caso, mdc(a, p) = 1 significa que p não divide a. Mas se p divide a então $\overline{a} = \overline{0}$. Portanto, quando p é primo, todas as classes diferentes de $\overline{0}$ tem inversão. Assim

$$\mathcal{U}(n) = \mathbb{Z}_p \setminus \{\overline{0}\}$$

Mas isso só vale *se p for primo*. O que acontece quando n é composto, é mostrado em alguns exemplos:

$$\mathcal{U}(4) = \{\overline{1}, \overline{3}\} \qquad \text{e} \qquad \mathcal{U}(8) = \{\overline{1}, \overline{3}, \overline{5}, \overline{7}\}$$

[1]Máximo divisor comum

Uma propriedade importante do conjunto $\mathcal{U}(n)$ é que o produto de dois elementos de $\mathcal{U}(n)$ é um elemento de $\mathcal{U}(n)$. Em outras palavras, se $\overline{a}$ e $\overline{b}$ em $\mathbb{Z}_n$ têm inverso, então $\overline{a} \cdot \overline{b}$ também tem inverso em $\mathbb{Z}_n$. Supondo $\overline{\alpha}$ o inverso de $\overline{a}$ e $\overline{\beta}$ o inverso de $\overline{b}$, o inverso de $\overline{a} \cdot \overline{b}$ será $\overline{-\alpha} \cdot \overline{\beta}$. A verificação é:

$$(\overline{a} \cdot \overline{b})(\overline{\alpha} \cdot \overline{\beta}) = (\overline{a} \cdot \overline{\alpha})(\overline{b} \cdot \overline{\beta}) = \overline{1} \cdot \overline{1} = \overline{1}.$$

Agora, voltando à questão de executar divisões em $\mathbb{Z}_n$, se é desejado dividir $\overline{a}$ por $\overline{b}$ precisa-se saber se $\overline{b}$ está em $\mathcal{U}(n)$. Caso não esteja, a divisão não é possível. Se estiver, calcula-se o inverso de $\overline{b}$ em $\mathbb{Z}_n$, supondo que seja $\overline{\beta}$, e divide-se $\overline{a}$ por $\overline{b}$ calculando o produto $\overline{a} \cdot \overline{\beta}$. Um exemplo: calcularа divisão de $\overline{2}$ por $\overline{3}$ em $\mathbb{Z}_8$. Como mdc(3,8) = 1 então $\overline{3}$ tem inverso em $\mathbb{Z}_8$. Calculando através do algoritmo estendido, encontra-se que o inverso de $\overline{3}$ é o próprio $\overline{3}$. Assim, o resultado da divisão de $\overline{2}$ por $\overline{3}$ em $\mathbb{Z}_8$ é $\overline{6}$. Observa-se que o fato de $\overline{2}$ não ter inverso em $\mathbb{Z}_8$ não tem nenhuma importância, já que $\overline{2}$ é o dividendo.

Pode-se usar o que já foi estudado até aqui para resolver congruências lineares em $\mathbb{Z}_n$. Uma *congruência linear* é uma equação do tipo

$$ax \equiv b \text{ (mod } n),$$

onde $a, b \in \mathbb{Z}$. Para resolvê-la, pode dividir por a para que o x fique livre do lado esquerdo da equação. Isto só é possível se mdc(n, a) = 1. Se isso for satisfeito, então existe $\alpha \in \mathbb{Z}$ tal que $a\alpha \equiv 1$ (mod n). Multiplicando a equação por α, obtemos $ax\alpha \equiv b\alpha$ (mod n). E, como α é o inverso de $\overline{a}$ em $\mathbb{Z}_n$ esta equação se reduz a

$$x \equiv b\alpha \text{ (mod } n),$$

E, um exemplo numérico seria: resolver a congruência $7x \equiv 3$ (mod 15). O inverso de $\overline{7}$ em $\mathbb{Z}_{15}$ é $\overline{-2}$ = $\overline{13}$. Assim, tem-se

$$x \equiv 13 \cdot 3 \equiv 39 \equiv 9 \text{ (mod } 15),$$

que é a solução da equação.

5.5 Sistemas de congruência

Essa seção foi baseada em [Coutinho, 2003, capítulo 7], onde também encontram-se exemplos adicionais do assunto que será discutido, que é a resolução de equações lineares.

Considerando o caso de uma única equação linear

$$a \equiv b \text{ mod}(n),$$

(1.1)

onde n é um inteiro positivo. Já foi visto que estas equações são muito fáceis de se resolver quando $\overline{a}$ é inversível em $\mathbb{Z}_n$. Mas se $\overline{a}$ *não* tem inverso em $\mathbb{Z}_n$, o problema se torna mais complicado. Foi visto que se $\overline{a}$ não tem inverso, pode-se dizer que mdc$(a, n) \neq 1$. Assim, se a equação (1.1) tem solução, isto quer dizer que existem $x, y \in \mathbb{Z}$ tais que

$$ax - ny = b \tag{1.2}$$

Mas isto só é possível se o mdc(a, n) divide b. Primeira conclusão: (1.1) *só tem solução quando* b *é divisível pelo mdc*(a, n). Se $\overline{a}$ tem inverso em $\mathbb{Z}_n$ esta condição é satisfeita, porque neste caso mdc(a, n) = 1.

Supõe-se então que $d =$ mdc(a, n) divide b. Diz-se que $a = da´$, $b = db´$ e $n = dn´$. Substituindo em (1.2) e cancelando d

$$a´x - n´y = b´.$$

que se converte na equação $a´x \equiv b´$ mod$(n´)$; uma nova equação em congruências. Mas, deve-se tomar cuidado, pois o módulo mudou. A equação original tinha n como módulo, e a nova tem $n´$, que é um divisor de n. Deve-se observar que mdc$(a´n´)$ é sempre 1.

Agora, um exemplo numérico: seja $6x \equiv 4$ (mod 8). Como mdc(6,8) = $2 \neq 1$, $\overline{6}$ não é inversível em $\mathbb{Z}_8$. Assim, se esta equação tiver solução, então existem inteiros x e y tal que $6x - 8y = 4$. Dividindo toda equação por 2, obtem-se $3x - 4y = 2$, que leva a $3x \equiv 2$ mod(4). Mas $\overline{3}$ é seu próprio inverso em $\mathbb{Z}_4$, então deve-se multiplicar essa ultima equação por 3 para se obter

$$x \equiv 2 \text{ mod}(4). \tag{1.3}$$

Mas, isso é um pouco insatisfatório, pois a equação inicial era módulo 8, e a obtida foi módulo 4. Para corrigir este problema, basta converter (1.3) em uma expressão de inteiros(isto é, sem congruências) e ver como as soluções se comportam módulo 8. Tem-se que (1.3) é equivalente a $x = 2 + 4k$, onde k é um inteiro qualquer. Há duas possibilidades para k. Se for par, então $x \equiv 2$ mod(8) é uma das soluções. Se k for ímpar, então $k = 2m + 1$ e $x = 6 + 8m$. Isto é $x \equiv 6$ mod(8), que é uma outra solução. Portanto, a equação $6x \equiv 4$ mod(8) tem duas soluções em $\mathbb{Z}_8$ que são $\overline{2}$ e $\overline{6}$. Deve-se observar que a equação é linear, mas possuiu duas soluções.

5.5.1 Algoritmo chinês do resto

Considere o sistema

$$x \equiv a \text{ mod(m)}$$
$$x \equiv b \text{ mod(n).}$$

Como já mencionado anteriormente, a primeira equação pode ser reescrita na forma $x = a + my$, onde y é um inteiro qualquer. Com isto pode-se substituir x na segunda equação, obtendo $a + my \equiv b$ (mod n), ou ainda

$$my \equiv (b - a) \text{ (mod n).}$$

Já se sabe como resolver essa equação. Sabe-se também que pode não ter solução. Para que tenha solução é preciso que o máximo divisor comum entre m e n divida $b - a$.

Para garantir que isso vai acontecer basta assumir que mdc(n, m) = 1. Com isto $\overline{m}$ tem inverso em $\mathbb{Z}_n$. Chmando $\overline{\alpha} \in \mathbb{Z}_n$ o inverso, a solução da equação acima é $y \equiv \alpha(b-a)$ (mod n). Isto é $y = \alpha(b-a)+nz$, onde z é um número inteiro. Substituindo na equação que dá x em função de y, tem-se

$$x = a + m\alpha(b - a) + mnz.$$

Mas $\overline{m\alpha} = \overline{1}$ em $\mathbb{Z}_n$. Logo, existe algum inteiro β tal que $1 - m\alpha = n\beta$. Assim

$$x = a(1 - m\alpha) + mb\alpha + mnz = an\beta + mb\alpha + mnz.$$

Esta maneira de escrever a solução tem uma vantagem: os inteiros α e β são obtidos facilmente. De fato, $1 = m\alpha + n\beta$. Como supõe-se que mdc(m, n) = 1, basta aplicar o algoritmo euclidiando estendido a m e n para obter α e β. Resumindo: se mdc(m, n) = 1, então o sistema mostrado acima sempre tem como soluções os números $an\beta + bm\alpha + kmn$, onde k é um inteiro qualquer.

Já foi visto que uma equação linear pode ter mais de uma solução se o módulo for composto. Quantas soluções tem o sistema acima? Infinitas, se pensa-se em soluções inteiras, já que há uma solução para cada escolha de k. Mas, diz-se que x e y são dois inteiros que são solução do sistema descrito acima. Então, tem-se que $x \equiv a$ mod(m) e $y \equiv a$ mod(m). Como se tratam de duas equações *com mesmo módulo* pode-se realizar a subtração entre elas. Obtem-se $x - y \equiv 0$ mod(m). Isto é, m divide $x - y$. Fazendo o mesmo com a segunda equação, concluí-se que n divide $x - y$. Supondo que mdc(m, n) = 1, tem-se que mn divide $x - y$. Assim, o sistema tem infinitas soluções inteiras, mas apenas uma solução em $\mathbb{Z}_{mn}$ quando mdc(m,n) = 1. Portanto, tudo isso pode ser resumido em um teorema.

Teorema Chinês do Resto.*Sejam m e n inteiros positivos, primos entre si. O sistema*

$$x \equiv a \text{ mod(m)}$$
$$x \equiv b \text{ mod(n)}$$

sempre tem uma única solução em $\mathbb{Z}_{mn}$.

Capítulo 6

O MÉTODO DA PARTILHA DE SENHAS

6.1 Considerações iniciais

Este capítulo trata do tema central do trabalho, que é denominado de partilha de senhas. Primeiramente será apresentado o problema central e posteriormente alguns métodos adotados para sua resolução serão discutidos. Esses métodos são essenciais para a compreensão do trabalho implementado.

6.2 Apresentação do problema

A partilha de senhas é um método que foi inventado por Adi Shamir e George Blakley independentemente, no ano de 1979. Esse método consiste em distribuir um segredo entre um grupo de participantes, onde cada um terá uma parte desse segredo. Sendo que este segredo será reconstruído com a junção de algumas das partes de cada um, fazendo com que um indivíduo não tenha como utilizar esse segredo em benefício próprio.

[Zhou, 2002] descreve um exemplo bem simples e de fácil compreensão, onde deve-se supor um mecanismo de lançamento de mísseis nucleares. Esse mecanismo é controlado por uma senha, e está sob o controle de três generais do exército. Uma solução simples pode ser distribuir a senha para cada um desses generais. Mas pode-se imaginar que apenas um dos generais pode iniciar uma guerra apenas inserindo a senha no mecanismo. Então, usa-se o método da partilha de senhas proposto acima, fazendo com que cada general possua uma parte da senha e um míssel não poderá ser disparado enquanto os três não concordarem em fazê-lo. Mas, por outro lado, pode-se pensar que um dos generais seja um espião inimigo, e assim não permitirá o lançamento de mísseis, usando para isso uma parte da senha que não é a verdadeira, fazendo com que o mecanismo nunca realize o lançamento. Pode-se imaginar também a situação em que um dos generais pode estar morto ou desaparecido, o que acarreta na eliminação de uma parte da senha, impossibilitando o uso dos mísseis nucleares. Então, um número mínimo de generais deve ser necessário, para que o mecanismo possa funcionar mesmo sem a presença de todas as senhas.

[Coutinho, 2003] cita um outro exemplo, onde um cofre de uma determinada agência bancária só pode ser aberto mediante uma senha, e que somente um certo número de funcionários têm acesso ao cofre. Mas o banco deseja, por segurança, que o cofre não seja aberto quando estão presentes na agência menos que

oito destes funcionários que têm acesso ao cofre. Supõe-se que há na agência treze funcionários com acesso ao cofre. Agora, como garantir que não será possível abrir o cofre a menos que haja, pelo menos, oito destes funcionários presentes na agência?

Para o caso geral é necessário conhecer a senha, que é um número s, e o que se deseja é partilhar s entre n pessoas (número de funcionários que tem acesso ao cofre). Assim, [Shamir, 1979] apresenta o verdadeiro conceito do método de partilha de senhas, onde se tem um determinado dado D que será dividido em n partes e, desse modo, D poderá ser reconstruído a partir de **quaisquer** k **partes**, mas nunca com um número menor ou igual a $k-1$ partes.

Assim, [Zhou, 2002] mostra duas propriedades distintas:

- **Avaliabilidade:** Onde um número maior ou igual de k partes pode reconstruir o dado D;
- **Confidencialidade:** Significa que um número menor que k partes não pode reconstruir D.

Agora, um esquema de partilha pode ser designado, e para isso [Zhou, 2002] utiliza uma geometria simples para explicar o esquema proposto por [Shamir, 1979]. Inicialmente, será mostrado um esquema de um grupo que possui n participantes e apenas 2 membros quaisquer deste grupo podem reconstruir a senha inicial s. A notação usada para este esquema é $(2, n)$.

De acordo com a figura 6.1, o ponto $(0, s)$ no eixo Y corresponde ao segredo (ou senha) s a ser compartilhado. Agora, uma reta que corta esse ponto deve ser desenhada aleatóriamente. Encontrando n pontos nessa reta têm-se

$$(x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$$

Cada ponto passa a representar uma parte da senha, e estes são distribuídos entre cada um dos n participantes. Assim, já se obtém um esquema $(2, n)$.

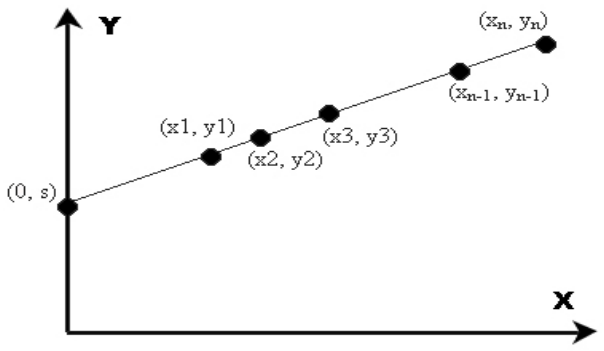

Figura 6.1: Esquema $(2, n)$ de partilha

Agora, resta mostrar como este esquema satisfaz as propriedades citadas anteriormente.

Para mostrar a avaliabilidade, deve-se provar apenas que 2 partes podem recuperar a senha original. Portanto, 2 pessoas têm duas partes da senha, que são dois pontos da reta. Dados estes dois pontos, como

recuperar a senha original? Sabe-se que dois pontos são suficientes para se determinar uma reta; então esta pode ser traçada por entre esses dois pontos. E, conhecida a reta, também é conhecida a interseção desta com o eixo Y, e essa interseção é a senha original determinada anteriormente.

Para a confidencialidade pode-se usar a mesma resolução, pois deve-se provar que apenas 1 parte não pode recuperar a senha original. Portanto, quando se conhece apenas um ponto, várias retas com diferentes inclinações podem ser traçadas por ele. Essas retas interceptam o eixo Y em diferentes pontos, e dão várias possibilidades de diferentes senhas. Então, com apenas um ponto nenhuma informação sobre a senha inicial será exposta.

Seguindo esta mesma idéia, um outro esquema pode ser designado. Trata-se de um esquema de um grupo que possui n participantes e agora apenas 3 membros quaisquer deste grupo podem reconstruir a senha inicial s. Esse esquema é denotado de $(3, n)$.

O esquema anterior usava uma reta para delimitar os dois pontos necessários para se reconstruir a senha. Mas, como agora serão necessários 3 pontos, uma reta não poderá ser usada. Então, uma solução para se delimitar esses pontos é usar uma parábola, sendo que essa parábola é representada por uma função quadrada do tipo $y = a_2x^2 + a_1x + a_0$. E, de novo, marca-se um ponto no eixo Y que representa a senha inicial. Em seguida traça-se uma parábola qualquer que passe pelo ponto inicial escolhido para ser a senha original, e, finalmente escolhe-se n pontos para representar as n partes da senha que serão distribuídas, como mostra a figura 6.2.

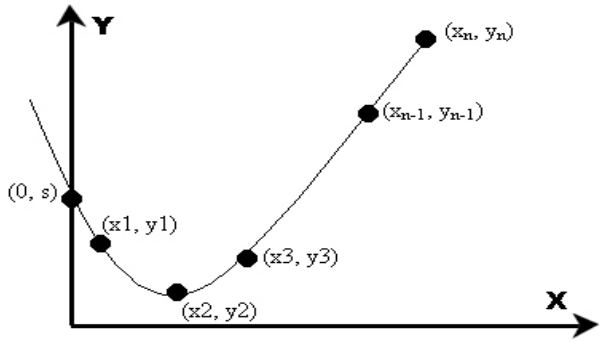

Figura 6.2: Esquema $(3, n)$ de partilha

Para provar que este esquema satizfaz as propriedades basta que se use uma explicação semelhante ao do esquema $(2, n)$.

Generalizando estes exemplos, têm-se uma designação para qualquer esquema na forma (k, n). E para isso usam-se polinômios de grau $k - 1$:

$$y = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1x + a_0$$

Assim, pode-se selecionar qualquer polinômio de grau n que intercepte o ponto $(0, s)$, que representa a senha inicial. Depois n pontos são escolhidos nesta parábola, e usando os mesmos argumentos acima

pode-se provar que o esquema satisfaz ambas as propriedades de avaliabilidade e confidencialidade. Como já foi citado anteriormente, este esquema foi proposto por [Shamir, 1979], onde o esquema original foi definido em um campo finito.

6.3 Algoritmos de partilha de senhas

Agora, serão apresentados alguns métodos que realizam a partilha de uma senha s qualquer entre um determinado grupo de n indivíduos.

6.3.1 Polinômio interpolador de Lagrange

[Shamir, 1979] usa equações polinômiais sobre um campo finito para construir esquemas de partilha de senhas. [Schneier, 1996] instrui a escolha de um número primo, p, que seja maior que o número de possíveis partes da senha e que também seja maior que a senha inicial que se deseja partilhar. Para realizar a partilha da senha, gera-se um polinômio arbitrário de grau $k-1$. Por exemplo, caso o esquema seja $(3, n)$ (onde três partes serão necessárias para reconstruir a senha original M), deve-se obter um polinômio de grau 2, ou seja, um polinômio quadrático

$$F(x) = (ax^2 + bx + M) \bmod p$$

onde p é um número primo aleatório maior que qualquer um dos coeficientes. Os coeficientes a e b também são números randômicos, mas são mantidos em segredo e descartados logo após que se obtêm as partes da senha original. O número primo p deverá ser divulgado.

As partes serão obtidas pela avaliação do polinômio nos n diferentes pontos:

$$k_i = F(x_i).$$

Portanto, a primeira parte poderia ser obtida ao se avaliar o polinômio no ponto onde $x = 1$, a segunda parte no ponto onde $x = 2$, e assim sucessivamente.

Assim, um polinômio quadrático possui três coeficientes desconhecidos,a, b e M, e quaisquer três partes que foram geradas anteriormente podem ser usadas para se criar três equações, formando assim um sistema que poderá ser facilmente resolvido. Portanto, duas partes não podem fazer isso, e uma parte também não. Agora, quatro ou cinco partes já se tornam redundantes.

[Schneier, 1996] apresenta um exemplo simples, onde adota-se $M = 11$(que é a senha a ser partilhada) e o esquema é da forma $(3, 5)$, que signifa que qualquer combinação de três pessoas de um grupo de cinco podem recuperar o valor de M. A equação obtida foi:

$$F(x) = (7x^2 + 8x + 11) \bmod 13.$$

Como $k = 3$, deve-se gerar um polinômio quadrático, e os números 7 e 8 foram escolhidos randomicamente, e representam os valores de a e b respectivamente. O número primo 13 também foi gerado randomicamente, mas deve atender ao critério de ser maior que qualquer um dos coeficientes.

Avaliando a função dada em n diferentes pontos, obtem-se as seguintes partes da senha:

$k_1 = F(1) = 7 + 8 + 11 \equiv 0$ mod 13;
$k_2 = F(2) = 28 + 16 + 11 \equiv 3$ mod 13;
$k_3 = F(3) = 63 + 24 + 11 \equiv 7$ mod 13;
$k_4 = F(4) = 112 + 32 + 11 \equiv 12$ mod 13;
$k_5 = F(5) = 175 + 40 + 11 \equiv 5$ mod 13;

Para reconstruir M através de quaisquer três partes obtidas acima, como por exemplo, k_2, k_3 e k_5, basta resolver o sistema de equações lineares:

$$a * 2^2 + b * 2 + M \equiv 3 \text{ mod } 13;$$
$$a * 3^2 + b * 3 + M \equiv 7 \text{ mod } 13;$$
$$a * 5^2 + b * 5 + M \equiv 5 \text{ mod } 13;$$

Assim, encontram-se as soluções $a = 7$, $b = 8$ e $M = 11$. Portanto, M foi reconstruído corretamente.

6.3.2 Esquema dos planos

[Blakley, 1979] citado por [Schneier, 1996], inventou um esquema de partilha usando pontos no espaço. O segredo, ou senha, é definido como um ponto em um espaço *m-dimensional*. Cada parte da senha é uma equação de um hiperplano *(m-1)-dimensional* que inclui o ponto definido inicialmente. A intercessão de quaisquer m hiperplanos determina o ponto exato.

O hiperplano citado é um conjunto de soluções $x = (x_1, \ldots, x_n)$ de uma equação na forma $a_1x_1 + \ldots + a_nx_n = b$, e [RSA, 2000] mostra um exemplo em que duas partes da senha são necessárias para se obter o segredo inicial, o que significa que um plano bi-dimensional será usado. Como pode ser visto na figura 6.3, S é o ponto do plano que representa o segredo a ser compartilhado, e cada parte desse segredo é uma linha que passa por esse ponto. Se duas linhas são inseridas juntas, a intercessão destas irá revelar o ponto S, que é o segredo inicial.

Mas, [Buholzer, 1999] afirma que este algoritmo não é perfeito, pois uma pessoa com uma parte do segredo inicial sabe que o ponto S está presente no seu hiperplano.

6.3.3 Karnin-Greene-Hellman

Este tipo de esquema de partilha utiliza multiplicação de matrizes e foi idealizado por [Karnin et al., 1983]. Segundo [Schneier, 1996], que supõe um esquema (m, n), deve-se escolher $n+1$ vetores de dimensão m,

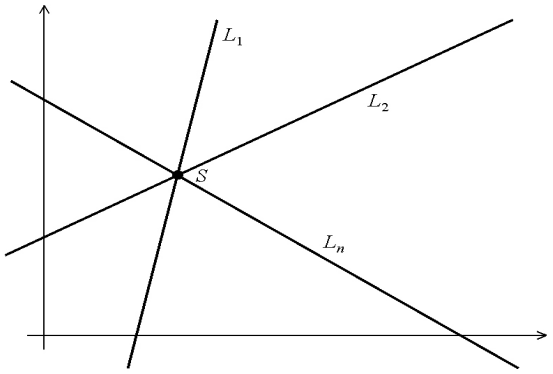

Figura 6.3: Esquema de Blakley

que são $V_0, V_1, V_2, \ldots, V_n$, tais que quaisquer $m * m$ possíveis matrizes formadas a partir desses vetores tenham ordem n. Existe também um vetor U, que é um vetor de linha, de dimensão $m + 1$.

A matriz M, que é o segredo a ser compartilhado, é obtida através do produto entre U e V_0, assim como as partes da senha são os produtos entre U e V_i, onde i é um número que varia de 1 a n.

Quaisquer m partes podem ser usadas para resolver um sistema $m * m$ de equações lineares, onde as incógnitas são os coeficientes de U. Assim, $UV_0 = M$ pode ser computado a partir de U.

De acordo com [Schneier, 1996], quaisquer $m - 1$ partes não podem resolver o sistema de equações linear e, portanto, não podem recuperar a matriz M, que possui o segredo inicial.

6.3.4 Aplicação do algoritmo chinês do resto

O método descrito em [Coutinho, 2003] mostra que para cada pessoa vai ser dado um elemento (sua "parte" da senha s) de um conjunto $\mathbb{S}$ de n *pares* de inteiros positivos, de modo que, para um inteiro positivo $k \leq n$, previamente escolhido têm-se que:

1. qualquer subconjunto de $\mathbb{S}$ com k elementos permite determinar s facilmente;

2. é muito difícil determinar s conhecendo menos de k elementos de $\mathbb{S}$.

A inspiração para a construção do conjunto $\mathbb{S}$ vem do teorema chinês do resto. Inicialmente escolhe-se um conjunto $\pounds$ de n inteiros positivos, dois a dois primos entre si. Seja N o produto dos k *menores* números de $\pounds$ e M o produto dos $k - 1$ *maiores* números de $\pounds$. Diz-se que este conjunto tem limiar k se

$$N > s > M.$$

Observe que esta condição implica que o produto de k ou mais elementos de $\pounds$ é sempre *maior* que N e o produto de menos de k elementos é sempre *menor* que M. O conjunto $\mathbb{S}$ será formado pelos pares da forma (m, s_m) onde $m \in \pounds$ e s_m é a forma reduzida de s módulo m. Note que o fato de se ter um conjunto com limiar $k \geq 1$ implica que $s > m$, para qualquer $m \in \pounds$. Portanto sempre se tem $s_m < s$, qualquer que seja $m \in \pounds$.

Supõe-se que mais de k funcionários se encontram no banco. Isto equivale a dizer que são conhecidos t dentre os pares de $\mathbb{S}$, onde $t \geq k$. Denota-se estes pares por $(m_1, s_1), \ldots, (m_t, s_t)$. Resolvendo o sistema de congruências

$$\begin{aligned} x &\equiv s_1 \pmod{m_1} \\ x &\equiv s_2 \pmod{m_2} \\ &\vdots \\ x &\equiv s_t \pmod{m_t} \end{aligned} \tag{1}$$

obtem-se x_0 como solução. De acordo com o teorema chinês do resto,

$$x \equiv s_1 \pmod{m_1 \ldots m_t}.$$

Mas será que são iguais? Nesse momento que se observa o fato da sequência de módulos ter limiar k. Sabe-se que, como $t \geq k$,

$$m_1 \ldots m_t \geq N > s.$$

Pelo teorema chinês do resto, o sistema (1) tem uma *única* solução menor que $m_1 \ldots m_t$. Mas s também é solução do sistema e $s < m_1 \ldots m_t$. Logo $s = x_0$.

Nada impede de resolver um sistema semelhante para o caso em que $t < k$. O problema é que o produto de menos de k módulos de $\pounds$ é sempre menor que s. Assim a solução do sistema é um número congruente a s, mas não pode ser igual a s. É claro que sempre é possível achar s fazendo uma busca. De fato, sabe-se que $M < s < N$ e que s satisfaz o sistema (1), só que agora com $t < k$. Diz-se então que foi encontrada uma solução x_0 do sistema. Como $x_0 < M < s$, não foi encontrada s. Mas o sistema (1) também é satisfeito por s, logo

$$s = x_0 + y.(m_1 \ldots m_t)$$

onde y é um inteiro positivo. Como

$$N > s > M > x_0.$$

têm-se que

$$\frac{M - x_0}{m_1 \ldots m_t} \leq y = \frac{s - x_0}{m_1 \ldots m_t} \leq \frac{N - x_0}{m_1 \ldots m_t}.$$

Isto significa que precisa ser feita uma busca para achar o valor correto de y entre, pelo menos,

$$d = \left[\frac{N - M}{m_1 \ldots m_t} \right]$$

inteiros. Escolhendo os módulos de modo que d seja muito grande, fica praticamente impossível encontrar s através de uma busca.

Na prática os dados iniciais do problema são o número total de funcionários da agência e o número mínimo de funcionários que têm que estar presentes para que o cofre possa ser aberto. O primeiro determina quantos elementos o conjunto $\pounds$ tem que ter. O segundo determina qual é o limiar k de $\pounds$. Com estes dados, escolhemos um conjunto $\pounds$ de limiar k. Com isto pode-se calcular M e N como acima. Só agora s será escolhido (de maneira aleatória) no intervalo entre M e N. Chegados a esse ponto, já temos todos os dados necessários para calcular $\mathbb{S}$, que nos diz quem são as senhas a serem distribuídas. A segurança do sistema se baseia no fato de que, quanto maior o valor de k, mais improvável a existência de k funcionários desonestos no banco.

[Coutinho, 2003] mostra um exemplo em que há apenas 5 funcionários e que pelo menos 2 têm que estar presentes para que o cofre possa ser aberto. Logo o conjunto $\pounds$ deve ter 5 elementos e seu limiar deve ser 2. Uma escolha possível, usando apenas primos pequenos, é:

$$\pounds = \{11, 13, 17, 19, 23\}.$$

De fato, o produto dos 2 menores inteiros no conjunto é $N = 11.13 = 143$. Por outro lado, M é o produto dos $k - 1$ maiores elementos de $\pounds$. Como $k = 2$, temos que M é igual ao maior elemento de $\pounds$; logo $M = 23$. Portanto $\pounds$ tem realmente limiar 2. O valor de s pode ser escolhido como sendo qualquer inteiro no intervalo que vai de 23 a 143. Supõe-se que $s = 30$. Então

$$\mathbb{S} = \{(11, 19), (13, 17), (17, 13), (19, 11), (23, 7)\}.$$

Finalmente, o que acontece se os funcionários que têm senhas (17,13) e (23,7) estão no banco? Para obter a senha s do cofre é preciso resolver o sistema

$$x \equiv 13 \text{ (mod } 17)$$
$$x \equiv 7 \text{ (mod } 23)$$

A solução geral do sistema é $x = 30 + 391k$, onde k é um inteiro positivo. Isto é $x \equiv 30$ (mod 391). Assim foi determinado que $s = 30$; que é o valor correto.

Capítulo 7

RESULTADOS E DISCUSSÕES

7.1 Considerações iniciais

Como resultado da pesquisa realizada, foi desenvolvido o *Easyshare: Sistema de proteção de pastas e arquivos distribuídos utilizando partilha de senhas.* Nas seções a seguir são descritas as principais características e funcionalidades do *Easyshare*, dentre elas bloquear e desbloquear arquivos e pastas, controle de usuários, gerência das chaves pelos usuários e formas de utilização. E, ao final, apresenta-se a modelagem do sistema na notação UML (*Unified Modeling Language*)[Ahmed e Umrysh, 2002] e também as modelagens do banco de dados e suas tabelas.

7.2 Apresentando o *EasyShare*

O *EasyShare* é um sistema que visa proteger informações contidas em pastas e arquivos que estejam distribuídos em uma determinada rede de computadores. E, para realizar tal proteção, o *software* utiliza a partilha de senhas para gerar uma senha inicial s e usa-la para criptografar o contéudo de um arquivo ou uma pasta do computador do usuário e em seguida as partes dessa senha inicial são distribuídas entre usuários cadastrados no sistema. Posteriormente essa senha inicial s é descartada, e para poder recuperá-la os usuários deverão compartilhar suas senhas. Assim, por se tratar de um sistema que atende à vários requisitos para implementar a solução da partilha de senhas e o compartilhamento de informações entre diferentes usuários atuando em diferentes computadores, essa seção se destina a apresentar as principais características de como o produto foi implementado e desenvolvido.

7.2.1 Características

O sistema foi desenvolvido utilizando a linguagem de programação *Java* [Microsystems, 2005] e utiliza o banco de dados *MySQL* [MySQL, 2005] para armazenar as informações centralizadas, assim, o funcionamento do *EasyShare* é praticamente independente do sistem operacional utilizado, pois tanto o Java quanto o MySQL atuam em diferentes plataformas. Os testes foram realizados em plataformas *Windows* XP, NT e 2000, mas nada de anormal foi detectado quando o sistema foi executado sobre *Linux* e *Windows* 9X. Não foram realizados testes em plataformas *Mac OS X* e *Solaris*, embora isso seja totalmente

possível, bastanto apenas que o usuário tenha instalada no computador a máquina virtual *Java*, ou JVM (*Java Virtual Machine*) [Microsystems, 2005], de versão 1.4.2 ou superior e o servidor de banco de dados *MySQL*. Portanto, o *EasyShare* é classificado como um *software* multiplaforma.

O desenvolvimento do *EasyShare* foi realizado inteiramente sobre a IDE[1] Eclipse [Foundation, 2005], sendo esta uma ferramenta de código aberto. A partir daí decidiu-se que o *EasyShare* também fosse um produto de código aberto, pois é uma ferramenta com grandes utilidades para fins acadêmicos, e até mesmo para que outras pessoas realizem melhorias, desde que se mantenha as referências dos autores originais. Assim, o produto é disponibilizado sob a licença de uso GPL (*General Public GNU Licence*) [FSF, 2005].

Este é um sistema projetado para funcionar em uma rede que possui vários computadores interligados (normalmente uma LAN[2]), e cada computador irá executar o *software* que estará conectado a um servidor com o SGBD[3] MySQL instalado. Isso requer que seja utilizada uma arquitetura que seja responsável por gerenciar os pedidos de conexão e o controle de pacotes entre as várias instâncias do *software*.

A arquitetura utilizada é do tipo cliente-servidor, pois apresenta uma separação distinta entre servidores (aplicações que disponibilizam algum serviço) e clientes (aplicações que utilizam os serviços oferecidos pelos servidores). [Tanenbaum, 2003] mostra que está foi uma arquitetura de comunicação predominante na década de 90, pois apresenta um aumento da confiabilidade (a falha de uma máquina não necessariamente inviabiliza a operação do sistema como um todo) e redução de custo (máquinas mais simples podem executar os serviços isoladamente, ao invés de ter uma grande máquina fazendo todos os serviços). Assim, verificou-se que essa arquitetura seria suficiente para que o *EasyShare* realize a troca de informações necessárias entre os hosts da rede. A figura 7.1 apresenta a arquitetura cliente-servidor utilizada pelo *EasyShare*, pois cada instância da aplicação atua como um cliente e como um servidor, como será observado posteriormente.

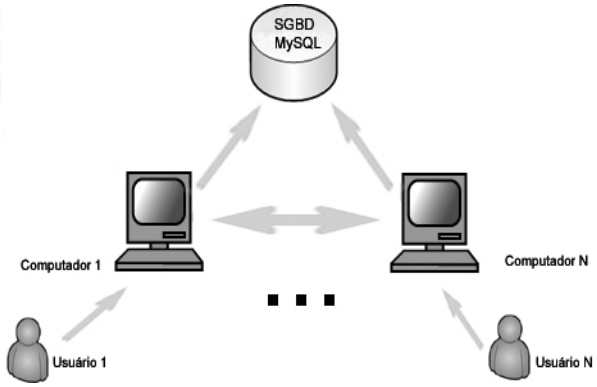

Figura 7.1: Arquitetura cliente-servidor aplicada ao *EasyShare* com n computadores

[1] *Integrated Development Environment*. É um ambiente gráfico utilizado para facilitar o desenvolvimento de produtos de *software*.

[2] *Local Area Network*, ou rede local. É uma rede de pequeno porte, que compreende, em média, computadores de um mesmo prédio ou empresa.

[3] Sistema de gerenciamento de banco de dados.

O *EasyShare* apresenta segurança aos usuários, pois é um sistema baseado em *login* e utilização de senhas, o que impede que o mesmo seja utilizado por qualquer pessoa que não seja cadastrada no banco de dados. E, além de controlar o acesso ao sistema, o *EasyShare* possuiu um usuário administrador, que é o tipo de usuário que poderá gerenciar os demais, realizando operações de cadastro, alteração e até remoção de usuários no sistema. Isso permite que a segurança esja restrita a apenas um membro, mas não significa que ele tenha acesso aos dados referentes às partes das senhas de todos os usuários, pois essas informações estão criptografadas no banco de dados, assim como as senhas que os usuários utilizam para realizar o *login*, e isso faz com que somente o usuário responsável pela parte da senha possa liberar o acesso à mesma aos outro usuários. Todos esses casos serão detalhados a seguir.

7.2.2 Instalação e configuração

Como foi mostrado anteriormente o sistema centraliza suas informações em um banco de dados MySQL, fazendo com que a aplicação necessite de um servidor de banco de dados. Esse servidor é instalado em apenas uma das máquinas, sendo que as outras irão apenas se conectar a ela para poder realizar as transações com os dados. Assim, basta seguir as instruções em [MySQL, 2005] e fazer o download da versão gratuita do SGBD para utilizá-lo como central de dados.

A seguir será necessário instalar a JVM (*Java Virtual Machine*), ou Máquina virtual Java, que é um mecanismo desenvolvido para diferentes plataformas e que tem o papel de ser o plano de fundo necessário para que códigos escritos em Java possam ser executados no computador em questão.

Feitos esses dois passos iniciais, deve-se realizar o *download* do arquivo compactado[4], e em seguida descompactá-lo. Ao fim do processo de descompactação a estrutura de diretórios da figura 7.2 será gerada.

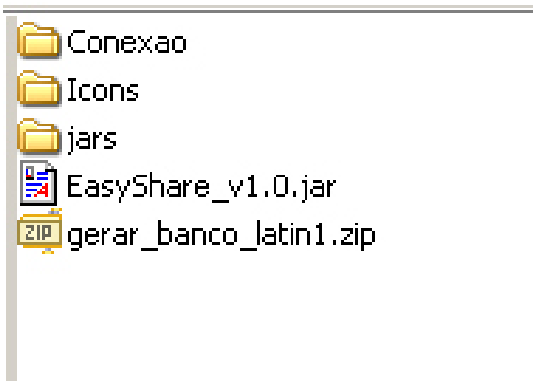

Figura 7.2: Estrutura de diretórios do *EasyShare*

Agora, com o banco de dados MySQL instalado no sistema, pode-se descompactar o arquivo "*gerar_banco_latin1.zip*" para que seja gerado o arquivo "*gerar_banco_latin1.sql*". Esse arquivo do tipo ".sql" contém as informações necessárias para que as tabelas do banco de dados possam ser criadas e o aplicativo esteja preparado para receber os dados. Para que o banco de dados esteja pronto basta criar um novo banco

[4]Encontra-se em www.comp.ufla.br/~felipe/easyshare_v1.0.zip

de dados no MySQL e dentro deste banco criado executar o código contido no arquivo "*gerar_banco_-latin1.sql*". O modelos do banco de dado está delhados no Apêndice A.

Com o banco de dados pronto para trabalhar em conjunto com a aplicação, basta configurar o *EasyShare* para se conectar com o servidor MySQL. Existe um arquivo de configuração, "*conec.txt*", que está dentro da pasta "*Conexao*" e tem a finalidade de informar ao sistema os dados necessários para conexão. A figura 7.3 mostra o conteúdo deste arquivo.

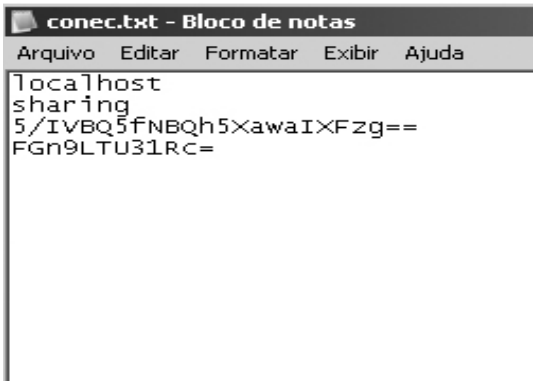

Figura 7.3: Conteúdo do arquivo *conec.txt*

Na seção seguinte será mostrada como é feita a configuração do sistema através deste arquivo.

7.3 A ferramenta em execução

A figura 7.4 mostra a tela inicial do sistema, onde se pode ver o *menu* e a janela para a realização do login. A figura 7.5 mostra a janela do login.

Figura 7.4: Tela inicial do *EasyShare*

O *EasyShare* é um sistema baseado em *login* e senha de usuários, fazendo com que a segurança seja sufiente para que pessoas não autorizadas não utilizem o aplicativo. Portanto, de acordo com a figura 7.6, a maioria das funções dos menus não estarão disponíveis até o momento em que o usuário entrar com um

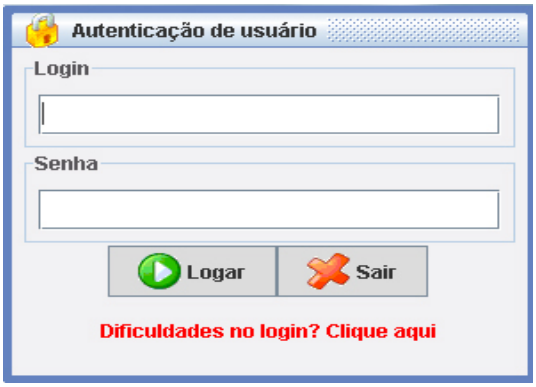

Figura 7.5: Janela para a realização do *login* no sistema

login válido e sua respectiva senha.

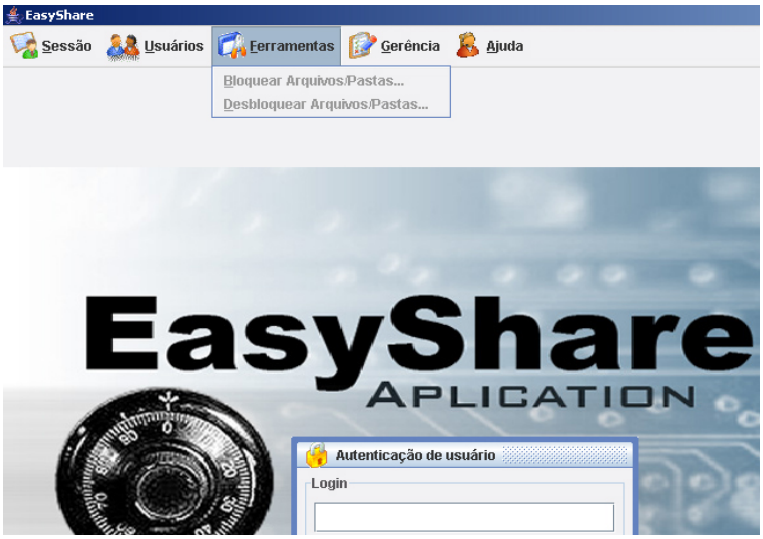

Figura 7.6: Funcionalidades inativas até que se realize *login* no sistema

Como foi dito anteriormente, existe o arquivo "*conec.txt*" que serve para se configurar a conexão do aplicativo com o banco de dados MySQL. Assim, cada máquina que executar uma instância do *EasyShare* deverá ter esse arquivo configurado corretamente, caso contrário a conexão com o banco de dados não será realizada.

A figura 7.3 visualizada acima mostra o conteúdo do arquivo de configuração "*conec.txt*" que contém apenas 4 linhas com os dados para conexão. Para um melhor entendimento, segue a explicação para cada linha contida no arquivo:

- *Linha 1:* é a linha que contém o endereço do computador onde está localizado o servidor de banco de dados. Os dados desta linha podem ser tanto em endereço IPs quanto em *strings* com os nomes. Ex.: 192.168.241.23; localhost; www.seudominio.com.br; 10.0.0.1; etc...

- *Linha 2:* é a linha onde se deve informar o nome do banco de dados onde estão armazenadas as informações.

- *Linha 3:* linha que contém o *login* utilizado para se conectar ao banco de dados.
- *Linha 4:* linha que contém a senha utilizada para se conectar ao banco de dados.

Como pode ser percebido, as linhas 3 e 4 do arquivo "*conec.txt*" estão criptografadas para que a segurança do banco de dados não seja comprometida. A técnica utilizada para gerar essas *strings* criptografadas é a do algoritmo MD5[5] [Schneier, 1996], que visa apenas gerar a *string* criptografada, sem possuir função para recuperar o antigo valor. Sendo assim, para que o usuário possa conseguir uma *string* criptografada basta acessar o menu "*Gerência*" do *EasyShare* e posteriormente escolher a opção "*Gerar string criptografada...*", fazendo com que a janela da figura 7.7 seja mostrada. Deve-se observar que para realizar esta operação o usuário não precisa estar logado no sistema.

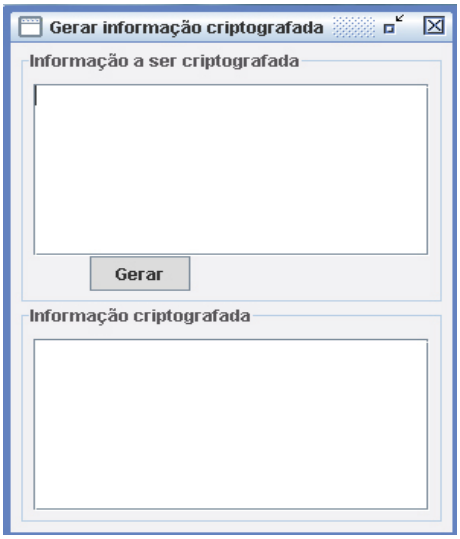

Figura 7.7: Tela que gera string criptografada

Assim, usa-se a caixa de texto superior para inserir o texto a ser criptografado, e ao pressionar o botão "Gerar" a caixa de texto inferior irá exibir a informação já criptografada, como na figura 7.8, que mostra um exemplo.

A partir daí basta copiar essas informações geradas pelo sistema e colar no arquivo de configuração "*conec.txt*".

7.3.1 Controle de Usuários

Como foi dito anteriormente, existe um controle de usuários que é realizado através de *login* e senha no sistema, fazendo com que a utilização do *software* se restrinja apenas a usuários cadastrados previamente.

Todo esse controle de usuários só pode ser realizado por apenas um membro: o administrador, que é o responsável pelo cadastramento, manutenção e exclusão de usuários no sistema. Esse tipo de usuário é criado juntamente com as tabelas do banco de dados, e só existe um administrador no sistema. O login

[5] *Message Digest 5*.

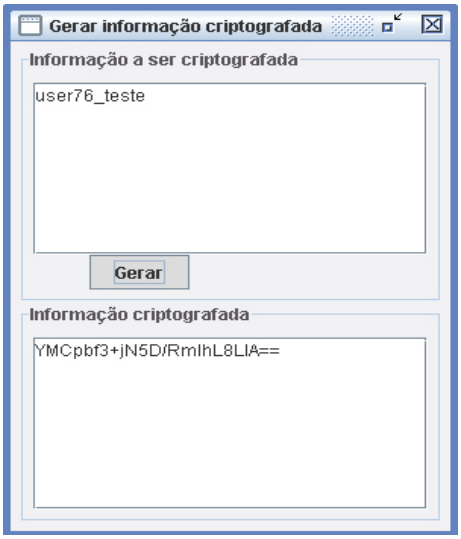

Figura 7.8: Exemplo de informação criptografada

para o adiministrador é "*admin*", e inicialmente a senha também; mas quando o administrador *logar* pela primeira vez no sistema sua senha poderá ser modificada normalmente.

O usuário administrador só se destaca por poder controlar o cadastro e a manutenção dos usuários, pois as outras funcionalidades são exatamente iguais aos dos usuários regulares do sistema.

Como mostra a figura 7.9, os usuários comuns não têm acesso às funcionalidades de gerenciar usuários no sistema. E a figura 7.10 mostra o usuário administrador logado no sistema, onde se pode verificar que as funcionalidades são liberadas.

Figura 7.9: Usuário comum não tem permissões para gerenciar usuários.

Inicialmente o *EasyShare* só possuiu o administrador como usuário cadastrado, e como depende apenas dele para que novos usuários utilizem o sistema, basta que se vá adicionando as informações referentes aos novos usuários através do menu "*Usuários*" e a opção "*Novo usuário...*". Assim, uma janela com as informações necessárias (figura 7.11) deverá ser preenchida para que o o novo usuário seja cadastrado e

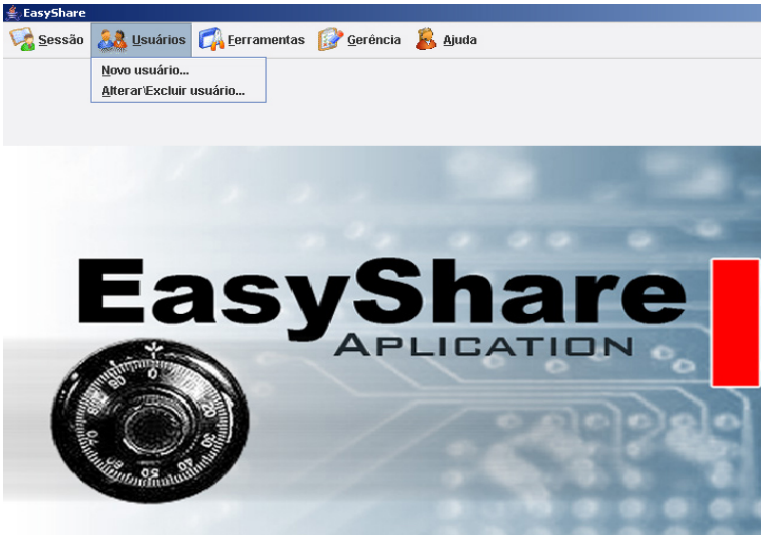

Figura 7.10: Usuário administrador com o controle de usuários liberado.

posteriormente possa iniciar a utilização do *EasyShare*.

As informações necessárias para o cadastro são:

- *Login*: responsável por identificar o usuário no sistema, pois o *login* é único para cada usuário.
- *Nome*: nome do novo usuário.
- *Email*: campo opcional onde se pode preencher com o email do usuário. Pode-se ser utilizado futuramente, caso novas funcionalidades sejam incorporadas ao projeto.
- *Senha*: senha de acesso do usuário.
- *Confirma senha*: campo para redigitar a senha.

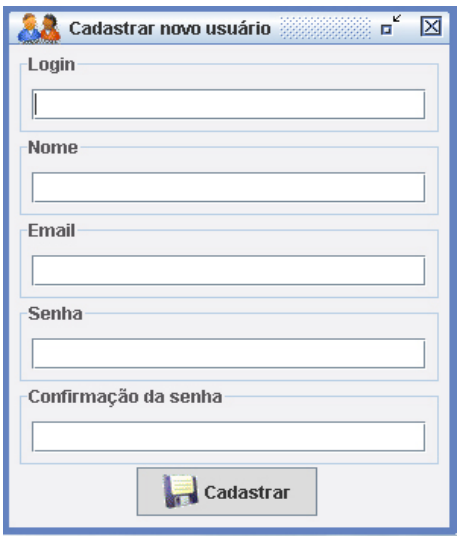

Figura 7.11: Janela de cadastro de novos usuários.

Quando existem mais usuários cadastrados no sistema há a necessidade de se ter um controle sobre os mesmos. Para isso o administrador poderá remover ou alterar informções sobre os usuários cadastrados através do menu "*Usuários*" e a opção *"Alterar \Excluir usuário..."*, fazendo com que a listagem da figura 7.12 esteja disponível para que o administrador possa selecionar um usuário para alterar suas informações ou removê-lo, como mostrado na figura 7.13.

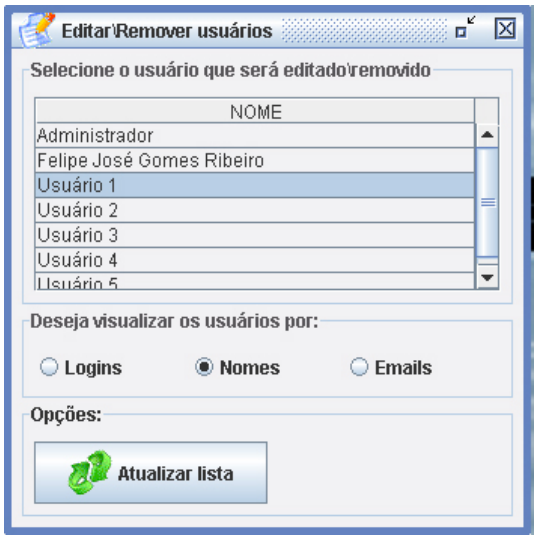

Figura 7.12: Listagem dos usuários cadastrados.

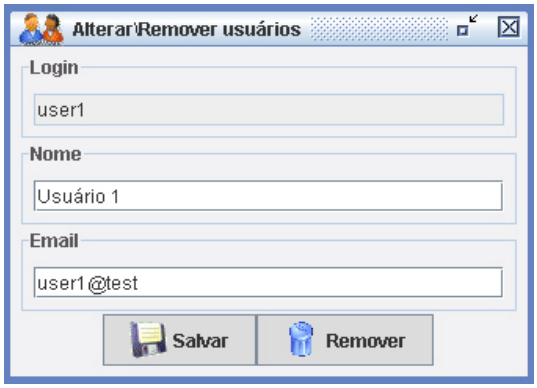

Figura 7.13: Janela onde se altera informações ou remove usuário.

7.3.2 Bloqueando e Desbloqueando Arquivos e Pastas

Após a descrição do processo de como os usuários são cadastrados no sistema, já se tem os elementos e o conhecimento necessário para que arquivos e pastas possam ser protegidos utilizando-se o processo da partilha de senhas.

Deve-se destacar que a finalidade de se proteger arquivos e pastas foi escolhida pela fato de que um computador possui informações que têm características de importância e privacidade muito elevadas, fazendo com que métodos mais seguros e confiáveis se tornem cada vez mais necessários para proteger essas entidades. Mas isso não quer dizer que a partilha de senhas se restrinja apenas a essas resoluções

de problemas; muito pelo contrário, pois este método se mostrou muito eficaz para muitos problemas que envolvem segurança de informação.

Para proteger um arquivo ou pasta dentro da rede onde está o *EasyShare* basta utilizar o menu "*Ferramentas*" e em seguida selecionar a opção "*Bloquear arquivos \pastas...*", fazendo com que a janela da figura 7.14 apareça.

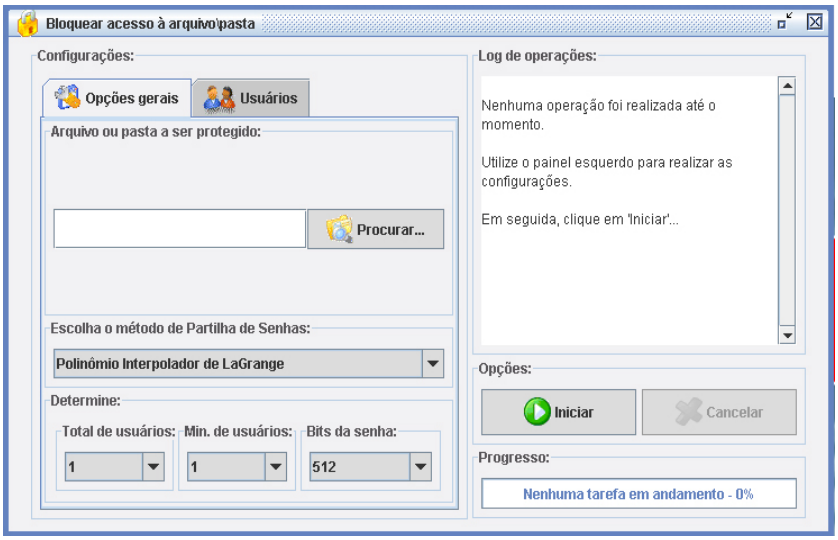

Figura 7.14: Janela para realizar o bloqueio (aba Opções gerais).

Como esta janela possuiu duas abas de configuração, inicialmente serão apresentadas as opções disponíveis na aba "*Opções gerais*". Nesta aba pode verificar a existência de um botão "*Procurar*" e uma caixa de texto ao lado. É nesta caixa de texto que deverá ser inserido o caminho da pasta ou do arquivo que será protegido. Assim, o botão "*Procurar*" abre uma caixa de seleção de arquivos onde se pode selecionar o arquivo ou a pasta escolhido.

Feita a seleção do arquivo ou da pasta, deve-se selecionar método para realizar a partilha. Neste projeto foi implementado apenas o método do Polinômio Interpolador de Lagrange, que já foi detalhado anteriormente. A escolha por esse método foi marcada pelo fato de que existe muito material disponível a respeito deste método e as operações necessárias para a implementação do mesmo, o que facilitou bastante o processo de desenvolvimento da ferramenta. Mas mesmo assim os nomes dos outros métodos foram mantidos na caixa de seleção, pois trabalhos futuros podem surgir apenas pela curiosidade de alguém em saber a respeito do funcionamento de outro método de partilha de senhas.

Após a escolha do método do Polinômio Interpolador de Lagrange, as informações a seguir devem ser escolhidas:

- *Total de usuários:*número total de usuários que irão partilhar a senha;
- *Mín. de usuários:*número mínimo de usuários necessários para gerar a senha inicial s que irá desbloquear o arquivo ou a pasta protegida;

- *Bits da senha:*número que representa o tamanho da senha em bits;

Nenhuma dessas opções é editável, mas sim selecionável, fazendo com o usuário se sinta mais a vontade com a interface. Isso também é uma prevenção contra erros de preenchimento incorreto de campos. Essas opções referentes aos usuários variam de acordo com a quantidade de usuários cadastrados no sistema. Agora, a opção de "*Bits da senha*" sempre possuirá os valores 512, 768, 1024 e 2048.

Realizadas as configurações necessárias na aba de *"Opções gerais"*, agora só é necessário escolher os usuários que deseja partilhar a senha na aba *"usuários"*, de acordo com a figura 7.15. O número de usuários selecionados deverá ser igual ao escolhido na aba de *"Opções gerais"*, opção "*Total de usuários*".

Figura 7.15: Janela para realizar o bloqueio (aba Usuários).

As figuras 7.16 e 7.17 mostram o processo quando atinge 70% da tarefa completa e quando a tarefa já está terminadam respectivamente.

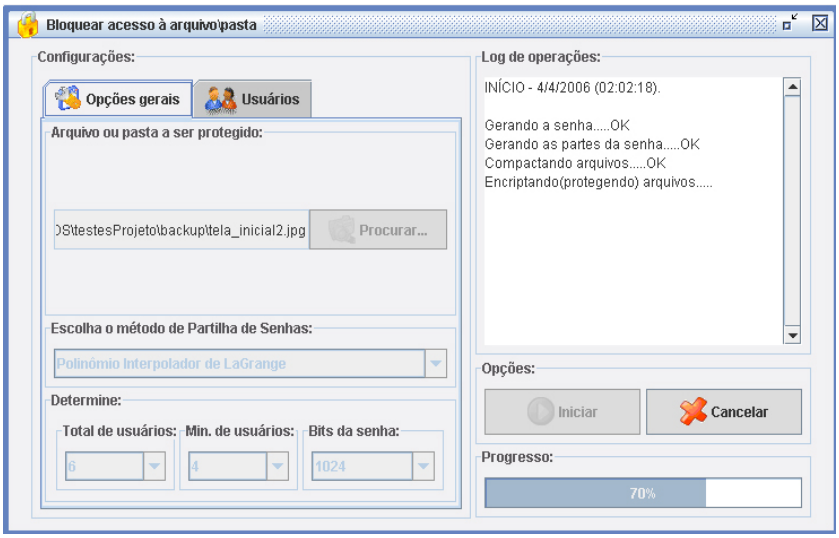

Figura 7.16: Tarefa 70% completa.

No fim do processo o arquivo ou pasta original é apagado e é gerado um arquivo de extensão ".aps"

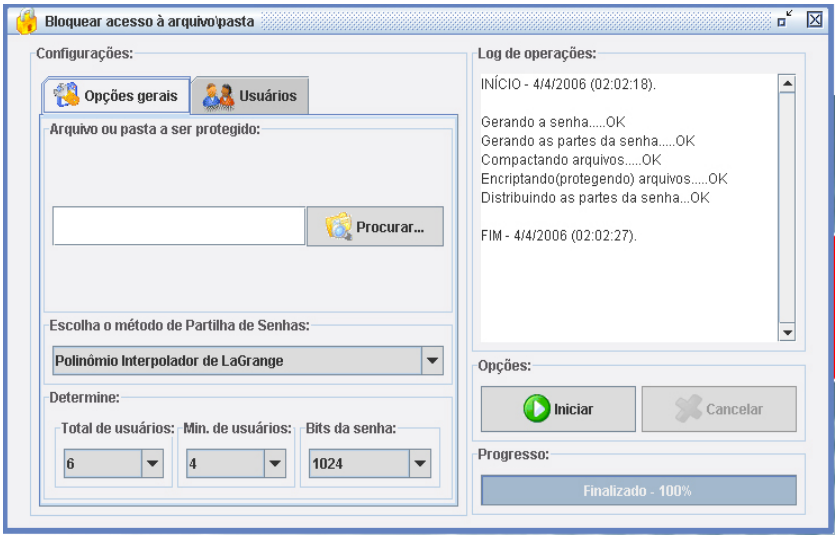

Figura 7.17: Tarefa já completa.

com os dados originais criptografados. Para cada proteção realizada, seja de arquivos ou pastas é gerado um arquivo do tipo ".aps", assim, para desbloquear o arquivo basta selecionar este arquivo e realizar as operações que serão descritas em breve. A informação do arquivo protegido e suas partes são armazenadas no banco de dados, de acordo com a estrutura das tabelas mostrada no Apêndice A. Caso o arquivo seja renomeado ou movido de lugar a sua referência no banco de dados não se altera, pois o arquivo é identificado pelo hash de seu conteúdo, e não pelo nome ou caminho. Hash é uma função que gera uma sequência de caracteres de acordo com o parâmetro passado, no caso o conteúdo do arquivo protegido. Assim essas sequência de caracteres é armazenada no banco de dados para identificar o arquivo. O Hash é gerado aplicando-se a função MD5, que já foi descrita anteriormente.

Agora, retornando à janela de bloqueio, pode-se observar que existe uma área de texto onde informações vão sendo adicionadas no decorrer do processo. Em caso de erro em alguma das etapas, o erro também é reportado nessa área de texto. Também pode-se visualizar o tempo em que a tarefa se inicia e o tempo em que a tarefa termina. No caso do nosso exemplo, a tarefa demorou 9 segundos para ser completada. Pode-se analizar os resultados obtidos nas tabelas 7.1 e 7.2. Esses resultados foram obtidos utilizando uma máquina com as seguintes características (relevantes para o estudo):

- Processador: AMD Athlon 1400+;
- Memória RAM: 512MB;

total usuários	min. usuários	bits senha	tempo(segundos)
5	1	1024	17
5	2	1024	17
5	1	2048	23
7	1	768	18

Tabela 7.1: Tempo gasto para bloquear uma pasta de tamanho 11,9 MB

total usuários	min. usuários	bits senha	tempo(segundos)
5	1	1024	2
5	2	1024	2
5	1	2048	12
7	1	768	2

Tabela 7.2: Tempo gasto para bloquear um arquivo de tamanho 136 bytes

Estes resultados mostram que o item de mais peso para se avaliar a performance do sistema é o tamanho de *bits* da senha. E, claro que se o tamanho do arquivo ou pasta aumenta o tempo também aumenta proporcionalmente, pois antes de realizar a proteção o *EasyShare* faz uma compactação dos dados, que se torna uma função muito útil para se economizar espaço no disco rígido.

Após todo esse processo de proteção, quando as partes das senhas já estiverem distibuídas, o usuário que possui uma nova parte receberá uma aviso quando realizar o próximo *login* no sistema, como mostrado na figura 7.18. Essa mensagem irá conter informações sobre o arquivo bloqueado, e no caso de o usuário possuir mais de uma nova parte, será listada a informação de cada novo arquivo.

Figura 7.18: Mensagem de nova parte de senha mostrada ao usuário.

Agora, com a informação já protegida, pode-se supor que algum usuário cadastrado deseja ter acesso a ela. Mas deve ser salientado que não é necessário que o usuário possua uma parte da senha do arquivo desejado, pois isso não vai ter nenhum tipo de influência sobre a segurança da informação, já que a autorização dos usuários que possuem as partes da senha ainda é obrigatória.

Para poder liberar a informação para utilização o usuário deverá utilizar o menu "*Ferramentas*" e em seguida selecionar a opção "*Desbloquear arquivos \pastas...*", fazendo com que a janela da figura 7.19 se torne visível.

Como foi detalhado anteriormente, arquivos com a extenção ".aps" são gerados com informação dos arquivos ou pastas que foram protegidos. Assim, a primeira coisa a se fazer para desbloquear uma informação é selecionar o arquivo ".aps" referente à informação original que foi protegida. Logo depois que

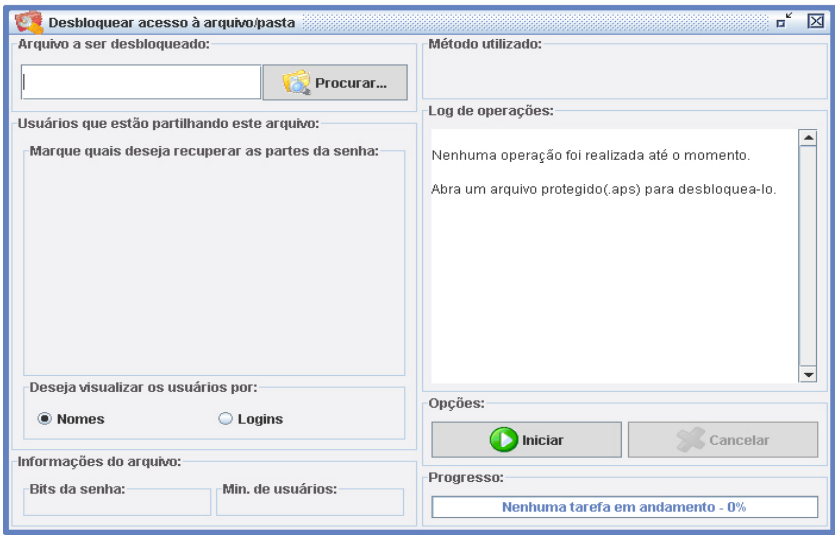

Figura 7.19: Janela para desbloqueio de informação.

o arquivo foi selecionado, o sistema irá calcular o *hash* deste arquivo e realizar uma busca no banco de dados para que as informações desta proteção possam ser exibidas ao usuário. Também serão exibidos os nomes (ou logins) dos usuário que partilham a senha da proteção, e ao lado dos nomes existem *checkboxes* que o usuário deverá marcar informando que deseja requisitar a parte da senha deste usuário. Detalhes mostrados na figura 7.20.

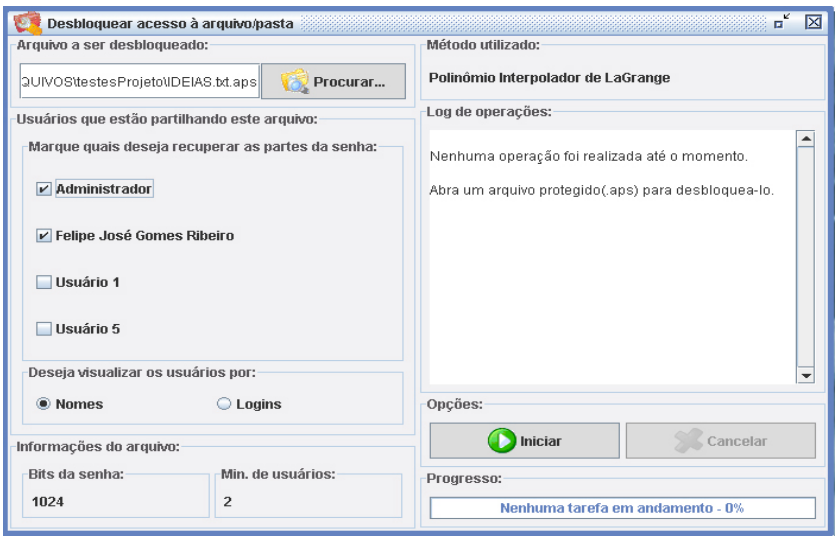

Figura 7.20: Depois de selecionar o arquivo, informações sobre a proteção são exibidas.

Após os usários serem selecionados para que suas partes da senha sejam recuperadas e o botão “Iniciar” clicado, uma nova janela é exibida, informando o nome do usuário e seu *status*. Este *status* possui duas fases:

- Na primeira fase o sistema verifica se os usuários estão utilizando o *EasyShare* em alguma máquina da rede. Assim, os *status* possíveis para essa fase são: *Verificando*, onde o sistema está fazendo a busca pelo usuário; *Online*, que informa que o usuário está disponível para enviar sua parte da

senha; e *Offline*, que informa que o usuário não está disponível para enviar sua parte da senha. A figura 7.21 mostra a janela nessa primeira fase.

- A segunda fase tem por objetivo recuperar a senha dos usuários que estão *online*, e o *status* desses usuários ainda pode ser modificado para: *Requisitando*, onde o sistema faz a requisição da parte da senha ao usuário; *Aceita*, informando que a parte da senha foi liberada pelo usuário; e *Negada*, que informa que o usuário não liberou o acesso à sua parte da senha. A figura 7.22 mostra a segunda fase da recuperação das senhas.

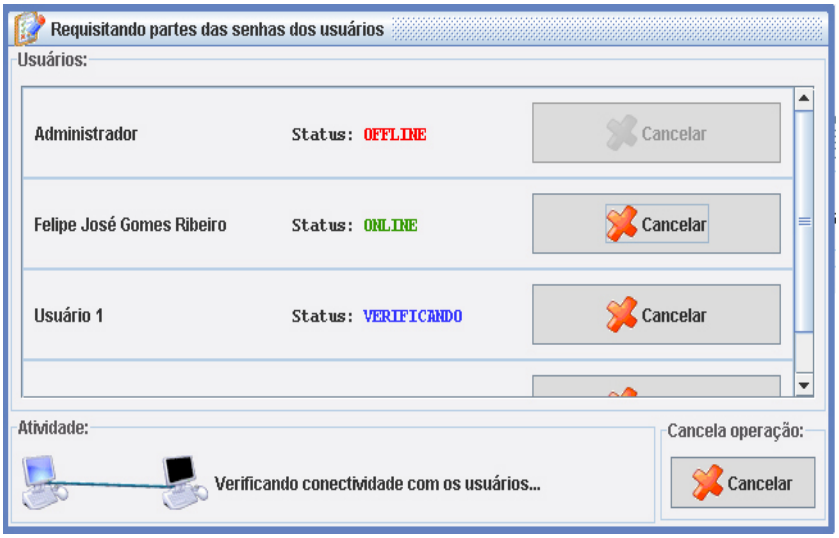

Figura 7.21: Procurando por usuários disponíveis.

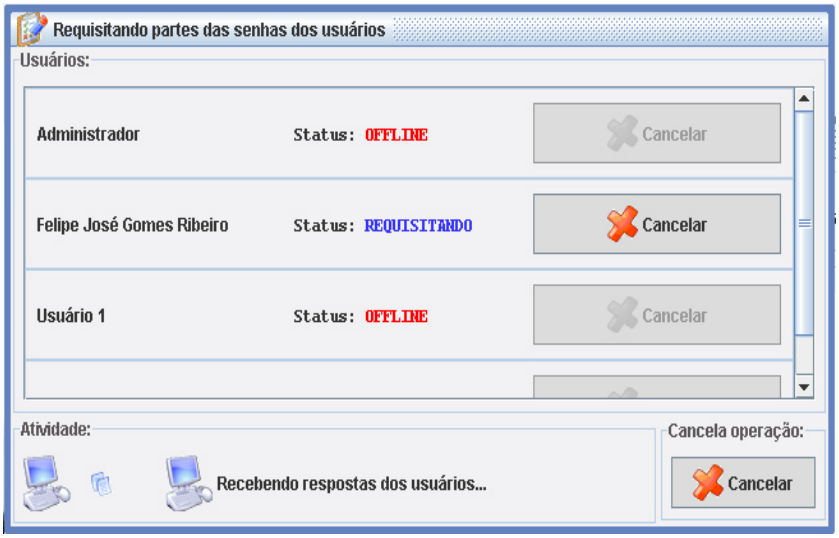

Figura 7.22: Requisitando partes aos usuários.

No início da segunda fase, quando o sistema já sabe quais usuários estão diponíveis, é enviada uma requisição de parte de senha aos usuários que estão *online*. O usuário recebe essa requisição em sua máquina enquanto está utilizando o sistema, e decide se libera o acesso à sua parte da senha ou não. Nesse

pedido estão presentes as informações do usuário que está solicitando a senha e também do arquivo ou pasta original que se deseja obter a parte da senha, como mostrado na figura 7.23.

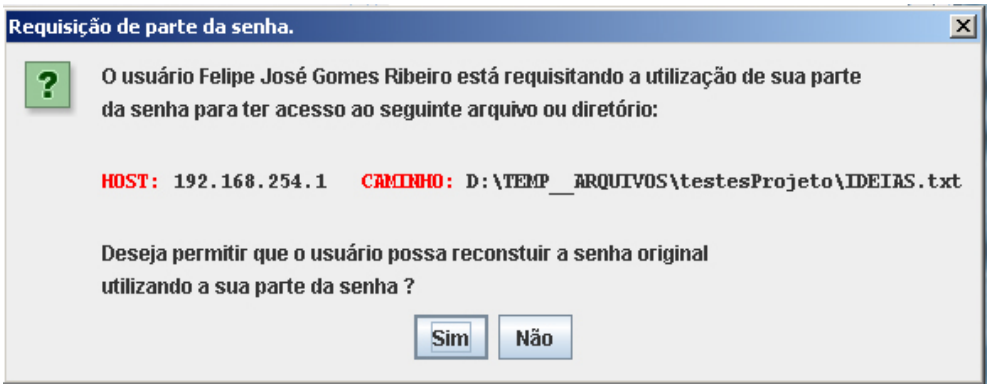

Figura 7.23: Janela que requisita a senha ao usuário.

Toda essa comunicação entre as instâncias do *EasyShare* é baseada no príncipio de cliente-servidor que já foi descrito anteriormente. Portanto, cliente é o usuário que faz o pedido da senha, e servidor é o usuário que irá receber a requisição e posteriormente irá responder. Assim, toda instância do *EasyShare* funciona como cliente e como servidor, pois o servidor fica constantemente "escutando" uma porta do computador esperando por uma requisição do cliente, que se inicia toda vez que um usuário deseja recuperar a parte de uma senha.

Deve-se salientar que quando um usuário concorda em liberar a utilização de sua parte da senha, ele não envia sua parte da senha, mas apenas uma mensagem de confirmação, pois sua parte da senha está armazenada no banco de dados.

Depois de finalizada a requisição, o sistema irá conferir se já possuiu o número mínimo de senhas para liberar o acesso à informação protegida. Caso não possua, mostra uma mensagem de erro será exibida.

Quando a informação é liberada para uso, o sistema pergunta se o usuário deseja manter o arquivo ".aps" que foi selecionado ou se deseja que este arquivo seja apagado. A figura 7.24 mostra o fim do processo.

7.3.3 Gerência de senhas

O *EasyShare* também possui uma funcionalidade que permite ao usuário saber quais partes de senhas estão associadas a ele. também permite ao usuário detalhar as informações sobre as partes e remover uma parte de senha. Quando remove uma parte de senha o usuário desliga sua parte de senha do arquivo à que era relacionado. A figura 7.25 mostra a tela de gerência e a figura 7.26 mostra a janela que detalha as informações de determianda parte de senha.

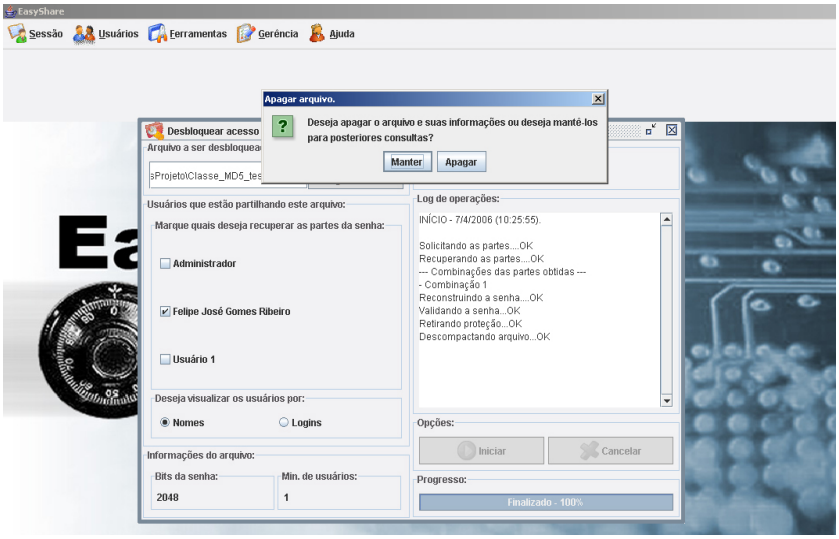

Figura 7.24: Fim do processo de liberação de informação.

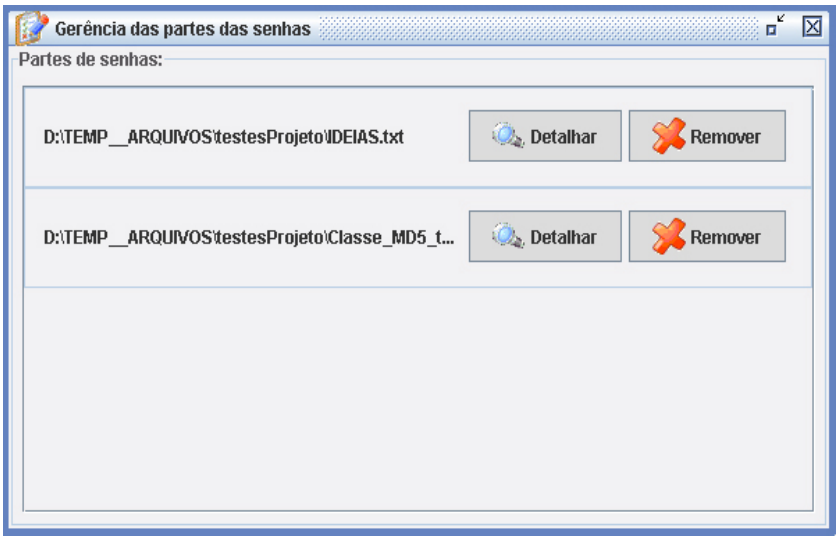

Figura 7.25: Gerência das partes de senhas

7.4 Modelagem UML

A seguir será mostrado o aspecto da modelagem do sistema utilizando diagramas baseados na linguagem UML, citado por [Ahmed e Umrysh, 2002]. Inicialmente é mostrado o diagrama de casos de uso, em seguida o diagrama de subsistemas e, posteriormente, o diagrama de classes, divido em três, para que possa ficar mais clara a apresentação de como foi o processo de desenvolvimento do *EasyShare*. Também é apresentada forma de que como as classes se relacionam.

7.4.1 Diagrama de casos de uso

Os casos de uso identificados são mostrados na figura 7.27. Os detalhamentos dos casos de uso podem ser encontrados no Apêndice B.

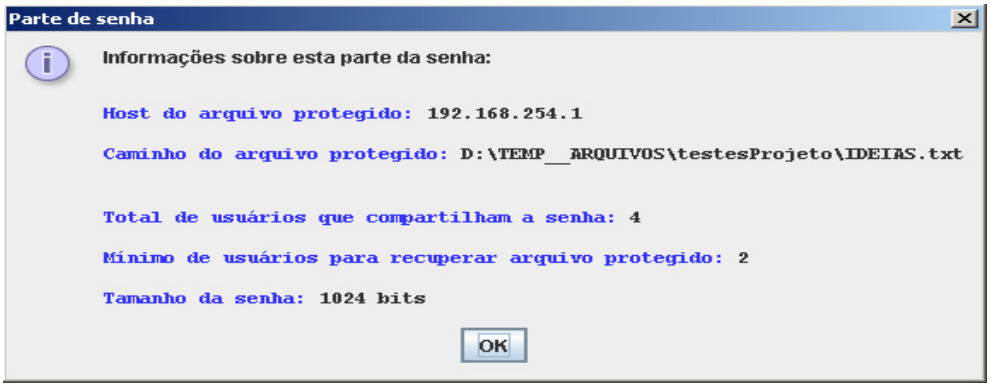

Figura 7.26: Datalhes de uma parte de senha.

7.4.2 Diagrama de classes dos subsistemas

O *EasyShare* é subdivido em camadas, ou subsistemas, e para que seja clara essa divisão a figura 7.28 mostra como foi feita. No processo de desenvolvimento foi utilizaa uma camada IHM, para interfaces; uma camada DAO[6], responsável pela persistência e, por último, uma camada Handler para a integração e comunicação entre as outras camadas.

Dentre as 3 camadas, primeiro existe a camada de IHM que controla o que é mostrado graficamente ao usuário, incluindo janelas, botões, mensagens de erro, formulários. As interfaces presentes no sistema estão descritas na figura 7.29.

Logo em seguida está a camada Handler, que trata da lógica de negócios do sistema e faz as requisições à camada DAO, responsável pela persistência de dados.

As classes *Handler* e *DAO* estão detalhadas com os respectivos métodos e atributos nas figuras 7.31 e 7.32.

Na figura 7.30 estão detalhadas as classes do tipo *Detail* e os seus devidos relacionamentos.

Na figura 7.33, está detalhado o realcionamento que ocorre entre as classes.

[6]camada *Data Access Object*, usada para encapsular todo o processo de acesso às fontes de dados.

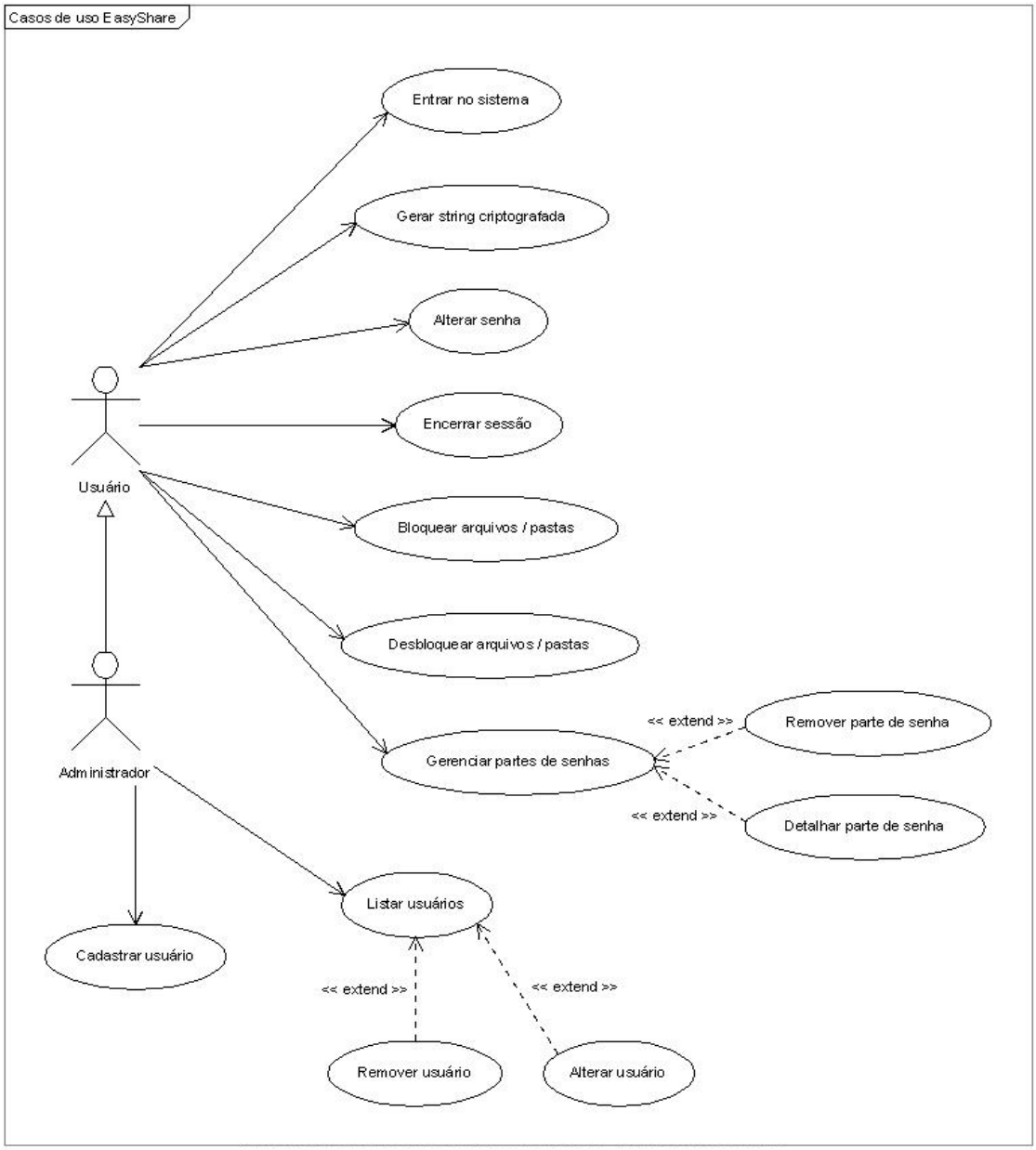

Figura 7.27: Diagrama de casos de uso do *EasyShare*.

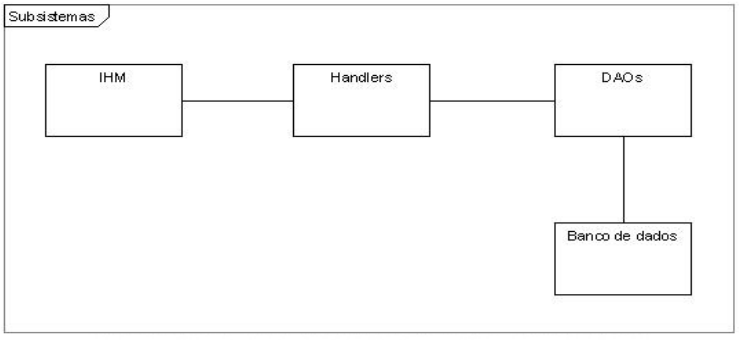

Figura 7.28: Susbsistemas do *EasyShare*.

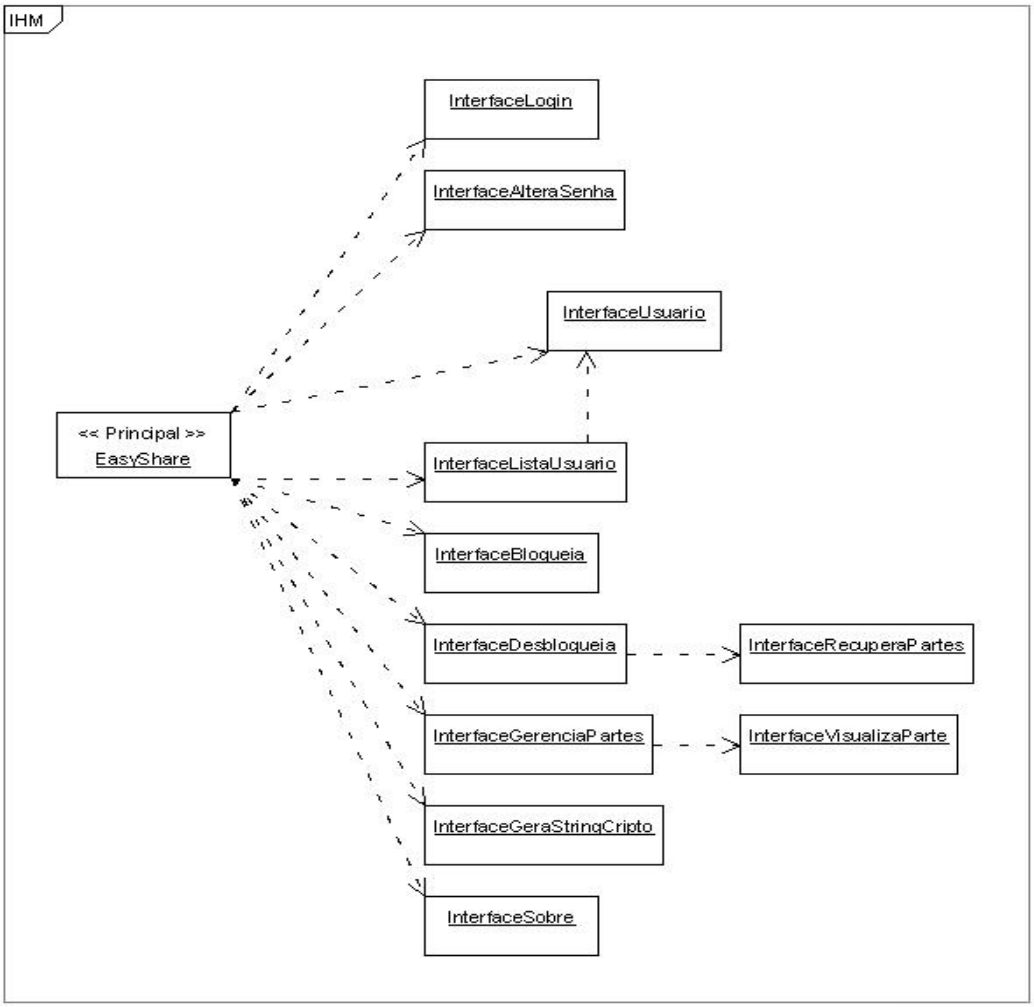

Figura 7.29: Diagrama de classes do subsistema IHM.

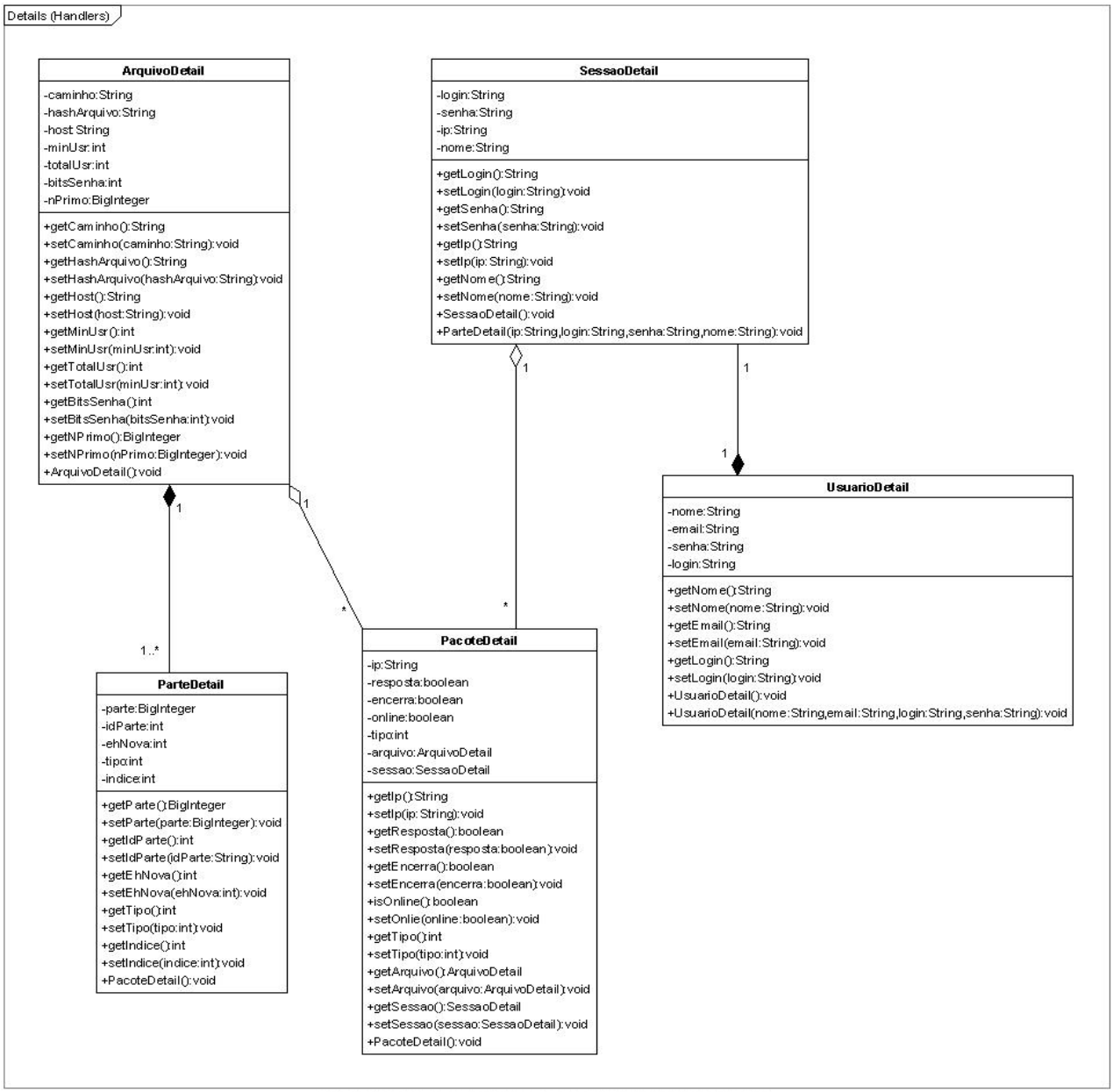

Figura 7.30: Diagrama das classes Details.

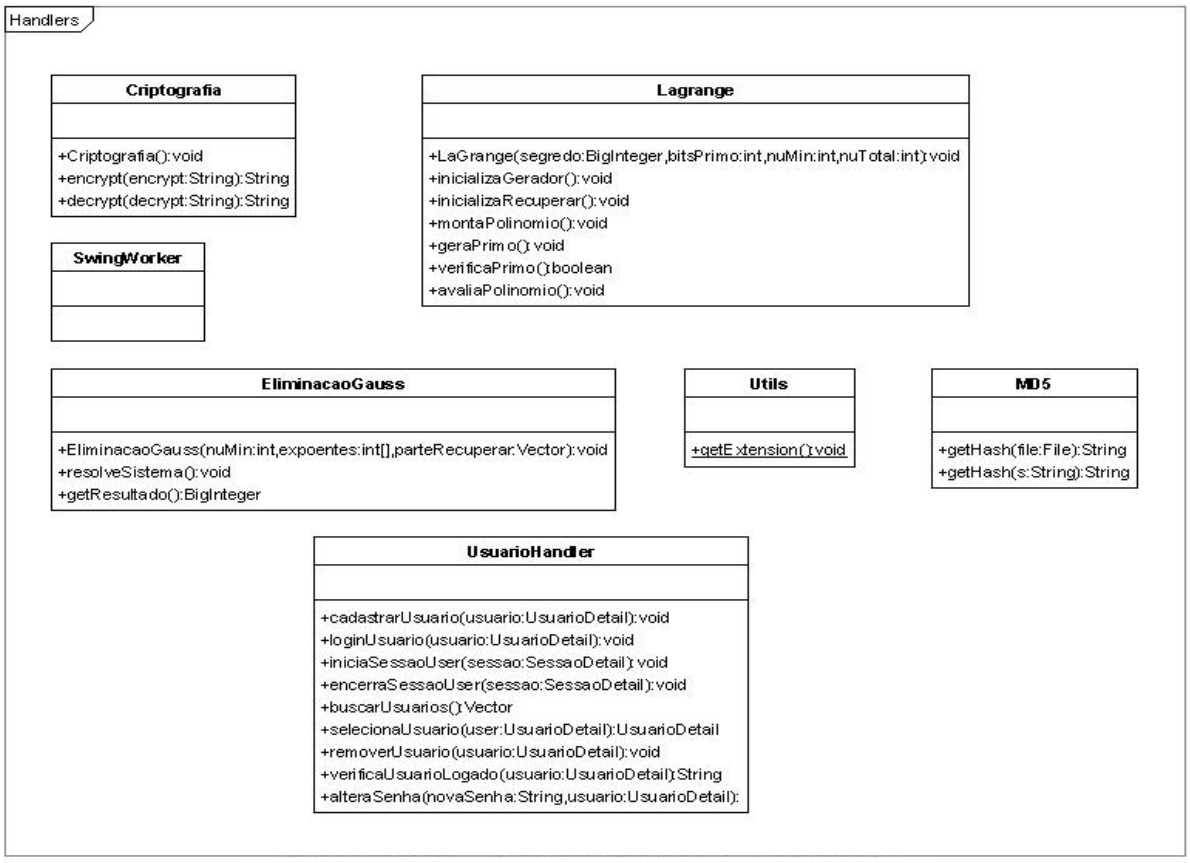

Figura 7.31: Diagrama das classes Handler

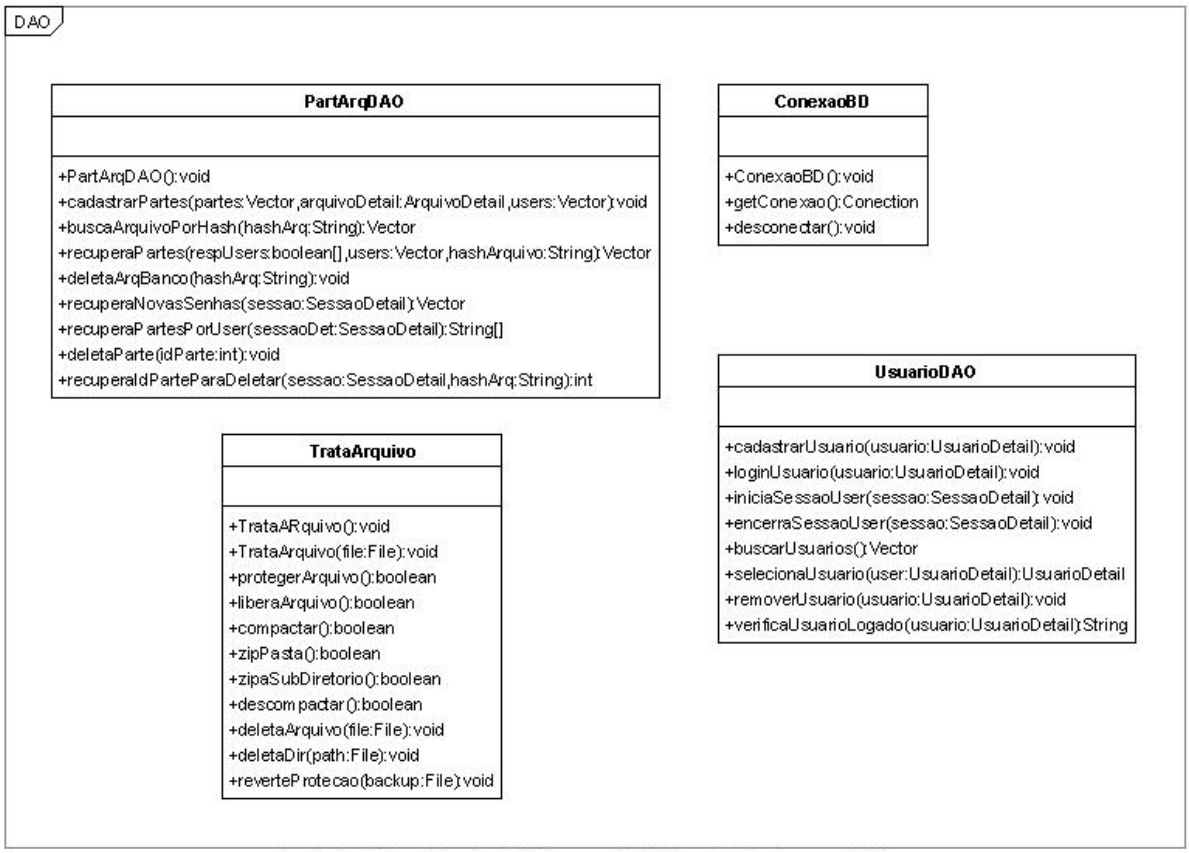

Figura 7.32: Diagrama das classes DAO

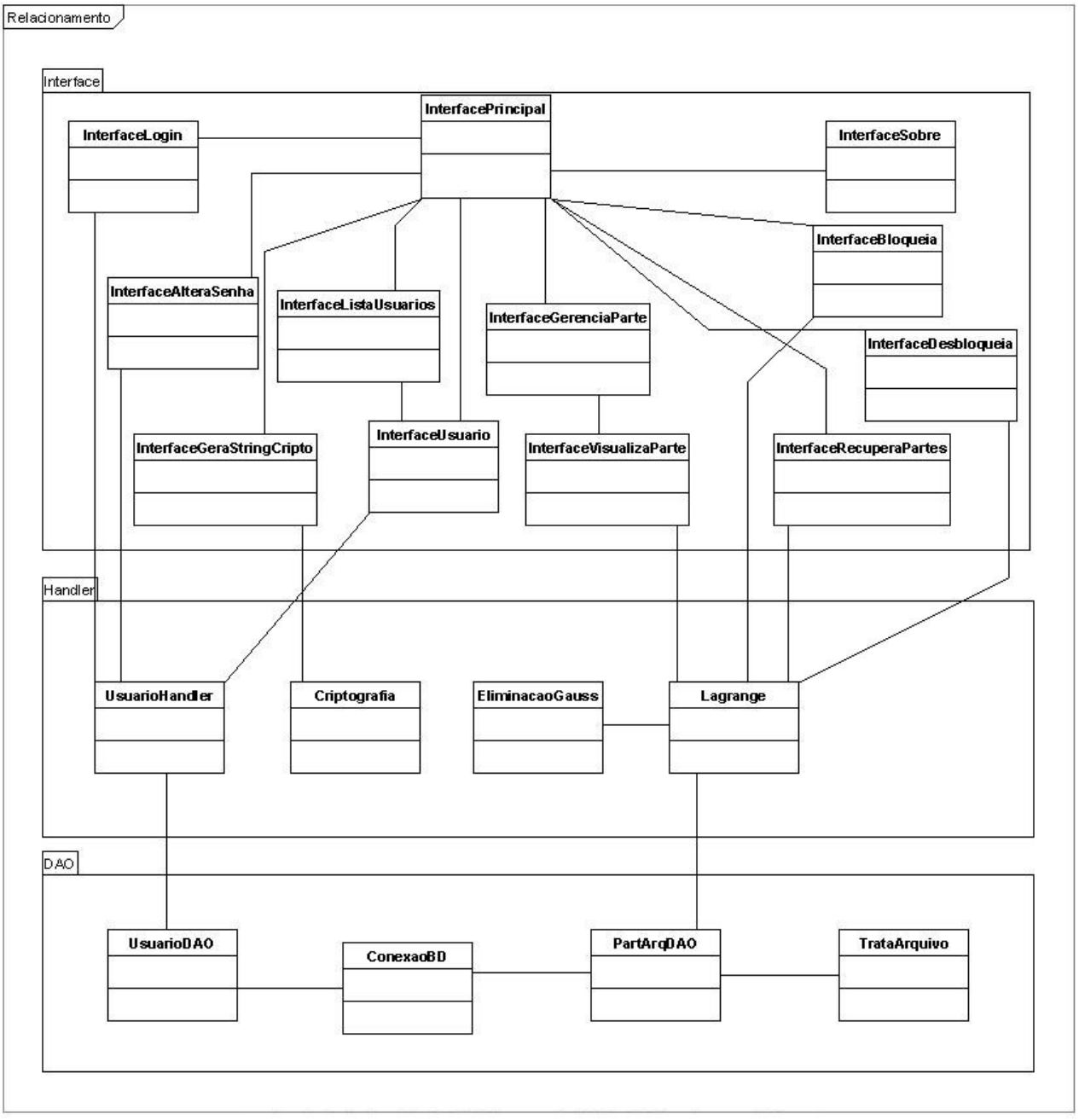

Figura 7.33: Relacionamento entre as classes.

Capítulo 8

CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões deste trabalho de pesquisa, assim como algumas propostas para trabalhos futuros.

8.1 Conclusões

Neste trabalho foram apresentadas as evoluções da criptografia e da segurança ao longo da história, mostrando que processos que antes eram bastante úteis e se mostravam eficazes perderam completamente a utilidade diante das modernas técnicas que se têm para poder quebrar códigos. Com a constante evolução da computação utilizada em conjunto com as técnicas matemáticas pode-se implementar os mais variados algoritmos propostos anteriormente e o *EasyShare* é um exemplo de que apesar dos métodos terem surgido há pelo menos 30 anos, eles ainda possuem um grande poder contra fraudes em senhas.

O *EasyShare* tem por finalidade proteger informações de acesso não autorizado e realiza esta função exatamente como foi designado, mas também pode ser utilizado para fins ilícitos, pois o sistema é aberto a quem desejar modificá-lo. Assim, o papel do autor neste trabalho é apresentar os benefícios do projeto, ficando a cargo dos usuários e desenvolvedores saber utilizar o sistema da maneira correta.

8.2 Trabalhos Futuros

Como este trabalho apresentou vários métodos para se resolver o problema da partilha de senhas, alguns trabalhos podem ser propostos a fim de se elevar o nível do projeto futuramente:

- Implementar outros métodos: existem vários métodos descritos neste trabalho, mas apenas um deles foi implementado.

- Abranger a finalidade do sistema: atualmente o *EasyShare* é utilizado para proteger arquivos e pastas, mas pode-se ampliar as utilizações para o sistema, como restringir o uso a outros aplicativos ou até mesmo periféricos, pois é totalmente possível controlar dispositivos ligados às portas seriais e paralelas do computador.

- Integração com a *web*: utilizar o sistema integrado à *internet*, onde qualquer usuário poderá compartilhar as senhas de suas informações com outro usuário fisicamente distante. O sistema também poderá utilizar *email* para que usuários possam liberar suas senhas, pois assim não será necessário que o usuário esteja *online* para que ele libere sua parte da senha.

Referências Bibliográficas

[Ahmed e Umrysh, 2002] Ahmed, K. Z. Umrysh, C. E. (2002). *Desenvolvendo Aplicações Comerciais em Java com J2EE e UML*. Ciência Moderna, 1 ed.

[Asmuth e Bloom, 1983] Asmuth, C. Bloom, J. (1983). A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, Vol.29, No.2, pp.208–210.

[Blakley, 1979] Blakley, G. R. (1979). Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, volume 48, pages 313–317. American Federation of Information Processing Societies.

[Buholzer, 1999] Buholzer, M. (1999). Secret sharing with threshold. University of Applied Sciences - Biel/Bienne - Switzerland.

[Coutinho, 2003] Coutinho, S. C. (2003). *Números Inteiros e Criptografia RSA*. Computação e Matemática. IMPA, 2 ed.

[de Rezende Rocha, 2003] de Rezende Rocha, A. (2003). Camaleão: um software para segurança digital utilizando esteganografia.

[Deitel e Deitel, 2001] Deitel, H. M. Deitel, P. J. (2001). *Java, Como Programar*. Bookman, 3 ed.

[Elmasri e Navathe, 1994] Elmasri, R. Navathe, S. B. (1994). *Fundamentals of database systems*. Addison-Wesley, 2 ed.

[Foundation, 2005] Foundation, T. E. (2005). Eclipse sdk. Disponível em `http://www.eclipse.org/`.

[FSF, 2005] FSF (2005). Free software foundation. Disponível em `http://www.fsf.org/`.

[Gomes, 2000] Gomes, O. J. A. (2000). *Segurança Total: Protegendo-se Contra os Hackers*. Makron Books, 1 ed.

[Hoffman, 1973] Hoffman, L. J. (1973). *Security and Privacy in Computer Systems*. Melville Publishing Company, 1 ed.

[Júnior, 1977] Júnior, H. K. (1977). *Segurança de Dados em Computação*. Livros Técnicos e Científicos Editora S.A., 1 ed.

[Kahn, 1996] Kahn, D. (1996). *The CODEBREAKERS: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner.

[Karnin et al., 1983] Karnin, E. D.; Greene, J. W.; Hellman, M. E. (1983). On secret sharing systems. *IEEE Transactions on Information Theory*, Vol.29, No.1, pp.35–41.

[Knuth, 1981] Knuth, D. E. (1981). *The Art of Computer Programming*. Addison-Wesley Publishing Company, 2 ed.

[Knuth et al., 1990] Knuth, D. E.; Graham, R. L.; Patashnik, O. (1990). *Concrete Mathematics*. Addison-Wesley Publishing Company, 6 ed.

[Loewenguth, 2004] Loewenguth (2004). Le chiffre des templiers. Disponível em `http://lwh.free.fr/pages/algo/crypto/templier.htm` Acessado em 15 de fevereiro de 2005.

[Microsystems, 2005] Microsystems, S. (2005). Java api specification. Disponível em `http://java.sun.com/`.

[MySQL, 2005] MySQL (2005). Mysql database. Disponível em `http://www.mysql.com/`.

[RSA, 2000] RSA, L. (2000). Rsa laboratories frequently asked questions about today's cryptography. Disponível em `http://www.rsasecurity.com` Acessado em 07 de março de 2005.

[Schneier, 1996] Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, 2 ed.

[Shamir, 1979] Shamir, A. (1979). How to share a secret. *Communications of the ACM*, Vol.22, No.11, pp.612–613.

[Tanenbaum, 2003] Tanenbaum, A. S. (2003). *Computer Networks*. Prentice Hall, 4 ed.

[Terada, 2000] Terada, R. (2000). *Segurança de Dados: Criptografia em redes de computador*. Edgard Blücher LTDA, 1 ed.

[Westin, 1967] Westin, A. F. (1967). *Privacy and Freedom*. Atheneum.

[Zhou, 2002] Zhou, L. (2002). System security - secret sharing. Disponível em `http://www.cs.cornell.edu/Courses/cs513/2000SP/SecretSharing.html` Acessado em 08 de fevereiro de 2005.

Apêndice A

MODELAGEM DO BANCO DE DADOS

A seguir, será mostrada a modelagem do banco de dados. Modelagem entidade relacionamento na figura A.1.

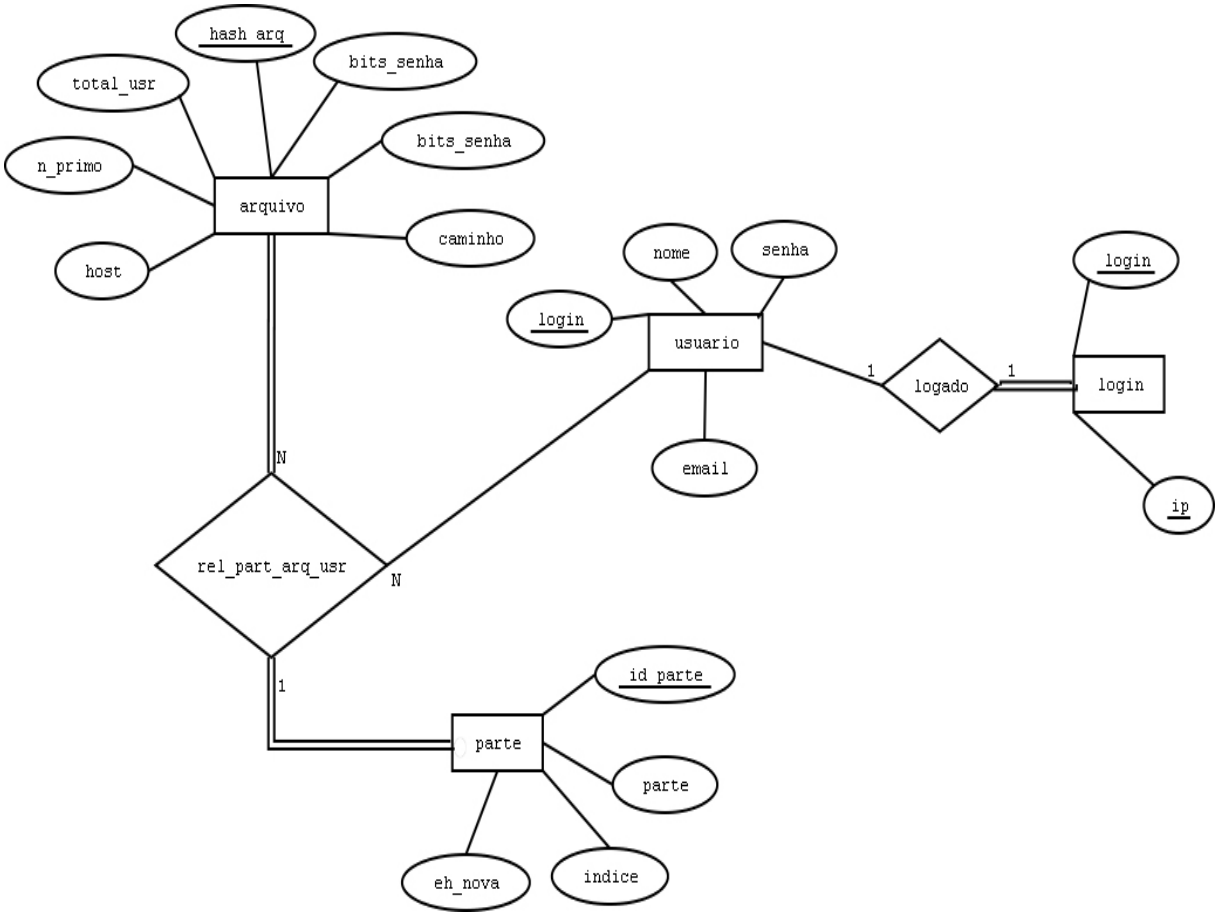

Figura A.1: Modelagem entidade relacionamento do banco de dados.

Apêndice B

DETALHAMENTO DOS CASOS DE USO

A seguir, são apresentados os casos de uso do sistema em mais detalhes.

ID do caso de uso: 001
Nome: Entrar no sistema
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite que o ator realize *login* no sistema e tenha suas operações liberadas para uso.
Execução normal: • Ator preenche o dado referente ao seu *login*; • Ator preenche dado referente à sua senha; • Entra no sistema.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Ator preencher *login* e / ou senhas incorretamente; - Banco de dados estiver inoperante.

ID do caso de uso: 002
Nome: Gerar s*tring* criptografada.
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Faz com que uma *string* criptografada seja gerada poder manter as informações de *login* e senha do banco de dados seguras.
Execução normal: • Ator preenche o campo com a string real a ser criptografada; • O sistema retorna a *string* criptografada.
Execução anormal: • Não há execução anormal.

ID do caso de uso: 003
Nome: Alterar senha
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite ao usuário mudar sua senha.
Execução normal: • Ator preenche os campos com a senha antiga, a nova senha e a confirmação da nova senha;
Execução anormal:

• O sistema adverte o usuário se algum campo for preenchido de maneira incorreta; • O sistema irá retornar uma mensagem de erro se o banco de dados estiver inoperante.

ID do caso de uso: 004
Nome: Encerrar sessão
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite ao ator finalizar sua sessão no sistema.
Execução normal: • Ator indica que vai realizar o fim da sessão.
Execução anormal: • O sistema irá retornar uma mensagem de erro se o banco de dados estiver inoperante.

ID do caso de uso: 005
Nome: Bloquear arquivos e pastas
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite ao ator escolher um arquivo ou pasta para ser protegido.
Execução normal: • Ator escolhe arquivo ou pasta para proteger; • Ator escolhe o número de usuários que irão partilhar a senha, o número mínimo de usuários necessários para recuperar a senha e o tamanho da senha;
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Ator preencher alguma informação de forma incorreta; - Banco de dados estiver inoperante. - Arquivo ou pasta está sendo usado por outro processo.

ID do caso de uso: 006
Nome: Desbloquear arquivos e pastas
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite liberar um arquivo que já foi protegido anteriormente.
Execução normal: • Ator escolhe arquivo ou pasta para ser liberado; • Ator escolhe quais usuários que irão receber a solicitação da liberação da senha; • Usuários respondem à solicitação; • Ator escolhe se deseja manter arquivo protegido e suas informações no banco de dados.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Ator escolher um arquivo inválido ou que está sendo utilizado por outro processo;

- Banco de dados estiver inoperante; - Não existem usuários suficientes para liberar o acesso ao arquivo antigo; - Há algum tipo de falha na conexão entre os usuários.

ID do caso de uso: 007
Nome: Gerenciar partes das senhas.
Casos de uso utilizados: - **Ator(es):** Administrador, Usuário
Descrição: Permite listar as partes da senha que o ator está partilhando no momento.
Execução normal: • Ator visualiza as partes das senhas que está partilhando.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante;

ID do caso de uso: 008
Nome: Remover parte da senha
Casos de uso utilizados: Gerenciar partes das senhas **Ator(es):** Administrador, Usuário
Descrição: Ator remove da base de dados a parte da senha que está partilhando
Execução normal: • Ator remove a parte das senhas que está partilhando.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante;

ID do caso de uso: 009
Nome: Detalhar parte da senha
Casos de uso utilizados: Gerenciar partes das senhas **Ator(es):** Administrador, Usuário
Descrição: Permite ao ator visualizar as informações detalhadas de sua parte da senha e do arquivo que ela protege.
Execução normal: • Ator detalha a parte das senhas que está partilhando.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante;

ID do caso de uso: 010
Nome: Listar usuários
Casos de uso utilizados: - **Ator(es):** Administrador
Descrição: Lista os usuários cadastrados no sistema naquele momento.
Execução normal:

• Ator lista os usuários cadastrados no sistema.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante;

ID do caso de uso: 011
Nome: Remover usuário
Casos de uso utilizados: Listar usuários **Ator(es):** Administrador
Descrição: Remove um usuário cadastrado no sistema.
Execução normal: • Ator remove a parte das senhas que está partilhando.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante;

ID do caso de uso: 012
Nome: Alterar usuário
Casos de uso utilizados: Listar usuários **Ator(es):** Administrador
Descrição: Altera informações de um usuário cadastrado no sistema.
Execução normal: • Ator preenche os novos dados (*email* e nome) pertinentes ao usuário em questão.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante; - Dados preenchidos forem inválidos.

ID do caso de uso: 013
Nome: Cadastrar usuário
Casos de uso utilizados: - **Ator(es):** Administrador
Descrição: Cadastra informações de um novo usuário no sistema.
Execução normal: • Ator preenche os novos dados (*login*, senha, *email* e nome) do novo usuário.
Execução anormal: • O *EasyShare* irá retornar uma mensagem de erro se: - Banco de dados estiver inoperante; - Dados preenchidos forem inválidos.