ADRIANA DE SOUZA GUIMARÃES

PROPOSTA DE UM SISTEMA DE APRESENTAÇÃO DE SENHAS EM AMBIENTES HOSTIS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de mestre em Ciência da Computação.

Área de concentração: Banco de Dados

Orientador: Prof. Dr. João Nunes de Souza

UBERLÂNDIA - MG
SETEMBRO DE 2008

Dados Internacionais de Catalogação na Publicação (CIP)

G963p Guimarães, Adriana de Souza, 1970-

Proposta de um sistema de apresentação de senhas em ambientes hostis / Adriana de Souza Guimarães. - 2008.

172 f. : il.

Orientador: João Nunes de Souza.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia.

1. Computadores - Medidas de segurança - Teses. 2. Criptografia - Teses. I. Souza, João Nunes de. II.Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

ADRIANA DE SOUZA GUIMARÃES

PROPOSTA DE UM SISTEMA DE APRESENTAÇÃO DE SENHAS EM AMBIENTES HOSTIS

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de mestre em Ciência da Computação.

Área de concentração: Banco de Dados

Uberlândia, 10 de setembro de 2008

Banca Examinadora:

Prof. Dr. João Nunes de Souza - UFU (Orientador)

Profa. Dra. Gina Maira Barbosa de Oliveira - UFU

Prof. Dr. Paulo Sergio Licciardi Messeder Barreto - USP

Aos meus pais
José Mário e Divina
com amor...

Agradecimentos

Agradeço primeiramente a Deus, pela força nas horas difíceis e de indecisão.

A Nossa Senhora, que me iluminou e protegeu em todos os momentos.

Ao professor Dr. João Nunes de Souza, pela orientação, paciência e contribuição para meu aperfeiçoamento profissional.

A minha família: meus pais e meus irmãos pelo apoio; minhas tias Lena, Lourdes, Luzia, meus avós João e Maria por me acolherem em sua casa durante meus estudos.

Ao meu amigo Saulo, pelo apoio e ajuda nos momentos mais complicados e desesperadores.

A todos que contribuíram de alguma forma para a realização desse Mestrado.

Resumo

Senhas usadas para autenticação de usuários e proteção de informações têm sido alvo freqüente de *crackers* que roubam informações para se beneficiarem financeiramente. Com isso, houve uma grande disseminação de programas maliciosos com o objetivo de roubar dados ou informações com o intuito de praticar fraudes. Isto causa muita preocupação, pois o prejuízo é alto. De acordo com o CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil - de janeiro a dezembro de 2007, a fraude foi o 2.º incidente mais reportado (28%).

Esta dissertação propõe o desenvolvimento de um sistema de apresentação de senhas que possa ser usado em máquinas que não sejam seguras, de forma que o usuário prove conhecer a senha sem ter que apontá-la ou digitá-la explicitamente durante o processo de comunicação. Esse sistema baseia-se na Teoria dos Códigos, principalmente no conceito de síndrome e arranjo padrão. A síndrome é uma codificação gerada a partir da senha e é usada para autenticação do usuário, cuja senha está armazenada em um arranjo padrão. O sistema visa proteger a senha de pessoas que tentem roubá-la para uso indevido.

Palavras-chave: senha, síndrome, autenticação

Abstract

Passwords used in order to authenticate users and to information protection have frequently being crackers' target, that steal information for financial benefit. So, a great dissemination of malicious software had happened with the objective of stealing data or information typed by keyboard (in some cases, clicked by mouse), sent by internet for some destiny, in order to practice fraud. This kind of incident causes many worry, because damages may be great. According with CERT.br - Center for Studies, Treatment of Incident Response and Security in Brazil - fraud was the second more reported incident (28%) from january to december of 2007.

This thesis proposes the development of a password presentation system to be used in insecure machines, whose purpose is that the user proves he knows the password without needing to type or to click it explicitly during the communication process. This system is based on Coding Theory, mainly on syndrome and standard array concept. Syndrome is a coding generated by password and is used to authenticate user, whose password is stored in the standard array. This system intends to proctect the password from people trying to steal it to make undue use.

Key-words: password, syndrome, authentication

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma senha (*password*) na Internet, ou em qualquer sistema computacional, serve para autenticar o usuário, ou seja, é utilizada no processo de verificação da identidade do usuário, assegurando que este é realmente quem diz ser [Cert, 2007].

Atualmente, em sistemas que necessitam de senhas para permitir acesso a seus dados, os usuários precisam digitar sua senha em um teclado ou apontá-la com o *mouse* em uma tela contendo caracteres. Em sistemas bancários, por exemplo, o usuário digita a senha ou escolhe os números ou letras que a compõem. Suponha que um usuário esteja digitando sua senha em um terminal e que exista um espião observando as teclas que o usuário digita, situação representada pela figura 1.1. Dessa forma, é possível que o espião consiga descobrir a senha do usuário somente pela observação das teclas utilizadas. Outra forma de descobrir a senha é quando o espião observa o canal de transmissão da senha e tenta capturá-la no momento da transmissão, ou ainda, o espião simplesmente observa e captura os arquivos do servidor que contém a senha para tentar descobri-la.

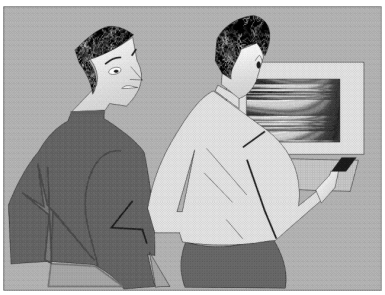

Figura 1.1: Espião observando usuário digitar senha [Febraban, 2007]

1.1 Sistemas de Autenticação Existentes

A maioria dos sistemas de autenticação de usuários necessitam da digitação da senha em um teclado numérico ou apontamento com *mouse* em um teclado virtual. A simples visualização do que está sendo digitado ou apontado por uma pessoa estranha pode revelar o conteúdo da senha.

O teclado numérico é muito simples de observar. Pela posição das teclas é possível verificar o que está sendo digitado e assim obter a senha, havendo apenas a necessidade de conseguir a identificação do usuário, um número de cartão eletrônico, por exemplo. Além disso, um programa espião pode capturar a posição das teclas pressionadas, salvar em um arquivo e enviar, via internet, à pessoa que o criou ou instalou no computador do usuário.

No teclado virtual é também fácil de observar o que está sendo apontado. Apesar de que, em alguns casos, o teclado é variável, isto é, a posição dos caracteres na tela é alterada a cada clique, existem programas que capturam a tela, e em seguida enviam por e-mail para a pessoa que o criou ou instalou no computador do usuário. Isso torna possível visualizar onde os cliques com *mouse* foram feitos.

Além da senha, muitos sistemas necessitam de um código de acesso para que transações sejam efetuadas. Nesses casos, o nível de segurança é maior, pois sem o código não há acesso ao sistema. Abaixo, alguns exemplos de código de acesso utilizados em sistemas bancários são citados:

- Em sistemas de *internet banking*, o usuário informa sua senha através de um teclado virtual variável que possui dois caracteres por botão e muda de lugar na tela a cada acesso. Esse procedimento não impede a descoberta de senha, se a digitação for observada por uma pessoa mal intencionada. Ou, uma senha eletrônica deve ser utilizada, mas nesse caso, existe um teclado virtual que varia as posições das letras e números. A senha eletrônica é diferente da senha do cartão eletrônico que identifica o usuário, mas mesmo assim a observação permite a descoberta do que está sendo escolhido pelo usuário, pois a senha é explicitamente usada.

- Em alguns sistemas usados em terminais bancários, o código de acesso é formado por caracteres que representam as letras do alfabeto. Existem 5 botões

na tela, em cada botão estão 5 letras posicionadas aleatoriamente. O usuário escolhe o botão que contém a letra correspondente ao seu código de acesso, e nova tela aparece com as letras em posições diferentes para a escolha da letra do código de acesso. Normalmente três letras formam o código de acesso. Em outros sistemas, no lugar de letras isoladas, sílabas são usadas como código de acesso. Exemplo desse tipo de código de acesso é XA, PSI, RO. Mesmo assim, se a pessoa mal intencionada observa a digitação da senha e a escolha do código de acesso é possível determinar a senha e o código de acesso. No caso da senha, basta observar a posição das teclas digitadas. No caso do código de acesso, basta observar a opção escolhida e verificar quais letras fazem parte dessa opção.

- Há sistemas que utilizam números tanto para a senha como para o código de acesso. Nesse caso, são apresentados na tela, 5 botões, cada botão com dois números, por exemplo, *2 ou 9*. O usuário escolhe o botão que contém o número de sua senha, se tiver o número 2 na senha ou o número 9, o botão que contém *2 ou 9* será selecionado. Como as escolhas são na ordem que os números correspondem a senha, se alguém estiver observado os botões selecionados, a pessoa com intenção de roubar a senha terá uma possibilidade muito maior de descobri-la, pois existem 10 números a serem utilizados, de 0 a 9, e como estão disponibilizados dois a dois, e a senha corresponde a um número ou ao outro, basta então fazer uma combinação de números.

Portanto, há várias formas de apresentação de senhas que são utilizadas pelos sistemas atuais. Além disso, tais sistemas também utilizam várias formas de armazenamento das senhas nos servidores.

Propomos, neste trabalho, uma nova forma de armazenamento de senhas no servidor, como também uma estratégia de apresentação de senhas pelos usuários.

Propomos um sistema de armazenamento de senhas que se fundamenta na teoria de códigos corretores de erros. Nesse sistema, a estratégia de apresentação de senhas utiliza a forma na qual tais senhas estão armazenadas e, portanto, também utiliza fundamentos da teoria de códigos. A idéia principal é apresentar conjuntos de vetores que contêm, além da informação da senha, outras informações que podem confundir um espião ou qualquer pessoa que tente descobri-la. No contexto da teoria dos

códigos, tais informações são representadas por ruídos.

Além disso, a utilização da teoria de códigos permite que a informação seja protegida durante a transmissão, pois não é a senha que é transmitida e sim uma informação que liga a senha ao usuário.

Observamos que a teoria de códigos tem sido utilizada para codificação de diversos tipos de informação. Mas, sua principal utilização é na detecção e correção de erros na transmissão de dados [Hamming, 1950].

O foco da proposta apresentada é utilizar um ambiente computacional em três níveis contendo um servidor seguro, um servidor parcialmente seguro e o cliente, conforme mostra a Figura 1.2. No nível seguro, temos um servidor de senhas que pode até mesmo não estar conectado à rede externa, e com um alto nível de segurança. Além disso, usuários comuns não têm acesso a ele. No servidor seguro, temos a definição dos parâmetros usados no sistema de autenticação, que só podem ser alterados pelo administrador do sistema e as matrizes de geração do código linear. No segundo nível, o servidor parcialmente seguro é o servidor que fará a autenticação do usuário e está conectado à rede externa. Nele usaremos a teoria de códigos para fazer a autenticação. Portanto, ele possui a tabela de identificação do usuário, usada na autenticação e também o arranjo que contém as informações relacionadas à senha do usuário. O nível do cliente é considerado hostil. Nesse nível, usaremos o que denominamos categoria e faremos a montagem dos vetores de apresentação que o usuário visualiza.

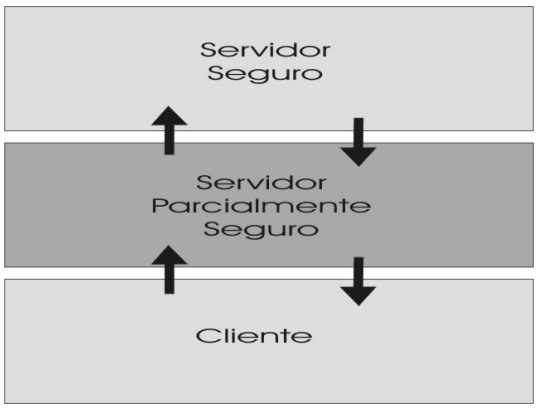

Figura 1.2: Níveis do Sistema Computacional

Essa divisão em três níveis garante maior segurança ao sistema, pois somente

o servidor parcialmente seguro será acessado por usuários comuns e, mesmo que algum espião consiga invadi-lo, não comprometerá as senhas dos usuários, que estão ali armazenadas. Para resolver o problema da invasão, basta mudar as matrizes da teoria dos códigos. Nesse caso, não é necessário mudar a senha do usuário, pois para matrizes diferentes, geramos códigos lineares diferentes. A autenticação se dá por meio de uma informação associada à senha e não da senha propriamente dita. Essa informação pode ser alterada a qualquer momento, sem a necessidade de mudar a senha. Assim, se houver a mudança da informação relacionada à senha, quando o usuário for autenticar-se, a mudança é informada a ele.

1.2 Motivação

A segurança é um grande problema para usuários e sistemas que utilizam serviços de senhas como bancos, comércio eletrônico e sistemas corporativos [Schneier, 2004].

Como o roubo de senhas tem aumentado muito nos últimos anos e causado muitos transtornos, há a necessidade de implementar modelos de armazenamento e apresentação de senhas cada vez mais seguros. Como o aumento da segurança implica em dificultar a descoberta de senhas, isso pode acarretar no aumento da dificuldade de utilização para o usuário.

Desenvolver um sistema de armazenamento e apresentação de senhas seguro e fácil de usar é a principal motivação para a realização desse trabalho. Nosso objetivo é melhorar as características de segurança, já que a senha não é transmitida e nem utilizada explicitamente, e como conseqüência não pode ser capturada.

1.3 Objetivo e Organização da dissertação

Este trabalho tem por objetivo desenvolver um sistema de armazenamento e apresentação de senhas que possa ser usado em máquinas que não sejam seguras, de forma que o usuário prove conhecer a senha sem ter que apontá-la ou digitá-la explicitamente durante o processo de comunicação.

Propomos um sistema de fácil utilização, cuja segurança possa ser analisada matematicamente.

A análise da segurança é uma parte importante desse estudo, pois permite verificar se o sistema é viável ou não em termos de segurança. Esse estudo tem como fundamento a análise de segurança de sistemas criptográficos apresentada em [Goldwasser, 2001].

Esta dissertação está organizada da seguinte forma:

- o capítulo 2 descreve os fundamentos matemáticos utilizados nessa dissertação;
- a proposta é apresentada no capítulo 3;
- no capítulo 4, fazemos a análise de segurança do sistema proposto;
- as nossas conclusões e sugestões de trabalhos futuros são apresentadas no capítulo 5;
- no apêndice A, encontra-se o código-fonte do sistema desenvolvido.

Capítulo 2

Fundamentos Teóricos

Este capítulo trata das notações, dos fundamentos matemáticos, dos conceitos da teoria dos códigos corretores de erros e demais fundamentos teóricos necessários para a formação da base para o desenvolvimento deste trabalho.

2.1 Conjunto

Um conjunto S é definido como sendo uma coleção de elementos $\{a_1, a_2, ..., a_n\}$. Dependendo do número de elementos contidos em S este é dito ser um conjunto finito ou infinito. Se um elemento a_i pertence ao conjunto S, representamos isso como $a_i \in S$ ou $a_i \in \{a_1, a_2, ..., a_n\}$ [Filho, 1998].

Seja S um conjunto. Uma operação binária "$\circ$" sobre elementos de S é uma regra que associa 2 elementos a e b de S a um terceiro elemento c, único, definido por $c = a \circ b$ e $c \in S$. Neste caso, a operação "$\circ$" é dita ser fechada em S. Além disso, uma operação binária "$\circ$" é dita ser associativa se, e somente se, para quaisquer elementos a, b e c do conjunto S, podemos expressar a relação $(a \circ b) \circ c = a \circ (b \circ c)$. Se para todo a, $b \in S$, temos $a \circ b = b \circ a$, a operação é comutativa [Filho, 1998].

2.2 Grupo

Sejam G um conjunto não vazio e $(x, y) \longmapsto x * y$ uma lei de composição interna de G. Dizemos que G é um grupo [Domingues, 1982] em relação a essa lei se, e somente se,

1. $a * (b * c) = (a * b) * c, \forall a, b, c \in G$, isto é, vale a propriedade associativa;

2. existe $e \in G$ de maneira que $a * e = e * a = a, \forall a \in G$, ou seja, existe elemento neutro;

3. todo elemento de G é simetrizável em relação à lei considerada:
   $\forall a \in G, \exists a' \in G | a * a' = a' * a = e$.

2.3 Anel

Um anel é um conjunto A munido de duas operações

$$\begin{array}{c} + : A \times A \longrightarrow A \\ (a, b) \mapsto a + b \end{array} \qquad \text{e} \qquad \begin{array}{c} \cdot : A \times A \longrightarrow A \\ (a, b) \mapsto a \cdot b \end{array}$$

chamadas adição e multiplicação, que possuem as seguintes propriedades [Abramo, 2002]:

A1) Associatividade da adição:

$$\forall a, b, c \in A, (a + b) + c = a + (b + c).$$

A2) Existência de elemento neutro para adição:

Existe um elemento chamado zero e denotado por 0, tal que

$$\forall a \in A, a + 0 = 0 + a = a.$$

A3) Existência de elemento inverso para adição:

Dado $a \in A$, existe um elemento chamado simétrico de a e denotado por $-a$, tal que

$$a + (-a) = -a + a = 0.$$

A4) Comutatividade da adição:

$$\forall a, b \in A, a + b = b + a.$$

M1) Associatividade da multiplicação:

$$\forall a, b, c \in A, (a \cdot b) \cdot c = a \cdot (b \cdot c).$$

M2) Existência de elemento neutro para multilicação:

Existe um elemento chamado unidade e denotado por 1, tal que

$$\forall a \in A, a \cdot 1 = 1 \cdot a = a.$$

M3) Comutatividade da multiplicação:

$$\forall a, b \in A, a \cdot b = b \cdot a.$$

M4) Distributividade da multiplicação em relação à adição:

$$\forall a, b, c \in A, a \cdot (b + c) = a \cdot b + a \cdot c.$$

2.4 Corpo

Um corpo é uma estrutura algébrica que consiste de um conjunto K e duas operações binárias [Koblitz, 1994]:

- Adição: $+ : K \times K \rightarrow K$
- Multiplicação: $\cdot : K \times K \rightarrow K$

Um elemento a de um anel A é dito invertível se existir um elemento $b \in A$ tal que $a \cdot b = 1$. Nesse caso, b é um inverso de A.

Um anel onde todo elemento não nulo é invertível é chamado de corpo.

Exemplo

Um exemplo de corpo é o *Corpo de Galois*, $GF(p)$, que pode ser o conjunto $A = \{0, 1\}$, onde $p = 2$, munido das seguintes operações [Abramo, 2002]:

$+$	0	1
0	0	1
1	1	0

e

$\cdot$	0	1
0	0	0
1	0	1

Assim, para determinado primo p, o corpo finito de ordem p, $GF(p)$, é definido como o conjunto Z_p de inteiros $\{0, 1, \cdots, p - 1\}$, juntamente com as operações ariméticas de módulo p [Stallings, 1998].

2.5 Matriz

Seja o quadro retangular de números como segue:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

A é uma matriz denotada por $A = [a_{ij}]$, onde $i = 1, ..., m$, $j = 1, ..., n$. A é uma matriz $m \times n$, sendo m o número de linhas e n o números de colunas da matriz [Lipschutz, 1994].

2.5.1 Adição de Matrizes

Se $A = [a_{ij}]$ e $B = [b_{ij}]$ são matrizes do mesmo tamanho, então a **soma** $A + B$ é a matriz obtida adicionando-se os elementos correspondentes das matrizes A e B. Ou seja,

$$A + B = [a_{ij} + b_{ij}],$$

onde a notação à direita significa que o elemento da i-ésima linha e j-ésima coluna da matriz $A + B$ é $a_{ij} + b_{ij}$ [Edwards, 2000].

2.5.2 Multiplicação de Matrizes

Sejam as matrizes $A = [a_{ij}]$ e $B = [b_{ij}]$ tais que o número de colunas de A seja igual ao número de linhas de B, isto é, A é uma matriz $m \times p$ e B uma matriz $p \times n$. Então o produto $C = AB$ é a matriz $m \times n$ cujo elemento de ordem ij se obtém multiplicando a i-ésima linha A_i de A pela j-ésima coluna B_j de B, onde $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ip}b_{pj}$ é a matriz C resultante [Lipschutz, 1994].

2.5.3 Matriz Transposta

A transposta de uma matriz, denotada por A^T, é a matriz obtida escrevendo-se as linhas de A, em ordem, como colunas:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Se $A = (a_{ij})$ é uma matriz $m \times n$, então A^T é a matriz $n \times m$ em que $A^T = (a_{ji})$ para todo i e j [Lipschutz, 1994].

2.5.4 Matriz Quadrada e Matriz Identidade

Uma *matriz quadrada* é uma matriz que tem o mesmo número de linhas e de colunas. Uma matriz quadrada $n \times n$ se diz de ordem n [Lipschutz, 1994].

A matriz quadrada de ordem n com $1s$ na diagonal principal, elementos (a_{ii}), e $0s$ nas demais posições denotada por I_n é chamada *matriz identidade* [Lipschutz, 1994].

2.6 Vetores

Os vetores do plano ou do espaço são representados por segmentos orientados. Todos os segmentos orientados que têm a mesma direção, o mesmo sentido e o mesmo comprimento são representados por um mesmo vetor. Os segmentos orientados AB e CD determinam o mesmo vetor v, e escreve-se $v = \overrightarrow{AB} = \overrightarrow{CD}$ [Steins, 1987].

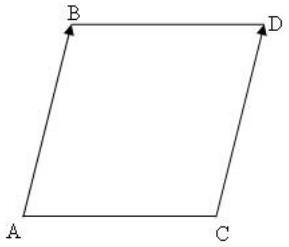

Figura 2.1: Representantes do Vetor v

2.6.1 Espaço Vetorial

Um espaço vetorial é um conjunto não-vazio V de objetos, chamados vetores, sobre os quais estão definidas duas operações, chamadas *soma* e *multiplicação por escalar* (número real), sujeitas aos dez axiomas listados a seguir. Os axiomas precisam valer para todos os vetores u, v e w em V e para todos os escalares c e d [Lay, 1999].

1. A soma de u e v, denotada por $u + v$, está em V.
2. $u + v = v + u$.
3. $(u + v) + w = u + (v + w)$.
4. Existe um vetor **nulo**, **0**, em V tal que $v + 0 = v$.
5. Para cada v em V, existe um vetor $-v$ em V tal que $v + (-v) = 0$.
6. O múltiplo escalar de v por c, denotado por cv, está em V.
7. $c(v + w) = cv + cw$.
8. $(c + d)v = cv + dv$.
9. $c(dv) = (cd)v$.
10. $1v = v$.

O espaço vetorial contém subconjuntos de vetores que formam outros espaços vetoriais menores chamados de subespaço vetorial.

2.6.2 Subespaço Vetorial

Um subespaço de um espaço vetorial V é um subconjunto H de V que satisfaz três propriedades [Lay, 1999]:

1. O vetor nulo de V está em H.
2. H é fechado com respeito à soma de vetores. Isto é, para cada u e v em H, a soma $u + v$ está em H.

3. H é fechado com respeito à multiplicação por escalar. Isto é, para cada v em H e cada escalar c, o vetor cv está em H.

2.6.3 Combinação Linear

De acordo com [Lipschutz, 1994], um vetor v é uma combinação linear de vetores $u_1, u_2, \cdots, u_n$ se existem escalares $k_1, k_2, \cdots, k_n$ tais que

$$v = k_1u_1 + k_2u_2 + \cdots + k_nu_n.$$

2.6.4 Dependência e Independência Linear

Segundo [Lipschutz, 1994], os vetores $u_1, u_2, \cdots, u_n$ de R^n são linearmente dependentes (LD) se existem escalares $k_1, k_2, \cdots, k_n$, não simultaneamente nulos tais que

$$v = k_1u_1 + k_2u_2 + \cdots + k_nu_n = 0.$$

Os vetores $v_1, v_2, \cdots, v_k$ de um espaço vetorial V são linearmente independentes (LI) se a equação

$$c_1v_1 + c_2v_2 + \cdots + c_kv_k = 0$$

tem somente a solução trivial $c_1 = c_2 = \cdots = c_k = 0$. Isto é, a única combinação linear de que representa o vetor nulo 0 é a combinação trivial [Edwards, 2000]:

$$0v_1 + 0v_2 + \cdots + 0v_k$$

Em R^2 ou R^3, um conjunto de dois vetores é linearmente independente se, e somente se, os vetores não estão numa mesma reta quando colocados com seus pontos iniciais na origem [Anton, 2001]. Os vetores v_1 e v_2 da figura 2.2 são linearmente dependentes, pois estão na mesma reta quando colocados com seus pontos iniciais na origem.

Os vetores v_1 e v_2 da figura 2.3 são linearmente independentes, pois não estão na mesma reta quando colocados com seus pontos iniciais na origem.

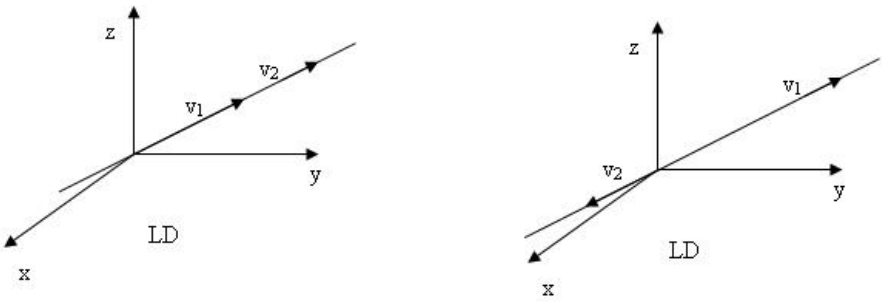

Figura 2.2: Vetores Linearmente Dependentes

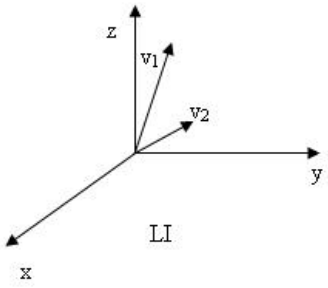

Figura 2.3: Vetores Linearmente Independentes

Em R^3, um conjunto de três vetores é linearmente independente se, e somente se, os vetores não estão num mesmo plano quando colocados com seus pontos iniciais na origem [Anton, 2001]. Os vetores v_1, v_2 e v_3 da figura 2.4 são linearmente dependentes, pois estão no mesmo plano quando colocados com seus pontos iniciais na origem.

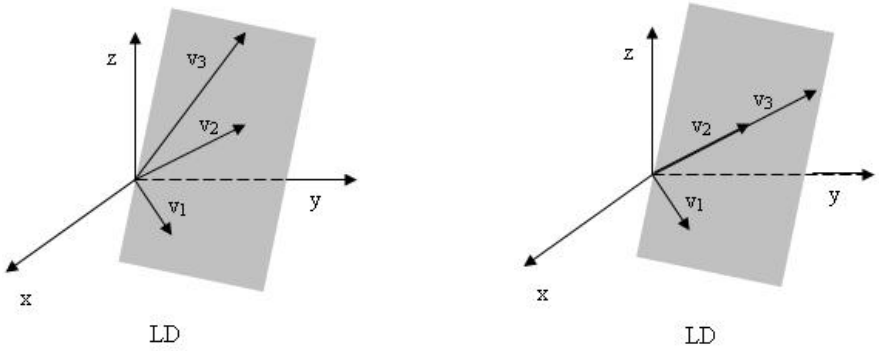

Figura 2.4: Vetores Linearmente Dependentes

Os vetores v_1, v_2 e v_3 da figura 2.5 são linearmente independentes, pois v_1 não está no mesmo plano que v_2 e v_3, quando colocados com seus pontos iniciais na origem.

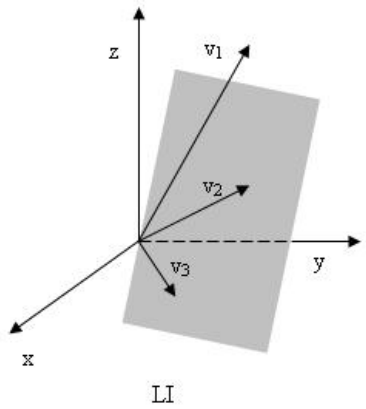

Figura 2.5: Vetores Linearmente Independentes

2.6.5 Base e Dimensão

Um conjunto $S = u_1, u_2, \cdots, u_n$ de vetores é uma *base* de V se valem as duas condições seguintes:

1. $u_1, u_2, \cdots, u_n$ são linearmente independentes,

2. $u_1, u_2, \cdots, u_n$ geram V.

Outra forma de definir uma base é: um conjunto $B = \{u_1, u_2, \cdots, u_n\}$ de vetores é uma *base* de V se todo vetor $v \in V$ pode escrever-se de maneira única como combinação linear dos vetores base [Lipschutz, 1994].

Diz-se que um espaço vetorial V tem *dimensão finita n*, ou que é *n-dimensional*, e se escreve $dimV = n$ se V tem uma base com n elementos.

A base canônica do R^n é formada pelos vetores unitários

$$e_1 = (1, 0, 0, ..., 0), e_2 = (0, 1, 0, ..., 0), \cdots, e_n = (0, 0, 0, ..., 1).$$

Se $x = (x_1, x_2, ..., x_n)$ é um vetor de R^n, então

$$x = x_1e_1 + x_2e_2 + \cdots + x_ne_n.$$

Logo, os vetores e_1, e_2,...,e_n geram R^n [Edwards, 2000].

2.6.6 Produto Escalar

Sejam $u = (u_1, u_2, \cdots, u_n)$ e $v = (v_1, v_2, \cdots, v_n)$ vetores do R^n, o produto escalar, ou produto interno, de u e v, denotado por $u.v$, é o escalar obtido pela multiplicação das componentes correspondentes, somando-se os produtos resultantes:

$$u.v = u_1v_1 + u_2v_2 + \cdots + u_nv_n.$$

Os vetores u e v são ortogonais (ou perpendiculares) se seu produto escalar é zero, isto é, $u.v = 0$ [Lipschutz, 1994].

2.6.7 Transformações Lineares

Sejam V e W espaços vetorais. Uma aplicação $T : V \rightarrow W$ é chamada transformação linear de V em W se:

1. $T(u + v) = T(u) + T(v)$
2. $T(\alpha u) = \alpha T(u)$

para $\forall u, v \in V$ e $\forall \alpha \in R$ [Steins, 1987].

2.7 Teoria dos Códigos

A teoria dos códigos é uma parte da Matemática Aplicada usada em transmissão de informações que utiliza conceitos da Álgebra. Essa teoria é composta de diversos códigos, mas neste trabalho será abordado apenas o Código Linear de Blocos, que é uma classe dos códigos corretores de erros.

Os códigos corretores de erros são importantes no tratamento de informação digitalizada que trafega por canais de comunicação, como, por exemplo, em transmissões de televisão, telefone e navegação na Internet ou mesmo no armazenamento óptico de dados.

Ao enviar mensagens por um canal de comunicação com ruído, podem ocorrer interferências e as mensagens chegarem ao destino com algum erro. No caso de mensagem binária, por exemplo, um bit 0 pode sofrer a interferência e ser alterado para 1. A teoria dos códigos visa detectar e corrigir esses erros. Todo sistema de envio de mensagem pode ser esquematizado na forma da figura 2.6 [Hefez, 1994].

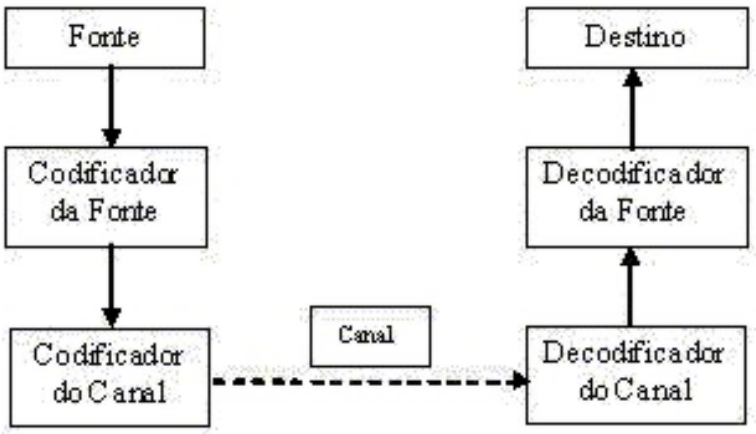

Figura 2.6: Esquema de Transmissão

Os seguintes blocos se destacam em um sistema de informação [Filho, 1998]:

- Fonte: é a origem da informação, na forma de uma mensagem a ser transmitida. Esta mensagem pode se constituir de um texto em uma linguagem natural, voz, bits, um sinal elétrico, ou qualquer outra forma que a informação possa assumir.

- Canal: é o meio através do qual a informação é transmitida, podendo assumir a forma de uma linha telefônica, um enlace de rádio de comunicação, um meio de armazenamento, um organismo biológico, etc. É no canal que o ruído é introduzido na informação, sendo esta uma das grandes preocupações da codificação: a proteção da informação contra erros de transmissão.

- Codificador: é o responsável por um pré-processamento do sinal gerado pela fonte, antes de sua transmissão pelo canal. O resultado da codificação da mensagem é a obtenção de outra mensagem, a mensagem codificada.

- Decodificador: processa a saída do canal, de forma a entregar ao destino uma cópia o mais fiel possível da mensagem emitida pela fonte.

- Destino: representa o usuário para o qual a informação está sendo transmitida, podendo tratar-se de um ser humano, uma máquina ou um organismo vivo qualquer.

Usualmente, divide-se o problema da codificação em dois: codificação da fonte e codificação do canal. A codificação da fonte é a conversão da fonte em código binário, e a codificação do canal é o acréscimo da redundância no código da fonte, transformando um vetor de tamanho k em uma palavra-código de tamanho n, em que $n > k$. Esta divisão traz como vantagem o "isolamento" entre a fonte e o canal, de forma que os problemas podem ser tratados independentemente, através do uso de uma interface padrão para o canal [Filho, 1998].

Os códigos corretores de erros têm como função acrescentar nova informação às mensagens que serão transmitidas fazendo com que elas possam ser corrigidas quando ocorrem erros na transmissão de dados.

2.7.1 Código de Blocos Linear

A teoria dos códigos de correção de erros é baseada em corpos com um número finito de elementos q. Um corpo de q elementos é chamado de corpo de Galois de q elementos e é representado por $GF(q)$ [Abramo, 2002].

Uma mensagem é uma seqüência de bits de tamanho fixo, que é dividida em blocos de mensagens de tamanho k. Esses blocos são codificados, transmitidos e decodificados ao chegarem ao destino.

Poderíamos usar códigos sobre $GF(q)$ para $q > 2$, como por exemplo, q = número primo ou mesmo $q = 2^m$, mas para efeitos didáticos utilizamos o código linear sobre $GF(2)$.

Um código de bloco de tamanho n e 2^k palavras-código é chamado de Código de Bloco Linear (n, k) se e, somente se, suas 2^k palavras-código formam um sub-espaço k-dimensional do espaço vetorial de todas as n-uplas pertencentes a GF(2) [Costello, 1983].

Para efeitos didáticos, utilizamos

Um código linear é um código de blocos que codifica mensagens, isto é, seqüências de bits de tamanho k, em 2^k palavras-código distintas [Hefez, 1994]. No processo

de codificação, bits, denominados redundância, são adicionados aos blocos de mensagens. Após a transmissão da mensagem, o decodificador usa a redundância para corrigir os erros que possam surgir na palavra recebida devido às interferências no canal com ruído.

2.7.2 Codificação

Um código linear é formado a partir de uma matriz geradora G de dimensão $k \times n$, cujas linhas são compostas por k palavras-código linearmente independentes. A soma módulo-2 (ou-exclusivo) de duas palavras-código resulta em outra palavra-código [Lint, 1992].

Como o código linear $C(n,k)$ é um subespaço vetorial k-dimensional do espaço vetorial V_n, então existem k palavras-código linearmente independentes de forma que toda palavra-código de C é uma combinação linear dessas k palavras-código [Lint, 1992].

Uma Matriz geradora G do Código Linear $C(n,k)$ é da forma:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & g_{0,2} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & g_{1,2} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

onde $g_i = (g_{i,0}, g_{i,1}, ..., g_{i,n-1})$ para $0 \leq i < k$ [Costello, 1983].

Seja $u = (u_0, u_1, ..., u_{k-1})$ a mensagem a ser codificada, a palavra-código correspondente será:

$v = u \cdot G$

$$v = (u_0, u_1, ..., u_{k-1}). \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

$v = u_0 g_0 \oplus u_1 g_1 \oplus ... \oplus u_{k-1} g_{k-1}$

Assim, as linhas da matriz G geram o código linear $C(n,k)$.

Mensagens	Palavras-código
000	000000
100	011100
010	101010
110	110110
001	110001
101	101101
011	011011
111	000111

Tabela 2.1: Código Linear com $k = 3$ e $n = 6$

Exemplo

Dada a matriz geradora abaixo e uma mensagem $u = (011)$ a ser codificada:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

A palavra-código v é:

$v = u \cdot G$

$v = 0 \cdot g_0 \oplus 1 \cdot g_1 \oplus 1 \cdot g_2$

$v = 101010 \oplus 110001$

$v = 011011$

Um código definido dessa forma é denominado Código de Blocos Linear Sistemático. Os $n - k$ bits da palavra-código (011) são bits de verificação de paridade e os k bits (011) restantes formam a mensagem u sem nenhuma alteração. Um exemplo do código linear é apresentado na tabela 2.1.

O código linear sistemático utiliza as seguintes estruturas:

- Matriz Geradora G;
- Matriz de Verificação de Paridade H;
- Síndrome.

Matriz Geradora G

Uma matriz geradora G é uma matriz formada pela junção de duas matrizes, a matriz identidade $I_{k\times k}$ e a matriz $P_{k\times n-k}$. A matriz identidade é utilizada porque suas linhas são linearmente independentes e forma a base canônica de um espaço vetorial.

Para obter uma matriz P, basta fazer somas lineares das colunas da matriz identidade. Dada, por exemplo, a matriz identidade $I_{3\times 3}$ a seguir:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1. A primeira coluna da matriz P é a soma da primeira, terceira e quarta colunas da matriz I;

2. A segunda coluna da matriz P é a soma da primeira, segunda e terceira colunas da matriz I;

3. A terceira coluna da matriz P é a soma da segunda, terceira e quarta colunas da matriz I;

A matriz P resultante é dada por:

$$P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Observe que as colunas de P aparecem em uma ordem que depende das combinações lineares consideradas. A matriz G é da forma $G = [PI]$. Portanto, de acordo com o exemplo acima, G é dada por:

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Verificação de Paridade H

Associado a todo código linear de blocos, existe uma outra matriz, denominada matriz de verificação de paridade H. Para toda matriz $G_{k\times n}$, a matriz $H_{(n-k)\times n}$, com $n-k$ linhas linearmente independentes, é tal que $H = [I_{n-k}P^T]$. Uma n-upla v é uma palavra-código gerada pela matriz G, se e somente se, $v.H^T = 0$. As 2^{n-k} combinações lineares de linhas da matriz H formam o código linear $C_d(n, n-k)$, que é denominado **código dual** [Lint, 1992]. Nesse caso, temos:

$$G \cdot H^T = 0.$$

E, se

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

então

$$P^T = \begin{bmatrix} p_{0,0} & p_{1,0} & \cdots & p_{k-1,0} \\ p_{0,1} & p_{1,1} & \cdots & p_{k-1,1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

e

$$H = \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \cdots & 0 & p_{0,0} & p_{1,0} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & p_{0,1} & p_{1,1} & \cdots & p_{k-1,1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{array}\right]$$

De acordo com nosso exemplo, a matriz de verificação de paridade para o código linear $C(6,3)$ é dada por:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

2.7.3 Distância Mínima e Peso mínimo de um Código de Blocos Linear

O ponto de partida para a construção de um código corretor de erros é um conjunto finito A, chamado de *alfabeto*. O número de elementos de A, denotado por $|A|$, é simbolizado por q [Abramo, 2002], isto é, $q = |A|$. Além do alfabeto, a distância mínima de um código linear é um parâmetro que determina a capacidade de detecção e de correção de erros de um código. A distância é definida a partir do peso de Hamming.

Peso de Hamming de uma n-upla v, denotado por $W(v)$, é o número de componentes diferentes de zero [Costello, 1983].

Exemplo

Se $v = (011001)$, então $W(v) = 3$.

Distância de Hamming entre duas n-uplas u e v, denotada por $d(u, v)$, é o número de posições nas quais elas diferem [Lint, 1992].

Exemplo

Para as n-uplas $u = (010010)$ e $v = (100011)$, a distância de Hamming é $d(u, v) = 3$, pois diferem nas posições 2, 3 e 7.

Seja C um código. A *distância mínima* de C é o número d_{min} tal que

$$d_{min} = min\{d(u, v) : u, v \in C \text{ e } u \neq v\}$$

ou seja, é o menor número de posições em que u e v diferem [Abramo, 2002].

Exemplo

Em $\{0, 1\}^3$, temos

$$d(001, 111) = 2$$
$$d(000, 111) = 3$$
$$d(100, 110) = 1$$

Nesse exemplo, temos que $d_{min} = 1$.

Se um código é tal que $d_{min} = 3$, então a menor número de posições em que os vetores do código diferem é 3. Podem diferir em mais posições, mas precisam diferir em pelo menos 3 posições.

A distância de Hamming é uma métrica que satisfaz a desigualdade triangular. Sejam v, w e x três n-uplas. Então,

$$d(v, w) + d(w, x) \geq d(v, x) \tag{2.1}$$

Pelas definições de distância de Hamming e soma módulo-2, a distância entre duas n-uplas, v e w, é igual ao peso de Hamming da soma de v e w. Assim,

$$d(v, w) = W(v + w) \tag{2.2}$$

e

$$d_{min} = min\{W(v + w) : v, w \in C, v \neq w\} \tag{2.3}$$

Se C é um código linear de blocos, a soma de dois vetores é também uma palavra-código. Segue de 2.2 que a distância de Hamming entre duas palavras-código em C é igual ao peso de Hamming de uma terceira palavra-código em C. Então, segue de 2.3 que

$$d_{min} = min\{W(x) : x \in C, x \neq 0\} \tag{2.4}$$

Isto é, $d_{min} = W_{min}$, onde $W_{min} = \{W(x) : x \in C, x \neq 0\}$. O parâmetro W_{min} é o *peso mínimo* do código linear C.

Teorema 1: A distância mínima de um código linear de blocos é igual ao peso mínimo de suas palavras-código não-zero [Lint, 1992].

Teorema 2: seja C um código linear (n, k) com matriz de verificação de paridade H. Para cada palavra-código com peso de Hamming igual a l, existem l colunas de H, tais que o vetor soma dessas l colunas é igual ao vetor zero. Inversamente, se existem l colunas de H cujo vetor soma é o vetor zero, existe um vetor código de peso de Hamming l em C [Lint, 1992].

2.7.4 Síndrome

Dados um código C com matriz de verificação de paridade H e um vetor $v \in V^n$, chamamos o vetor $v \cdot H^T$ de *síndrome* de v [Abramo, 2002]. A síndrome de um vetor recebido r é o vetor de n - k bits [Fernandez, 2005]:

$$s = r \cdot H^T$$

onde:

H é matriz de paridade do código C;
r é a n-upla recebida (o vetor);
s é síndrome de r.

Seja $v \in K^n$, então a *classe lateral* de v segundo C é dada por [Abramo, 2002]

$$v + C = \{v + c : c \in C\}.$$

Lema 5: Os vetores u e v têm a mesma síndrome se, e somente se, [Abramo, 2002]:

$$u \in v + C$$

Demonstração: $Hu^t = Hvt \iff H(u-v)^t = 0 \iff u - v \in C \iff u \in v + C$.

Proposição 8 [Abramo, 2002]: Seja C um (n, k)-código linear. Temos que

1. $v + C = v' + C \Leftrightarrow v - v' \in C$;
2. $(v + C) \cap (v' + C) \neq \emptyset \Longrightarrow v + C = v' + C$;
3. $\bigcup_{v \in K^n} (v + C) = K^n$;
4. $|(v + C)| = |C| = q^k$.

Segue de (2) e (4) que o número de classes laterais, segundo C, é $\frac{q^n}{q^k} = q^{n-k}$.

Exemplo: Seja C o $(4, 2)$-código gerado sobre F_2 pela matriz

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Logo, $C = \{0000, 1011, 0101, 1110\}$, e as classes laterais segundo C são

$$0000 + C = \{0000, 1011, 0101, 1110\}$$
$$1000 + C = \{1000, 0011, 1101, 0110\}$$
$$0100 + C = \{0100, 1111, 0001, 1010\}$$
$$0010 + C = \{0010, 1001, 0111, 1100\}$$

A síndrome é utilizada tanto na decodificação da palavra-código recebida como também na detecção de erros de transmissão. Isso é feito pelo algoritmo para decodificação.

Algoritmo para Decodificação

Seja H a matriz de verificação de paridade do código C e seja r um vetor recebido [Abramo, 2002]:

1. Calcule $r.H^t$;
2. Se $r \cdot H^t = 0$, aceite r sendo a palavra transmitida e fim;
3. Se $r \cdot H^t = s$ e $s \neq 0$, compare s com as colunas de H;
4. Se existirem i e α, tais que $s^t = \alpha h^i$ para $\alpha \in K$, faça e o vetor cuja i-ésima entrada é igual a i-ésima entrada de α e as demais são iguais a zero. Corrija r, fazendo $c = r - e$ e fim;
5. Se o contrário de (4) ocorrer, então mais de um erro foi cometido.

2.7.5 Capacidade de Detecção e Correção de Erros

Quando uma palavra-código v é transmitida por um canal com ruído, se o padrão de erros com l erros é adicionado, a palavra recebida r difere da palavra v transmitida em l posições. Logo, $d(v, r) = l$ [Lint, 1992].

Suponha que a mensagem $u = (u_1, \cdots, u_k)$ seja codificada na palavra-código $v = (v_1, \cdots, v_n)$, que é enviada por um canal. Por causa do ruído do canal, a palavra recebida $r = (r_1, \cdots, r_n)$ pode ser diferente de v. Definimos vetor-erro como

$$e = r - v = (e_1, \cdots, e_n)$$

em que $e_i = 0$ tem probabilidade $1 - p$, e $e_i = 1$ tem probabilidade p. Em geral, p é tal que $0 \leq p < \frac{1}{2}$ [Costello, 1983].

A figura 2.7 sintetiza os princípios básicos da codificação por blocos em GF(2) [Barroso, 2007]:

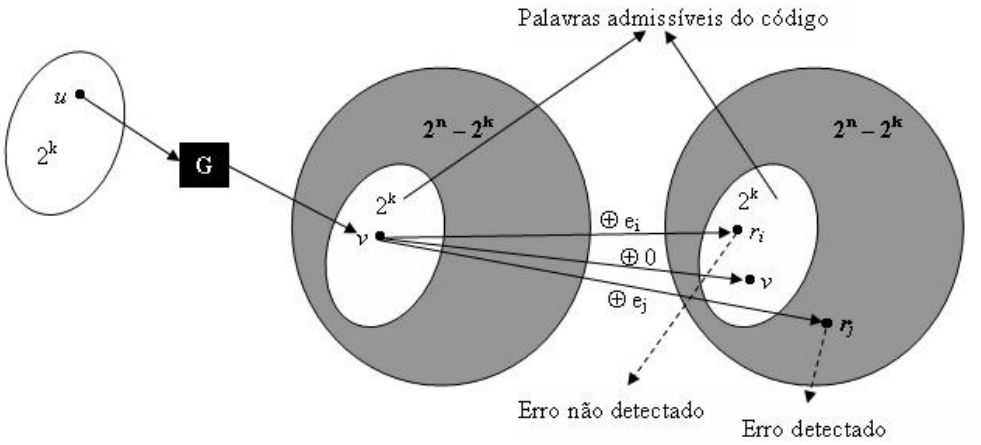

Figura 2.7: Codificação e Detecção de Erro

onde,

$$e = [e_0, \cdots, e_i, \cdots, e_n],$$

$$e = \begin{cases} 1, \text{ se houver erro no bit i} \\ 0, \text{ caso contrário} \end{cases}$$

Se o vetor recebido não é uma palavra do código C, dizemos que erros foram detectados. Um código de bloco com distância mínima d_{min} é capaz de detectar todos os padrões de $d_{min} - 1$ ou menos. Entretanto, ele não pode detectar todos os padrões de erros de d_{min} erros porque existe pelo menos um par de palavras do código que diferem em d_{min} posições e existe um padrão de erros de d_{min} erros que levaram uma à outra. Por essa razão, dizemos que a capacidade de detecção de erro randômica de um código de bloco com distância mínima d_{min} é $d_{min} - 1$ [Lint, 1992].

De fato, um código linear (n, k) é capaz de detectar $2^n - 2^k$ padrões de erros de tamanho n. Entre os $2^n - 1$ possíveis padrões de erro não-zero, existem $2^k - 1$ padrões de erro que são idênticos às $2^k - 1$ não-zero palavras-código. Se qualquer um desses padrões de erros ocorrem , ele altera a palavra-código v transmitida em outra

palavra-código w. Assim, w será recebida e sua síndrome é zero. O decodificador aceita w como a palavra-código transmitida e assim faz uma decodificação incorreta. Existem $2^k - 1$ padrões de erro indetectáveis. Existem exatamente $2^n - 2^k$ padrões de erro que não são idênticos às palavras-código de um código linear (n, k). Esses padrões de erros são detectáveis [Lint, 1992].

Exemplo

A probabilidade do vetor-erro ser 0 é dada por $\mathrm{P}[e_i = 0] = (1 - p)$ e a probabilidade do vetor-erro ser 1 é dada por $\mathrm{P}[e_i = 1] = p$, então:

$\mathrm{P}[e = 000000] = (1 - p)^6$;

$\mathrm{P}[e = 010000] = p(1 - p)^5$;

$\mathrm{P}[e = 100010] = p^2(1 - p)^4$.

Em geral, se v é algum vetor fixo de peso a, então:

$$\mathrm{P}[e = v] = p^a(1 - p)^{n-a}$$

como $p < \frac{1}{2}$, temos $1 - p > p$ e $(1 - p)^6 > p(1 - p)^5 > p^2(1 - p)^4 > \cdots$

Nesse caso, r é decodificado como o mais próximo de v, isto é, usa-se o vetor erro e que tem o menor peso. Isso é chamado de *decodificação do vizinho mais próximo* [Costello, 1983].

Seja $C(n, k)$ um código linear. Seja A_i o número de palavras-código de peso i em C. Os números $A_0, A_1, \cdots, A_n$ são chamados *distribuição de peso* de C. Seja $P_n(E)$ a probabilidade de um erro não detectado, então

$$P_n(E) = 1 - \textstyle\sum_{i=0}^{n} A_i p^i (1 - p)^{n-1}$$

onde p é a probabilidade de transição do canal. Se a distância mínima de C é d_{min} então A_1 até A_{dmin-1} são zero [Lint, 1992].

Exemplo

Considere o código $(6, 3)$ dado anteriormente. A distribuição de peso desse código é $A_0 = 1$, $A_1 = A_2 = 0$, $A_3 = 4$, $A_4 = 3$ e $A_5 = A_6 = 0$. A probabilidade de não detectar um erro é

$$P_n(E) = 1 - (1-p)^6 + 4p^3(1-p)^3 + 3p^4(1-p)^2.$$

Se $p = 10^{-2}$, essa probabilidade é aproximadamente 9×10^{-2}.

A distância mínima de um código pode ser ímpar ou par. Seja t um inteiro positivo tal que

$$2t + 1 \leq d_{min} \leq 2t + 2 \tag{2.5}$$

Vamos mostrar que o código C é capaz de corrigir todos os padrões de erro de t ou menos erros. Seja v e r os vetores transmitido e recebido, respectivamente. Seja w qualquer outro vetor em C. As distâncias de Hamming entre v, r e w satifazem a desigualdade triangular:

$$d(v, r) + d(w, r) \geq d(v, w) \tag{2.6}$$

Suponha que um padrão de erros com t' erros ocorre durante a trasmissão de v. Então, o vetor r recebido difere de v em t' posições e $d(v, r) = t'$. Como v e w são vetores em C, temos

$$d(v, w) \geq d_{min} \geq 2t + 1 \tag{2.7}$$

Combinando 2.5 e 2.6 e usando o fato de que $d(v, r) = t'$, obtemos a seguinte desigualdade:

$$d(w, r) \geq 2t + 1 - t'.$$

Se $t' \leq t$, então

$$d(w, r) > t.$$

Essa desigualdade diz que se um padrão de erros com t ou menos erros ocorre, então o vetor recebido r está mais próximo (em distância de Hamming) do vetor transmitido v do que a qualquer outro vetor w em C [Lint, 1992]. Isso ocorre porque $d(v, r) = t'$, $d(w, r) > t$ e $t' \leq t$.

Sejam e_1 e e_2 dois padrões de erro que satisfazem as seguintes condições:

1. v e w são vetores transmitidos tais que $d(v, w) = d_{min}$;

2. $e_1 + e_2 = v + w$;

3. e_1 e e_2 não têm componentes não-zero em posições comuns.

Obviamente, temos

$$W(e_1) + W(e_2) = W(v + w) = d(v, w) = d_{min} \tag{2.8}$$

Suponha que v é trasmitido e é corrompido por um padrão de erro e_1. O vetor recebido é

$$r = v + e_1.$$

A distância de Hamming entre v e r é

$$d(v, r) = W(v + r) = W(e_1) \tag{2.9}$$

A distância de Hamming entre w e r é

$$d(w, r) = W(w + r) = W(w + v + e_1) = W(e_2) \tag{2.10}$$

Agora, suponha que o padrão de erro e_1 contém mais do que t erros, isto é, $W(e_1) > t$. Como $2t + 1 \leq d_{min} \leq 2t + 2$ segue de 2.8 que

$$W(e_2) \leq t + 1 \tag{2.11}$$

Combinando 2.9 e 2.10 e usando o fato de que $W(e_1) > t$ e $W(e_2) \leq t + 1$, obtemos:

$$d(v, r) \geq d(w, r) \tag{2.12}$$

Essa desigualdade diz que dado um padrão de erros e_1 com $l(l > t)$ erros, então o vetor recebido $v + e_1$ é mais próximo a um vetor-código incorreto w do que ao vetor transmitido u [Lint, 1992].

Um código de bloco com distância mínima d_{min} garante a correção de todos os padrões de erro de $t = \left\lfloor \frac{(d_{min} - 1)}{2} \right\rfloor$ ou menos erros. O parâmetro $t = \left\lfloor \frac{(d_{min} - 1)}{2} \right\rfloor$ é denominado *capacidade de correção de erro aleatório* do código [Lint, 1992].

2.7.6 Arranjo Padrão

O arranjo padrão é uma forma de escrever os vetores de V^n e serve como ferramenta de decodificação. Ele contém vetores do espaço vetorial V^n, dispostos em uma matriz, onde cada linha é uma classe lateral do espaço dos vetores do código linear.

A construção de um arranjo padrão é feita da seguinte forma: os 2^k vetores do código linear são colocados na primeira linha de uma matriz de tamanho $2^{n-k} \times 2^k$. Escolha um vetor e_2 que não pertença à primeira linha do arranjo. A segunda linha do arranjo é dada pelo conjunto: $\{e_2 + v_j;\ 0 \leq j \leq 2^k;\ v_j$ ocorre na primeira linha$\}$. Escolha um vetor e_3 que não pertença à primeira e segunda linhas do arranjo. A terceira linha do arranjo é dada pelo conjunto: $\{e_3 + v_j;\ 0 \leq j \leq 2^k;\ v_j$ ocorre na primeira linha$\}$, e assim por diante. Construído dessa forma, os vetores da primeira coluna são 2^{n-k} vetores do subespaço complementar ao espaço dos vetores do código [Lint, 1992].

Além disso, a soma linear módulo-2 dos vetores da primeira coluna com os vetores da primeira linha geram os outros vetores e definem o arranjo [Lint, 1992].

Os vetores do arranjo, representados na Tabela 2.2, formam o espaço vetorial R^6, pois nele estão todos os vetores binários de 000000 a 111111. Além disso, ele possui 8 classes laterais dos vetores do código linear $C(6, 3)$. Isso significa que, para qualquer espaço vetorial de dimensão n, o arranjo conterá os 2^n vetores do espaço, espalhados em 2^k colunas e 2^{n-k} linhas. As linhas do arranjo são denominadas classes laterais e o primeiro elemento da n-upla e_i de cada classe lateral é chamado de líder da classe lateral [Lint, 1992]. Os vetores que pertencem à mesma classe lateral possuem a mesma síndrome.

O procedimento para correção de erro, através do arranjo padrão é simples e pode ser descrito da seguinte forma [Costello, 1983]:

1. O vetor recebido r deve ser localizado no arranjo padrão;
2. Em seguida, o vetor decodificado v, será aquele que fica na primeira linha da mesma coluna onde se encontra o vetor recebido r;
3. O padrão de erro associado ao vetor recebido é o vetor localizado na primeira coluna na mesma linha do vetor recebido (líder da classe lateral).

Linha	Líder	Vetores do Código Linear						
1	000000	110001	101010	011011	011100	101101	110110	000111
2	000001	110000	101011	011010	011101	101100	110111	000110
3	000010	110011	101000	011001	011110	101111	110100	000101
4	000100	110101	101110	011111	011000	101001	110010	000011
5	001000	111001	100010	010011	010100	100101	111110	001111
6	010000	100001	111010	001011	001100	111101	100110	010111
7	100000	010001	001010	111011	111100	001101	010110	100111
8	001001	111000	100011	010010	010101	100100	111111	001110

Tabela 2.2: Arranjo com Oito Classes Laterais

Existem outros códigos da Teoria que poderiam ser utilizados tanto para armazenamento quanto para apresentação de senha, mas nosso estudo tinha como objetivo usar o código de blocos linear.

2.8 Segurança de um Sistema

Dizemos que um sistema é **seguro** se seus recursos forem usados e acessados conforme o pretendido, sobre todas as circunstâncias [Silberschatz, 2004].

Os sistemas computacionais têm 3 objetivos principais, do ponto de vista da segurança [Tanenbaum, 2007]:

- Confidencialidade dos dados: é manter secreto dados secretos. O sistema deve garantir que não ocorra liberação dos dados para pessoas não autorizadas.
- Integridade dos dados: garantir que usuários não autorizados não devem ser capazes de modificar os dados sem a permissão do proprietário. Modificar significa alteração, mas também remoção e inclusão de dados.
- Disponibilidade do sistema: garantir que o sistema esteja disponível para seu usuário legítimo.

Um sistema é considerado seguro se os 3 objetivos principais são garantidos. Privacidade é um outro aspecto do problema de segurança. Ela deve garantir proteção ao indivíduo contra o mau uso da informação sobre ele [Tanenbaum, 2007].

Em muitas aplicações, garantir a segurança de um sistema computacional requer um esforço considerável. Grandes sistemas comerciais contendo dados de folha de pagamento ou dados financeiros são alvos que atraem ladrões [Silberschatz, 2004].

2.8.1 Definição de Segurança

Dizemos que um sistema é seguro se qualquer adversário, que pode ver o texto cifrado e sabe qual algoritmo de criptografia foi usado, não consegue recuperar o texto original por inteiro [Goldwasser, 2001]. Além disso,

1. deve ser difícil recuperar mensagens do texto cifrado quando as mensagens são obtidas de distribuições de freqüência de letras na língua;

2. deve ser difícil computar informação parcial sobre mensagens do texto cifrado;

3. deve ser difícil detectar fatos simples, mas úteis sobre o tráfego de mensagens, tais como o envio da mesma mensagem duas vezes;

4. as propriedades acima devem ocorrer com alta probabilidade.

2.8.2 Canal de Transmissão de Dados

O canal de transmissão de dados é outro requisito importante de segurança. Um canal é considerado seguro quando oferece confiabilidade. O TCP (*Transmission Control Protocol*) foi projetado para oferecer um fluxo de bytes fim a fim confiável em uma inter-rede não-confiável [Tanenbaum, 2003].

O serviço TCP é obtido quando tanto o transmissor quanto o receptor criam pontos extremos chamados soquetes. Cada soquete tem um número de soquete (endereço) que consiste no endereço IP (*Internet Protocol*) do host e em um número de 16 bits local para esse host, chamado porta. Para que o serviço TCP funcione, é necessário que uma conexão seja explicitamente estabelecida entre um soquete da máquina transmissora e um soquete da máquina receptora [Tanenbaum, 2003].

Muitas aplicações da Internet são baseadas no TCP/IP. O TCP/IP é baseado em conexões - uma conexão é feita, a comunicação ocorre e, então, a conexão é desativada. Exatamente como uma chamada telefônica. O servidor cria um *socket*

com um número de porta (chamada de porta efêmera) e aguarda que um cliente após a criação do *socket* do servidor, crie o seu *socket* utilizando a mesma porta onde se estabelece a conexão [Morgan, 2000].

2.9 Ataques à Segurança de um Sistema

Os ataques à um sistema podem ser de dois tipos:

- Ataque Passivo: não há modificação nem fabricação de informação, nesse caso, o atacante apenas observa a informação e procura por informação sensível, como senha por exemplo.
- Ataque Ativo: há modificação e/ou fabricação de informação.

Ataques passivos são ataques de observação ou monitoração da transmissão de dados. Nesse caso, o objetivo do atacante é obter a informação que está sendo transmitida [Stallings, 1998].

Ataques ativos envolvem a modificação da informação que está sendo transmitida ou a criação de uma informação falsa [Stallings, 1998]. O objetivo do atacante é transmitir uma informação falsa.

A análise de segurança do sistema proposto nesse trabalho é em relação a ataques passivos, pois tratando-se de senhas de acesso o grande problema considerado é a observação da senha, para posterior uso da mesma, com a intenção de obter acesso à conta do usuário.

2.10 Famílias de Função

Conforme [Goldwasser, 2001], uma família de função é um mapeamento

$$F : Keys(F) \times Dom(F) \rightarrow Range(F)$$

onde $Keys(F)$ é um conjunto de chaves de F; $Dom(F)$ é o domínio de F; e $Range(F)$ é o contradomínio de F.

Em famílias de funções são considerados $Keys(F) = \{0,1\}^k$ e $Dom(F) = \{0,1\}^l$ e $Range(F) = \{0,1\}^L$ para alguns valores inteiros de $k, l, L \geq 1$, isto é, $Keys(F)$, $Dom(F)$ e $Range(F)$ são *strings* binárias de tamanho k, l, L respectivamente. Algumas vezes, domínio e contradomínio podem ser conjuntos de diferentes tipos, contendo *strings* de tamanhos variados [Goldwasser, 2001].

Considere, por exemplo, F_K uma instância da família de funções F, definida pela escolha randômica de uma chave $K \in Keys(F)$. Nesse caso, cada chave K identifica ou aponta um membro dessa família. Assim, dados K e x como entrada, F_K retorna o mapeamento que associa a cada valor x tal que $x \in Dom(F)$ a um valor y tal que $y \in Range(F)$. Nesse caso, $y = F_K(x)$, ou seja, $y = F(K; x)$. A operação que seleciona uma *string* randômica no conjunto de chaves $Keys(F)$ é denotada por $K \xleftarrow{R} Keys(F)$. De forma análoga, $f \xleftarrow{R} F$ faz de f a função F_K onde K é uma chave randômica.

Se $l = L$ e a função F é injetora, isto é, se domínio e contradomínio são iguais, e a função é um-para-um dentro do mesmo conjunto, então $F_K(x)$ é chamada de permutação.

Segundo [Goldwasser, 2001], uma função $F : \{0,1\}^n \rightarrow \{0,1\}^n$ é uma função randômica se tem a seguinte propriedade: $\forall x \in \{0,1\}^n$, o valor y, tal que $F(x) = y$ e $y \in \{0,1\}^n$, é escolhido randômicamente [Pass,2006].

Exemplo

Os sistemas criptográficos *DES - Data Encryption Standard* e *AES - Advanced Encryption Standard* são famílias de permutações, visto que:

- $Keys(DES) = \{0,1\}^{56}$, $Dom(DES) = \{0,1\}^{64}$ e $Range(DES) = \{0,1\}^{64}$
- $Keys(AES) = \{0,1\}^{128}$, $Dom(AES)\{0,1\}^{128}$ e $Range(AES)\{0,1\}^{128}$.

No nosso exemplo de aplicação, temos:

$$Keys(F) = \{0,1\}^{24},\ Dom(F) = \{0,1\}^{32} \text{ e } Range(F) = \{0,1\}^{56}$$

2.11 Resumo

Nesse capítulo foram apresentados os fundamentos teóricos que são utilizados nessa dissertação, principalmente a Teoria de Códigos que é a base para a proposta apresentada.

Capítulo 3

Proposta e Implementação de um Sistema de Apresentação de Senha

Esse capítulo apresenta a proposta de um sistema para apresentação de senhas por usuários que utilizam terminais inseguros e a implementação de uma aplicação usada como exemplo.

3.1 Sistema de Autenticação de Usuários

Nesta seção, a proposta de um sistema de apresentação de senhas baseado na teoria dos códigos é apresentada. Essa proposta é denominada Sistema de Apresentação de Senhas em Máquinas Hostis. O sistema é uma aplicação Cliente/Servidor, na qual temos um servidor de senhas conectado a vários terminais cliente. Vamos, inicialmente, contextualizar o problema. Em seguida, o funcionamento do sistema é apresentado. Na seção seguinte, falamos sobre a implementação de uma aplicação usada como exemplo.

3.1.1 Contexto do Problema

Os usuários de sistemas computacionais Cliente/Servidor acessam seus dados por meio de uma técnica chamada autenticação do usuário. A técnica mais comum para autenticar um usuário é o uso de uma identificação para o usuário, que chamamos de ID, e uma senha [Silberschatz, 2004].

Na maioria dos sistemas, a autenticação é feita por meio da leitura de um cartão

magnético ou com código de barras, ou por meio de digitação da identificação do usuário via um terminal, seguida da digitação da senha ou do uso do *mouse* para selecionar os caracteres da senha. A senha é enviada ao servidor por um canal de comunicação, TCP/IP por exemplo, e o servidor autentica se a identificação e a senha estiverem corretas. Essa forma de autenticação pode gerar alguns problemas, como:

- a senha pode ser capturada no momento da digitação ou seleção pelo *mouse* por programas maliciosos, como o *keylogger* ou *screenlogger*;
- a senha pode ser interceptada por pessoa mal intencionada durante sua transmissão do terminal do usuário para o servidor;
- o servidor pode ser invadido e ter o arquivo de senhas roubado.

Sabemos que a criptografia é largamente utilizada para manter o sigilo da senha, mas como os sistemas para quebra de senhas estão cada vez mais sofisticados, propomos uma nova forma de apresentação da senha na qual ela não é transmitida, nem digitada e muito menos apontada com o *mouse* explicitamente. A senha, nesse caso, sofre uma modificação, uma codificação que chamamos de síndrome e que fica cadastrada no banco de dados.

O sistema que está sendo proposto baseia-se na teoria dos códigos, descrita no capítulo 2, e usa os conceitos de síndrome e arranjo para autenticar o usuário. Dividimos o sistema em níveis. No primeiro nível, está um servidor seguro que contém as senhas dos usuários. Dizemos que o servidor é seguro, pois ele pode não está conectado à rede externa e possui todos os requisitos necessários à segurança do sistema. Esse servidor possui também a definição dos parâmetros usados na teoria de códigos, bem como as matrizes geradora e de verificação de paridade. No segundo nível, temos um servidor parcialmente seguro, que usa a teoria de códigos para autenticação do usuário. Dizemos parcialmente seguro, porque o servidor está conectado a vários terminais cliente na rede externa, portanto, é um servidor passível de sofrer acesso não autorizado. Sabemos que todo computador ligado à internet e com acesso externo, não é totalmente seguro. Esse servidor parcialmente seguro armazena o arranjo e a tabela de identificação do usuário. No terceiro nível, temos os terminais cliente, que exibem ao usuário os vetores de apresentação para sua

autenticação. Nesse nível, temos armazenadas as categorias dos usuários do sistema, que são representadas por imagens de objetos e que são usadas na montagem dos vetores de apresentação. Essa forma de divisão do sistema faz parte da proposta de um sistema seguro, que não use a senha explicitamente e que, diferentemente dos sistemas atuais, facilita a vida do usuário, já que ele não precisa memorizar seqüências de números ou letras.

Nesse sistema, ao cadastrar o usuário, uma identificação e uma senha são solicitadas. Elas são armazenadas no servidor seguro e a síndrome é gerada a partir da senha digitada. Tanto a identificação quanto a síndrome são armazenadas no banco de dados que se encontra no servidor parcialmente seguro. A partir daí, a senha não é mais utilizada pelo usuário, o servidor trabalha somente com a identificação do usuário e a síndrome gerada para fazer a autenticação. A síndrome é informada ao usuário.

Em sistema bancário, por exemplo, o funcionário do banco tem autorização para fazer o cadastro, assim, o cliente é atendido por esse funcionário, munido de seus documentos. O funcionário cadastra o cliente, que em seguida digita sua senha em um terminal numérico. Dependendo do banco, a senha vai pelo correio para sua casa. No primeiro acesso do cliente ao caixa eletrônico, um código de acesso é impresso para que o cliente o utilize sempre que for fazer qualquer transação no caixa eletrônico. Além da senha, o código de acesso é necessário para autenticação.

No sistema proposto, uma pessoa fica encarregada de cadastrar os usuários do sistema e assim que o cliente informa sua senha, a síndrome associada à senha é gerada. Em seguida, o sistema informa ao usuário as categorias que ele deve escolher no momento da autenticação. As categorias são conjuntos de imagens que são relacionadas com seqüências de bits que fazem parte da síndrome. Cada categoria possui o mesmo número de imagens que são selecionadas aleatoriamente e apresentadas ao usuário.

Poderíamos optar por usar um protocolo convencional assimétrico de identificação para evitar a necessidade de armazenar a informação secreta, mas como os protocolos assimétricos possuem uma carga de processamento alta, utilizá-los durante a montagem dos vetores de apresentação poderia resultar em um tempo elevado de processamento, o que não é interessante em um processo de autenticação.

Considere, por exemplo, que a síndrome gerada seja composta pelos bits (010000 100100 110000 101110). Nesse caso, o sistema determina que a categoria da parte 010000 é a categoria "Móveis", que a categoria da parte 100100 é a categoria "Trator", que a categoria da parte 110000 é a categoria "A+Número" e que a categoria da parte 101110 é a categoria "Letras gregas". Dessa forma, para informar a síndrome (010000100100110000101110) ao sistema, o usuário deverá selecionar, na primeira tela que aparece em seu terminal, um vetor que contém um objeto que representa um móvel. Na próxima tela, ele deve selecionar um vetor que contém um objeto que representa um trator, na terceira tela, um vetor que contém um objeto que representa a letra "A" concatenada com um número (de 1 a 9) deve ser selecionado e, finalmente, na quarta tela, um vetor que contém um objeto que representa uma letra grega deve ser selecionado.

Além disso, consideramos o sistema composto de dois módulos: um sistema seguro e outro parcialmente seguro. Um servidor que contém as senhas do usuário é o sistema seguro. Um servidor que usa a teoria de códigos é ligado a vários terminais de usuários e é chamado de servidor parcialmente seguro, conforme mostrado na Figura 1.2. Consideramos também, no contexto do problema, que o canal de transmissão dos dados entre o servidor parcialmente seguro e os terminais também é seguro. Isso significa que tanto o nível seguro do servidor, quanto o canal de transmissão têm os três objetivos de segurança garantidos: disponibilidade do sistema, confidencialidade e integridade dos dados. Mas, por outro lado, o nível parcialmente seguro pode, eventualmente, ter seus dados acessados por pessoa não autorizada.

No sistema proposto, as senhas dos usuários são cadastradas previamente no módulo seguro do servidor. As síndromes são armazenadas no servidor parcialmente seguro e as categorias nos terminais de usuários.

3.2 Implementação da proposta

Nessa seção, mostramos a implementação de uma aplicação exemplo, para testarmos a viabilidade do sistema proposto. Descrevemos como foi desenvolvido o cadastro de usuários e a autenticação, bem como a definição dos parâmetros do sistema.

3.2.1 Definição de Parâmetros do Sistema

Na teoria de códigos, os principais parâmetros são a dimensão do espaço e do subespaço vetorial, denotados por n e k respectivamente. São eles que definem o tamanho da palavra-código, o tamanho do código linear e a dimensão da síndrome. Todo código linear é definido como $C(n, k)$ e nesse estudo, o código de blocos linear foi usado como $C(56, 32)$, com $d_{min} = 1$, assim o código consegue detectar e corrigir 1. Usamos a distância de Hamming igual a 1 por questões de carga de processamento, como o código é grande, o processamento também é. A Tabela 3.1 lista os parâmetros usados no sistema desenvolvido como exemplo.

Parâmetro	Valor	Definição
n	56	Número de bits da palavra-código
k	32	Número de bits da mensagem
c	4	Número de partes da síndrome
t	5	Número de vetores em cada conjunto
x	7	Dimensão dos vetores
y	6	Quantidade de bits das partes da síndrome
K	512	Número total de objetos das categorias
cat	8	Quantidade de objetos por categoria

Tabela 3.1: Tabela de Parâmetros do Sistema

As senhas dos usuários pertencem ao espaço de mensagem $\{0, 1\}^{32}$. Portanto, no nosso exemplo, as senhas têm 32 bits, isto é, 4 caracteres de 8 bits cada. Essa senha com 32 bits é transformada em uma palavra-código de 56 bits, formando um código linear $C(56, 32)$. Nesse caso, o código linear de blocos $C(56, 32)$ possui 2^{32} palavras-código.

No nosso exemplo, as palavras-código formam um subespaço 32-dimensional do espaço vetorial de todas as 56-uplas pertencentes a $\{0, 1\}^{56}$. O código linear é formado a partir de uma matriz geradora $G_{32\times56}$, cujas linhas são compostas por 32 palavras-código linearmente independentes. A multiplicação da senha por essa matriz é o que chamamos de codificação da senha. A senha é transformada em uma palavra-código. Observamos que quanto maior os valores de n e k maior é a possibilidade de combinações. Para se ter uma idéia, com $n = 56$, temos um espaço com 2^{56} possíveis vetores, isto é, cerca de $7.205.759.404 \times 10^{16}$ vetores. O código linear, é um subespaço com 2^{32} vetores, isto é, 4.294.967.296 palavras-código. A

senha de um usuário está entre esses possíveis vetores.

Além disso, podemos verificar que nesse espaço vetorial existem 2^{24} síndromes, isto é, 16.777.216 síndromes diferentes, que pertencem a $\{0,1\}^{24}$, o que resulta em uma boa gama de opções. A palavra-código da senha é transformada em síndrome, que é um vetor do espaço $\{0,1\}^{24}$ e, no nosso exemplo, é dividida em 4 partes. Logo, obtemos 4 vetores pertencentes ao subespaço $\{0,1\}^6$. Assim, temos 2^6 possíveis vetores, o que resulta em 64 vetores binários diferentes.

Para representar esses vetores binários, escolhemos imagens que representassem objetos facilmente identificados por um usuário, e criamos conjuntos de objetos comuns, denominados categoria. Como exemplo, podemos citar a categoria "animal", que possui imagens de animais. Cada categoria está relacionada a um vetor do espaço $\{0,1\}^6$. Portanto, cada imagem de uma categoria está associada ao mesmo vetor do espaço $\{0,1\}^6$, conforme mostram as Tabelas 3.4 e 3.5.

No nosso exemplo, temos 64 categorias e cada uma delas possui 8 imagens. Denominamos objetos, as imagens cujo código corresponde a um vetor do espaço $\{0,1\}^6$, que corresponde a uma categoria. Portanto, o exemplo proposto possui 512 objetos que representam as partes das senhas dos usuários do sistema.

No código linear sistemático, a matriz G é formada por $G = [PI_k]$, onde a matriz P é obtida por somas lineares dos bits da matriz identidade I_k, conforme vimos no capítulo 2. Essas somas não devem permitir a repetição de linhas da matriz P nem da matriz P^T, pois caso isso ocorra haverá síndromes iguais para linhas diferentes do arranjo, fato que, de acordo com a Teoria de Códigos, não pode ocorrer. Somente os vetores que possuem a mesma síndrome devem ocupar a mesma linha no arranjo. Logo, são necessárias 2^{n-k} síndromes diferentes. Portanto, no exemplo desenvolvido, existem $2^{(56-32)} = 2^{24}$ síndromes possíveis.

3.2.2 Cadastro de Usuários

Para cadastrar o usuário no sistema proposto, é apresentada uma janela solicitando a identificação, como na Figura 3.1.

Após a digitação da identificação, uma tela similar à primeira aparece solicitando a senha a ser cadastrada. Essa senha é convertida em binário e a palavra-código correspondente à senha é gerada pela teoria dos códigos. A palavra-código

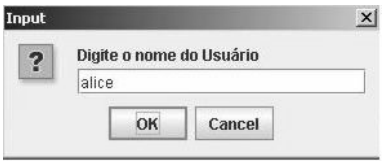

Figura 3.1: Cadastro de Usuário

é gerada multiplicando a senha, em binário, pela matriz geradora G. Em seguida, sua síndrome é calculada. A síndrome é calculada multiplicando a palavra-código pela transposta da matriz de verificação de paridade H. A síndrome resultante é armazenada no banco de dados do módulo parcialmente seguro do servidor juntamente com a identificação do usuário, conforme mostra a Tabela 3.2. Tais dados são armazenados em um módulo parcialmente seguro do servidor, pois caso alguém descubra a síndrome da senha de um usuário, isso não significa o comprometimento total de sua senha, pois conhecer a síndrome não significa conhecer a senha. A senha foi codificada e a síndrome foi obtida dessa codificação.

Identificação do usuário	Síndrome
Usuário1	$(s_0 s_1 s_2 \cdots s_{n-k})$
Usuário2	$(s_0 s_1 s_2 \cdots s_{n-k})$
Usuário3	$(s_0 s_1 s_2 \cdots s_{n-k})$
$\vdots$	$\vdots$

Tabela 3.2: Tabela de Nome de Usuário e Síndrome Arbitrária

Na aplicação desenvolvida como exemplo, a Tabela 3.3 apresenta a identificação dos usuários e suas síndromes.

Identificação do usuário	Síndrome
Alice	(010000 100100 110000 010001)
José	(000010 000110 000101 010111)
Ana	(010000 000000 100001 010101)
Rosa	(011010 000000 000100 001010)
Pedro	(001010 000000 000100 010101)
Maria	(001000 000001 000010 101010)
Joana	(010000 000100 001010 101100)
Paulo	(111010 000000 000100 001010)

Tabela 3.3: Tabela de Nome de Usuário e Síndrome da Aplicação

Se não houver problemas, como duplicidade de usuário, por exemplo, o que não

deve ser permitido, a operação é realizada com sucesso.

Em seguida, os nomes dos objetos associados às partes da síndrome são informados ao usuário. Nessa fase, cada parte da síndrome é associada a um tipo de objeto, que no caso da Figura 3.2 são: peixes, aviões, letra D concatenada com um número e informática. Isso significa que para informar sua senha, o usuário deverá, nas etapas posteriores, escolher, nesta ordem, os vetores que contenham as imagens de peixes, aviões, letra D concatenada com um número e alguma imagem relacionada à informática.

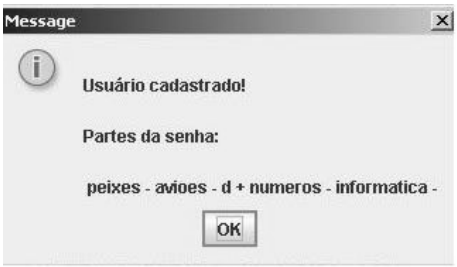

Figura 3.2: Categorias do Usuário Cadastrado

3.2.3 Autenticação de Usuários

No sistema proposto, o cliente envia sua identificação digitando ou inserindo um cartão para leitura. Assim que o servidor recebe essa identificação, ele busca pela síndrome correspondente no seu banco de dados.

Com esses dados, o servidor envia ao usuário uma seqüência de conjuntos de vetores que mostram informações que o usuário relaciona como sua senha. Essas informações podem ser seqüência de caracteres ou qualquer outro tipo de informação. Na aplicação desenvolvida, usamos imagens. As informações são elementos de vetores, que chamamos de vetores de apresentação.

Para uma melhor compreensão do contexto, vamos descrever como os vetores de apresentação são montados.

3.2.4 Vetores de Apresentação da Aplicação

O exemplo a seguir mostra a montagem dos vetores de apresentação do sistema proposto, de acordo com a aplicação implementada como modelo.

No sistema proposto, como exemplo, uma interface pictórica foi escolhida como forma de apresentar informações ao usuário. Nos ambientes modernos, muitas interfaces de usuário são de natureza gráfica, mesmo que a informação exibida seja predominantemente textual. Nas interfaces pictóricas, a própria informação fornecida pelo sistema é de natureza intrinsicamente gráfica [Paula Filho, 2003].

Seguindo essa abordagem, usamos imagens para representar a informação que relaciona o usuário à senha na aplicação, a síndrome faz esse relacionamento. Inicialmente, a síndrome é dividida em c partes. Para cada parte, o servidor apresenta ao usuário um conjunto de vetores com imagens, como indicado na Figura 3.3. Observe que nessa figura temos 5 vetores, todos de dimensão 7. Na notação do sistema proposto, utilizamos a letra t para indicar o número de vetores do conjunto de apresentação. Portanto, $t = 5$. A dimensão dos vetores é denotada pela letra x. No nosso exemplo de aplicação, a síndrome foi dividida em 4 partes, isto é, $c = 4$. Portanto, 4 conjuntos de vetores, como o conjunto da Figura 3.3 são apresentados. Cada conjunto tem 5 vetores, e cada vetor possui 7 elementos (7 imagens).

Figura 3.3: Vetores de Apresentação

Tendo como base sua senha, o usuário escolhe a linha que contém uma imagem associada a uma parte da sua síndrome, clicando no botão numerado que se localiza à esquerda das seqüências de imagens (opções 1, 2 , 3, 4 ou 5). Em seguida, a seleção do usuário é enviada ao servidor parcialmente seguro e nova tela é apresentada ao usuário para outra parte da síndrome. Este processo repete até que todas as partes

da síndrome sejam consideradas.

Considere, por exemplo, o usuário Alice cuja senha está relacionada à síndrome (010000100100110000101110). Nesse exemplo, essa síndrome é dividida em 4 vetores de 6 bits. Indicamos estas partes por: $\vec{p}_1 = 010000$, $\vec{p}_2 = 100100$, $\vec{p}_3 = 110000$ e $\vec{p}_4 = 101110$. Portanto, o servidor apresenta ao usuário 4 conjuntos de vetores de apresentação. No primeiro conjunto, um objeto da categoria de $\vec{p}_1$ é escrito como uma parte de um vetor. No segundo conjunto de vetores, um objeto da categoria de $\vec{p}_2$ é escrito como parte de um vetor do conjunto de vetores de apresentação e assim por diante.

No nosso exemplo de aplicação, a síndrome é dividida em partes que são vetores do espaço $\{0,1\}^6$. Para a 1.^{a} parte da síndrome, um possível conjunto de vetores de apresentação informado ao usuário é indicado na Figura 3.4. Considerando que $\vec{p}_1 = 010000$ está associada à imagem "Móveis", Alice deve escolher a opção 2, pois tem a imagem de uma cadeira. Observe a opção 2 na última posição do vetor.

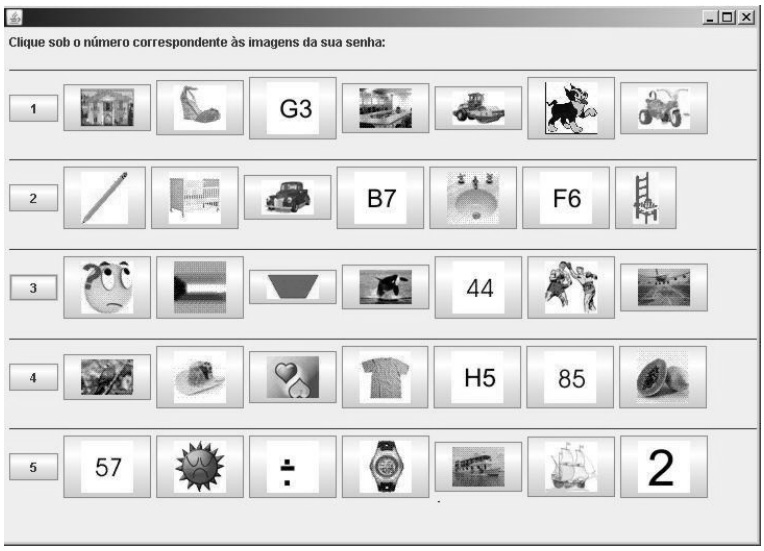

Figura 3.4: Vetor de Apresentação da Parte 1 da Síndrome

Para a 2.^{a} parte da síndrome, um possível conjunto de vetores de apresentação mostrado ao usuário é indicado na Figura 3.5. Alice deve escolher a opção 5, pois tem a imagem de um trator. Isso ocorre porque $\vec{p}_2 = 100100$ é associada à imagem de "Trator". Observe a imagem na 5.^{a} posição do vetor dessa opção.

Para a 3.^{a} parte da síndrome, um possível conjunto de vetores de apresentação

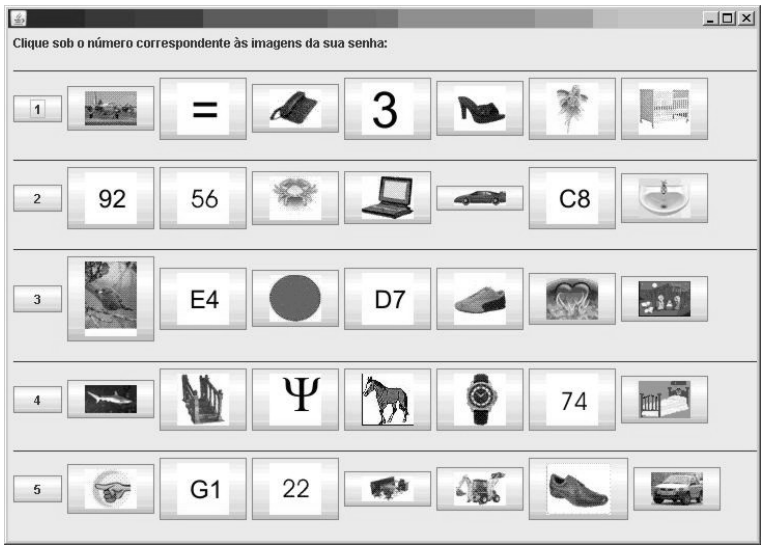

Figura 3.5: Vetor de Apresentação da Parte 2 da Síndrome

mostrado ao usuário é indicado na Figura 3.6. Alice deve escolher a opção 5, que tem a imagem "A4" na 3.^{a} posição do vetor, porque $\vec{p}_3 = 110000$ é associada à imagem "A + números".

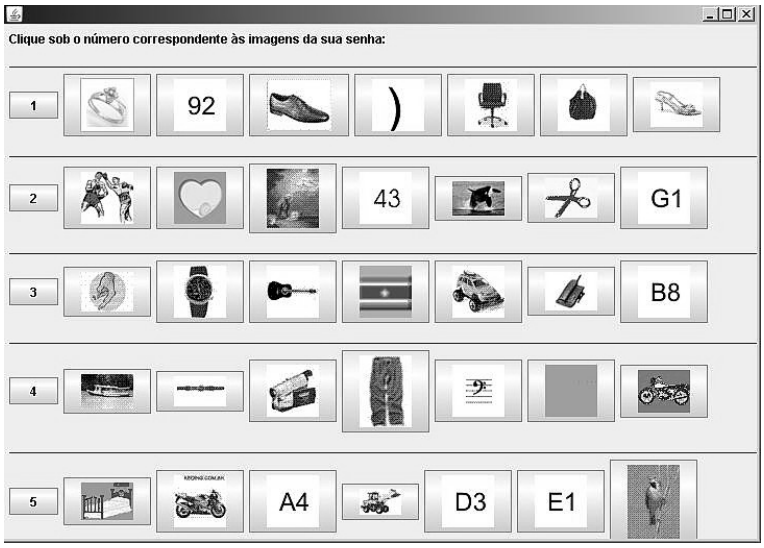

Figura 3.6: Vetor de Apresentação da Parte 3 da Síndrome

Para a 4.^{a} parte da síndrome, um possível conjunto de vetores de apresentação mostrado ao usuário é indicado na Figura 3.7. Alice deve escolher a opção 1, pois $\vec{p}_4 = 101110$ está associada à imagem de letras gregas, e nessa opção temos a imagem do π na 5.^{a} posição desse vetor.

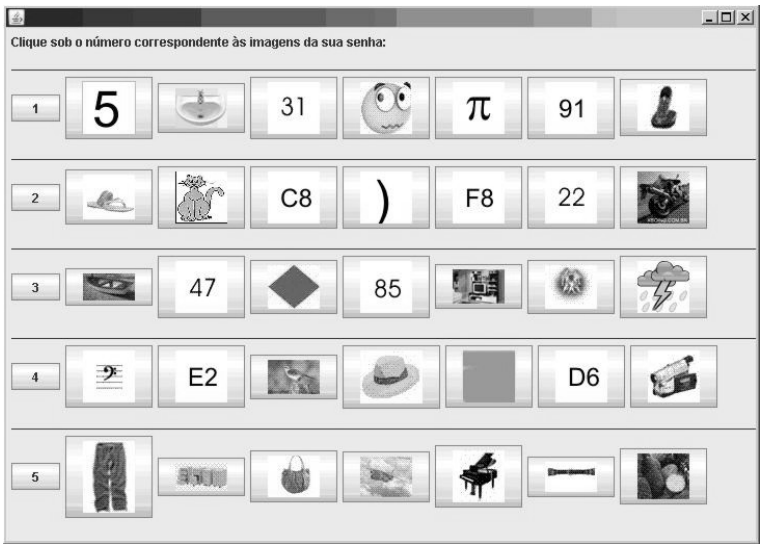

Figura 3.7: Vetor de Apresentação da Parte 4 da Síndrome

As opções escolhidas por Alice, que devem ser 2, 5, 5, 1, são enviadas ao servidor parcialmente seguro, que usa uma função para verificar se as escolhas estão corretas.

O servidor recebe o número i do vetor escolhido pelo usuário, que possui a parte j da síndrome. Seguindo o exemplo anterior, o servidor recebe como valores de i os números 2, 5, 5, 1 e como valores de j as seqüências 1, 2, 3, 4. De posse desses valores, a função que determina se as escolhas estão corretas é definida como:

Para j de 1 até c faça

$$f(i, j) = (i \times x) - 1 \tag{3.1}$$

onde,
i: número do vetor escolhido pelo usuário;
x: dimensão do vetor;

Com o resultado da equação 3.1, a função busca os valores binários armazenados nas posições $f(i, j) - x$ a $f(i, j)$, encontra a classe lateral e verifica se é a mesma classe lateral da síndrome do usuário no arranjo. Depois, busca os valores binários armazenados nas demais posições, encontra a classe lateral e verifica se é a mesma classe lateral das outras partes da síndrome do usuário. Se todos os vetores recebidos tem as mesmas classes laterais da síndrome do usuário, então o acesso é permitido e a autenticação é feita. Mas, se pelo menos uma das classes laterais diferem da síndrome do usuário, o acesso não é permitido, conseqüentemente, a autenticação

não é realizada.

A função busca os vetores binários das imagens que vão da posição $f(i,j) - x$ até a posição do vetor enviado ao usuário e verifica se a parte j da síndrome do usuário pertence a um desses vetores. Se for, então para aquela parte da síndrome a escolha do usuário foi correta. O sistema servidor faz essa verificação para todos os números de vetores recebidos do sistema do usuário e, se todos estiverem corretos, o servidor faz a autenticação do usuário, caso contrário, o servidor avisa o usuário que houve erro e que a autenticação falhou.

3.2.5 Montagem dos Vetores de Apresentação

Para a montagem dos vetores de apresentação foi desenvolvido o algoritmo descrito a seguir:

- A síndrome do usuário é dividida em c partes: $\vec{p}_w = (\vec{p}_1, \vec{p}_2, ..., \vec{p}_c)$.
- Cada parte $\vec{p}_i$ da síndrome é associada a um conjunto de objetos denominados Categorias. Cada categoria deve conter objetos com a mesma semântica. Consideremos, por exemplo, a categoria dos móveis, dos animais, das plantas, etc.
- Para cada parte $\vec{p}_i$, são gerados t vetores de apresentação. Além disso, um objeto da categoria de $\vec{p}_i$ ocorre em apenas um dos t vetores gerados. Como a síndrome é dividida em c partes, são gerados c conjuntos com t vetores em cada um.
- Observe que um conjunto de apresentação pode ser armazenado em uma matriz $M_{t \times x}$, onde t é o número de vetores em cada conjunto e x é a dimensão desses vetores. Essa matriz contém a informação que o usuário relaciona com sua senha.

Os valores t, c, bem como a dimensão dos vetores, x, devem ser definidos no servidor pelo administrador do sistema.

O algoritmo para montagem dos vetores de apresentação tem como dados de entrada: $\vec{p}_w$, y (número de bits de cada parte $\vec{p}_i$ da síndrome), t (número de vetores de cada conjunto), x (dimensão de cada vetor).

Isso significa que:
se $n = |\vec{p}_w|$, então $c = n \div y$, logo
para $j = 1$ até c definimos

t vetores $\vec{z}_1^j, \vec{z}_2^j, ...\vec{z}_j^j, ..., \vec{z}_t^j$; $\vec{z}_i^j \in \{0,1\}^x$, $1 \leq i \leq t$ tal que existe um único vetor $\vec{z}_k^j$ no qual um objeto da categoria de $\vec{p}_j$ ocorre como parte de $\vec{z}_k^j$ e $1 \leq k \leq t$.

Como resultado, temos os vetores de apresentação: $\{\vec{z}_1^j, \vec{z}_2^j, ...\vec{z}_j^j, ..., \vec{z}_t^j\}$, onde $1 \leq j \leq c$;

No nosso exemplo, dado que cada parte da síndrome tem 6 bits, então o espaço vetorial das partes da síndrome é $\{0,1\}^6$. Isso significa que temos $2^6 = 64$ possibilidades de vetores binários. Além disso, no nosso exemplo, cada um desses 64 vetores possui 8 imagens associadas. Usamos 8 imagens diferentes para dificultar a descoberta das categorias de determinado usuário por um espião. Se a cada acesso aparece uma imagem diferente a pessoa que estiver observando as escolhas do usuário tem uma certa dificuldade em entender o que foi escolhido. A Figura 3.8, por exemplo, apresenta as imagens associadas ao vetor binário "000000", que são imagens de animais. Nesse caso, se uma das partes da síndrome estiver associada ao vetor "000000", o servidor seleciona aleatoriamente uma dessas 8 imagens e a coloca em uma das opções do vetor de apresentação. Em seguida, o servidor seleciona aleatoriamente mais 34 imagens diferentes. O usuário deve escolher a opção que possuir a imagem de um animal para indicar que conhece a respectiva parte da síndrome. A escolha do usuário é enviada ao servidor e novo vetor de apresentação é enviado ao usuário para escolha da próxima parte da síndrome.

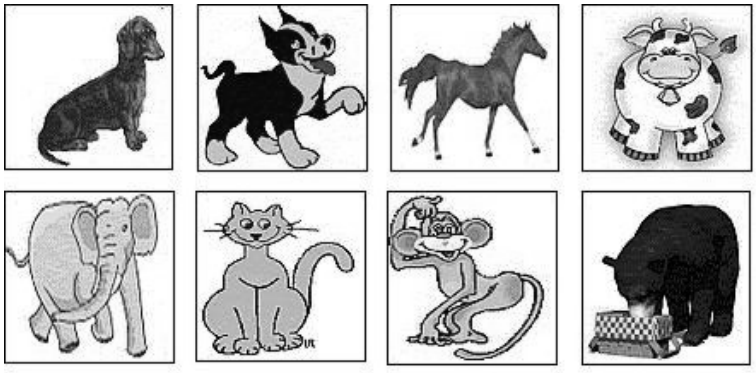

Figura 3.8: Figuras Associadas ao Vetor "000000" são Animais

3.2.6 Funcionamento do Sistema

O funcionamento global do sistema proposto é descrito nesta seção, na forma de passos.

Dado de entrada: Identificação do usuário.

1. O usuário envia sua identificação (nome ou número de cartão ou qualquer outra identificação que foi definida no sistema).

2. O servidor recebe a identificação do usuário.

3. O servidor determina qual é a síndrome associada à senha do usuário. Observe que usamos a notação p_w para indicar a síndrome do usuário, que é representada por $p_w = (p_1, p_2, \cdots, p_c)$.

4. O servidor constrói os c conjuntos de vetores de apresentação $\{\vec{z}_1^j, \vec{z}_2^j, ..., \vec{z}_t^j\}$, $1 \leq j \leq c$. Um conjunto de vetores de apresentação possui vetores cujos elementos são objetos de categorias diferentes selecionadas aleatoriamente. Uma delas é a categoria de uma parte da síndrome relacionada à senha do usuário.

5. O servidor envia ao sistema do usuário, um por vez, todos os c conjuntos de vetores de apresentação $\{\vec{z}_1^j, \vec{z}_2^j, ..., \vec{z}_t^j\}$, $1 \leq j \leq c$.

6. Recebendo um conjunto $\{\vec{z}_1^j, \vec{z}_2^j, ..., \vec{z}_t^j\}$, o sistema do usuário monta a tela de apresentação.

7. O usuário visualiza, um por vez, cada conjunto $\{\vec{z}_1^j, \vec{z}_2^j, ..., \vec{z}_t^j\}$ em seu terminal.

8. Baseando-se em sua senha, o usuário escolhe um dos t vetores do conjunto $(\vec{z}_i^j)$, onde i representa o vetor escolhido, $1 \leq i \leq t$, e j representa o conjunto de vetores enviado, $1 \leq j \leq c$.

9. O sistema do usuário envia os números i e j ao servidor.

10. O servidor recebe os números do vetor, envia outro conjunto de vetores para o usuário, que repete os passos 6 a 10, até que todos os c conjuntos de vetores de apresentação sejam enviados.

11. O servidor aplica uma função nos números recebidos e obtém $f(i, j) = p$ com $1 \leq i \leq t$ e $1 \leq j \leq c$. A função tem como objetivo identificar a parte da síndrome relacionada à senha do usuário nos vetores recebidos selecionados pelo usuário.

12. A partir dos números retornados por $f(i, j)$, o servidor identifica uma classe lateral no arranjo, que armazena p_w.

13. O servidor compara a síndrome da classe lateral obtida no passo anterior com a síndrome do usuário. Se forem iguais, a conexão é autorizada, caso contrário, o servidor finaliza a execução do protocolo e não permite a conexão, nesse caso uma mensagem de erro é enviada ao usuário.

Na aplicação desenvolvida como exemplo, temos $c = 4$, $t = 5$ e $x = 7$, $cat = 8$. Logo, os vetores de apresentação são montados com 35 imagens pertencentes às 64 categorias dos vetores do espaço $\{0, 1\}^6$, que o usuário escolhe no momento da autenticação. Cada imagem foi cadastrada previamente em um banco de dados, e armazenada no terminal do usuário. Cada uma possui como identificação um vetor binário do espaço $\{0, 1\}^6$, uma seqüência numérica e um caminho, composto pelo diretório e o nome do arquivo "Gif". Para cada vetor existem 8 imagens diferentes, porém relacionadas entre si, como mostra a Figura 3.8. As Tabelas 3.4 e 3.5 mostram como as imagens foram cadastradas.

seq	código	nome
0	000000	animal
1	000001	casa
2	000010	comida
3	000011	escola

Tabela 3.4: Tabela de Categorias

No nosso exemplo, as 8 primeiras imagens da Tabela 3.5 são imagens de *animais*, as 8 imagens associadas ao código "000001" são imagens de objetos relacionados a *casa*. O servidor envia o caminho (diretório + nome) dos arquivos de imagens ao usuário para que o sistema monte os vetores de apresentação. Os arquivos de imagens estão gravados no terminal do usuário.

Cada vez que o usuário tentar conectar-se, novos vetores de apresentação são montados. A razão disso deve-se ao fato de que os vetores devem mudar a cada

seq	código	caminho
0	000000	\imagens\A0000
1	000000	\imagens\A0001
2	000000	\imagens\A0002
3	000000	\imagens\A0003
4	000000	\imagens\A0004
5	000000	\imagens\A0005
6	000000	\imagens\A0006
7	000000	\imagens\A0007
8	000001	\imagens\A0008
9	000001	\imagens\A0009
10	000001	\imagens\A0010
11	000001	\imagens\A0011
12	000001	\imagens\A0012
13	000001	\imagens\A0013
14	000001	\imagens\A0014
15	000001	\imagens\A0015

Tabela 3.5: Tabela de Imagens Associadas

sessão para dificultar caso um espião esteja observando e consiga determinar os objetos escolhidos pelo usuário.

No exemplo da seção anterior, as escolhas de Alice são 2, 5, 5, 1, respectivamente. Assim, o servidor recebe os valores $i = 2$ e $j = 1$, $i = 5$ e $j = 2$, $i = 5$ e $j = 3$, e finalmente, $i = 1$ e $j = 4$, onde i é o número do vetor escolhido e j é o número de vetor de apresentação que representamos por c. A função $f(i, j)$ faz:

$$f(1, 2) = (2 \times 7) - 1 = 13$$
$$f(2, 5) = (5 \times 7) - 1 = 34$$
$$f(3, 5) = (5 \times 7) - 1 = 34$$
$$f(4, 1) = (1 \times 7) - 1 = 6$$

Assim, a função busca os valores binários armazenados nas posições 7 a 13, encontra a classe lateral e verifica se é a mesma classe lateral da síndrome do usuário no arranjo. Depois, busca os valores binários armazenados nas posições 27 a 34, encontra a classe lateral e verifica se é a mesma classe lateral da 2.ª parte da síndrome do usuário. A função repete esse procedimento para os outros números de vetores recebidos. Se todos os vetores recebidos tem as mesmas classes laterais da síndrome do usuário, então o acesso é permitido e a autenticação é feita. Mas, se pelo menos

uma das classes laterais diferem da síndrome do usuário, o acesso não é permitido, conseqüentemente, a autenticação não é realizada.

3.3 Resumo

Nesse capítulo foi apresentada a proposta do sistema de apresentação de senhas utilizando teoria dos códigos e a implementação de uma aplicação como exemplo.

Capítulo 4

Análise de Segurança do Sistema Proposto

O objetivo desse capítulo é analisar a segurança do sistema de apresentação de senhas proposto neste trabalho. Na seção 4.1 definimos os tipos de análises da segurança a serem abordadas. Nas seções seguintes faremos as análises, usamos o modelo probabilístico para variáveis aletórias discretas, descrito na seção 4.2. Na seção 4.3, analisamos a probabilidade de alguém descobrir a senha apenas observando as escolhas do usuário. Na seção 4.4, definimos o modelo de oráculos aleatórios e analisamos a segurança do sistema proposto segundo esse modelo.

4.1 Análise de Segurança

Em geral, a análise de segurança de um sistema criptográfico é dividida em várias partes [Goldwasser, 2001]. Faremos as análises descritas a seguir:

1. Análise 1: Probabilidade de obter sucesso em algumas tentativas de acertar a senha, conhecendo apenas a identificação do usuário;

2. Análise 2: Probabilidade de descobrir a senha observando muitas vezes as escolhas de um usuário;

3. Análise 3: Vantagem de um adversário usando o modelo de oráculos aleatórios.

4.2 Probabilidade de Sucesso em Algumas Tentativas de Acesso

Para a análise da probabilidade de sucesso em algumas tentativas de acesso, usamos o modelo probabilístico para variáveis aleatórias discretas que se adaptam bem a uma série de problemas práticos. Um estudo pormenorizado dessas variáveis é importante para a construção de modelos probabilísticos para situações reais e a conseqüente estimação de seus parâmetros [Bussab, 2002].

4.2.1 Distribuição Binomial

Usamos a distribuição binomial para analisar a probabilidade de um espião descobrir a senha de um usuário com escolhas de vetores aleatórias.

Conforme [Bussab, 2002], a distribuição binomial é calculada da seguinte forma:

$$P(X = k) = \binom{n}{k} p^k q^{n-k} \tag{4.1}$$

onde
n é o número de tentativas;
k é o número de sucessos, $k = 0, 1, 2, \cdots, n$;
p é a probabilidade de sucesso;
q é a probabilidade de fracasso.

No sistema proposto, temos t vetores em cada conjuntos de vetores de apresentação. Logo, a chance do espião acertar sua escolha é igual a $\frac{1}{t}$ e, portanto, nesse caso, a chance de errar é igual a $1 - \frac{1}{t}$. Temos c conjuntos de vetores de apresentação. Logo, a chance do espião fazer a escolha certa em todos os conjuntos é igual a $(\frac{1}{t})^c$ e, portanto, $p = (\frac{1}{t})^c$. A chance dele errar é igual a $1 - (\frac{1}{t})^c$. Logo, $q = 1 - (\frac{1}{t})^c$.

No exemplo da aplicação desenvolvida, considerando a distribuição binomial para verificar a probabilidade de obter sucesso com $n = 3$ e $k = 1$, temos: $p = (\frac{1}{5})^4$ e $q = 1 - (\frac{1}{5})^4$. Logo, a chance de um espião acertar a senha em 3 tentativas é dada

pela igualdade:

$$\begin{aligned} P(X=1) &= \binom{3}{1} 0.0016^1 \cdot 0.9984^{(3-1)} \\ P(X=1) &= 4.78 \times 10^{-3} \end{aligned}$$

Agora, considerando a distribuição binomial para verificar a probabilidade de obter sucesso com $n = 3$ e $k = 2$, temos: $p = (\frac{1}{5})^4$ e $q = 1 - (\frac{1}{5})^4$. Logo, a chance de um espião acertar a senha em 3 tentativas é dada pela igualdade:

$$\begin{aligned} P(X=2) &= \binom{3}{2} 0.0016^2 \cdot 0.9984^{(3-2)} \\ P(X=2) &= 7.66 \times 10^{-6} \end{aligned}$$

Considere que o número de tentativas seja $n = 5$ e que o número de sucessos seja $k = 1$. Assim, temos:

$$\begin{aligned} P(X=1) &= \binom{5}{1} 0.0016^1 \cdot 0.9984^{(5-1)} \\ P(X=1) &= 7.94 \times 10^{-3} \end{aligned}$$

Para $n = 5$ e $k = 2$, temos:

$$\begin{aligned} P(X=2) &= \binom{5}{2} 0.0016^2 \cdot 0.9984^{(5-2)} \\ P(X=2) &= 2.54 \times 10^{-5} \end{aligned}$$

Observe que a chance de um espião obter sucesso na descoberta da senha depende do número de tentativas, além do número de vetores em cada conjunto de vetores de apresentação e também do número de conjuntos de vetores.

4.3 Análise da Probabilidade de Descobrir a Senha Observando o Usuário

Considere a seguinte situação: um espião observa várias vezes o usuário acessar o sistema e escolher suas opções nos conjuntos de vetores de apresentação. O objetivo do espião é identificar as escolhas desse usuário e, a partir daí, tentar descobrir a

senha. Na tentativa de descoberta, o espião pode, por exemplo, relacionar os vetores em cada acesso e identificar o que eles têm em comum.

Considere

$$\begin{aligned} \vec{z}_1^j &= (a_1^j, a_2^j, \cdots, a_x^j) \\ \vdots &= \vdots \\ \vec{z}_t^j &= (b_1^j, b_2^j, \cdots, b_x^j) \end{aligned}$$

um conjunto de vetores de apresentação, no qual $j = 1, 2, \cdots, c$. E, além disso, se a senha p_w é dada por $p_w = (p_1, p_2, \cdots, p_c)$, então para $j = 1, 2, \cdots, c$, p_j é igual a um elemento de um dos vetores do conjunto $\vec{z}_1^j, \cdots, \vec{z}_t^j$. Isto é, p_j é elemento de apenas um dos vetores

$$(a_1^j, a_2^j, \cdots, a_x^j)$$

$$\vdots$$

$$(b_1^j, b_2^j, \cdots, b_x^j)$$

Denotamos o vetor escolhido pelo usuário como S_j, ou seja, p_j é igual a uma parte de S_j. Como a senha é dividida em c partes, o usuário deverá escolher c vetores e teremos, portanto, em cada seção de autenticação do usuário, c vetores escolhidos:

$$S_1, S_2, \cdots, S_c$$

Se o usuário fizer m acessos ao servidor para diferentes autenticações, em cada autenticação, o usuário deverá escolher diferentes vetores

$$S_1, S_2, \cdots, S_c$$

Para denotar os diferentes vetores dos diferentes acessos, utilizamos

$$S_1^i, S_2^i, \cdots, S_c^i$$

onde i denota a i-ésima seção de autenticação do usuário.

Portanto, em m acessos para autenticação, o usuário deve escolher um conjunto de vetores da forma:

$$S_1^1, \cdots, S_c^1$$

$$\vdots$$

$$S_1^m, \cdots, S_c^m$$

Para descobrir a senha, o espião pode, por exemplo, fazer as interseções entre os vetores escolhidos em cada acesso, como se segue:

$$S_i^1 \cap \cdots \cap S_i^m \tag{4.2}$$

para $i = 1, \cdots, c$.

Se as escolhas do usuário são corretas, então devemos ter:

$$p_i \text{ é parte do vetor } S_i^1 \cap \cdots \cap S_i^m$$

Observamos que é possível identificar a senha dessa forma, pois como a seleção das componentes dos vetores apresentados é feita aleatoriamente, existe a possibilidade de ter apenas os elementos que fazem parte da senha como resultado da interseção.

Para evitar esse tipo de ataque, podemos atuar no sistema modificando o código e, portanto, as síndromes. A modificação do código implica modificar os valores de n e k, e conseqüentemente modificar as matrizes G e H.

4.4 Vantagem de um Adversário

A análise descrita nessa seção investiga a vantagem de um adversário usando o Modelo de Jogos com Oráculos. Esse modelo é definido a partir dos conceitos de famílias de funções apresentados no Capítulo 2.

4.4.1 Modelo de Jogos com Oráculos

No contexto do modelo de jogos com oráculos, um adversário A é um programa que tem acesso a um oráculo. Este oráculo é tal que, dado como entrada qualquer par (M_0, M_1) de mensagens de mesmo tamanho, o oráculo retorna um texto cifrado,

que é o resultado da cifração de M_0 ou de M_1 [Goldwasser, 2001]. O objetivo do adversário é advinhar qual mensagem foi cifrada.

Nessa análise utilizamos a noção de segurança do tipo ind-cpa [Goldwasser, 2001], no qual o adversário conhece textos originais e tenta descobrir qual deles foi cifrado. No nosso caso, o adversário conhece as categorias e tenta descobrir qual delas autentica o usuário. Para isso, utilizaremos o modelo de jogos com oráculos, com o adversário conhecendo duas senhas, em seguida, o adversário conhece três senhas e, a partir dessas duas análises, definimos um modelo geral.

No sistema de apresentação de senhas, definimos o modelo de jogos com oráculos como se segue:

Modelo com Duas Senhas

Considere um terminal de usuário ligado a um servidor e um adversário observando o terminal. O adversário sabe que a senha do usuário é p_{w_0} ou p_{w_1}, tais que $p_{w_0} = (\delta_1, \delta_2, \cdots, \delta_c)$ e $p_{w_1} = (\gamma_1, \gamma_2, \cdots, \gamma_c)$. O servidor utiliza apenas uma delas para autenticar o usuário.

Nesse contexto, dado o par de senhas (p_{w_0}, p_{w_1}), o servidor já sabe qual é a senha correta para autenticar o usuário, e nesse caso, definimos:

- Mundo 0: o servidor estará no mundo zero, se ele usa a senha p_{w_0} para autenticação e envia ao usuário vetores de apresentação associados à senha p_{w_0}.
- Mundo 1: o servidor estará no mundo um, se ele usa a senha p_{w_1} para autenticação e envia ao usuário vetores de apresentação associados à senha p_{w_1}.

O problema para o adversário é descobrir em qual mundo o oráculo está, isto é, se o servidor utiliza a senha p_{w_0} ou a senha p_{w_1}, e, portanto, descobrir qual é a senha do usuário.

Dado que o adversário conhece os vetores

$$p_{w_0} = (\delta_1, \delta_2, \cdots, \delta_c)$$
$$p_{w_1} = (\gamma_1, \gamma_2, \cdots, \gamma_c)$$

ele também conhece as categorias associadas às partes δ_j e γ_j tal que $1 \leq j \leq c$.

Suponha que as categorias associadas a δ_j e γ_j sejam os conjuntos cat_{δ_j} e cat_{γ_j} respectivamente. Analisamos a seguir em quais situações o adversário consegue identificar o mundo no qual o oráculo está operando.

O adversário está observando as escolhas do usuário, assim, se um objeto do conjunto cat_{δ_j} aparecer no vetor escolhido pelo usuário e nenhum objeto do conjunto cat_{γ_j} aparecer, o adversário identifica que o oráculo está no mundo zero. Da mesma forma, se um objeto do conjunto cat_{γ_j} aparecer no vetor escolhido pelo usuário e nenhum objeto do conjunto cat_{δ_j} aparecer, o adversário identifica que o oráculo está no mundo um.

Isto é, se no conjunto de vetores de apresentação

$$\begin{matrix} z_1^j \\ z_2^j \\ \vdots \\ z_t^j \end{matrix}$$

o usuário escolher um vetor z_i^j, com $1 \leq i \leq t$, no qual aparece apenas um objeto do conjunto cat_{δ_j} e nenhum objeto do conjunto cat_{γ_j}, então o adversário sabe que o oráculo está no mundo zero. Mas, se aparecer apenas um objeto do conjunto cat_{γ_j} e nenhum objeto do conjunto cat_{δ_j}, então o adversário está no mundo um.

Suponha que o vetor escolhido pelo usuário seja:

$$z_i^j = (\theta_1, \cdots, \theta_x)$$

Observando essa escolha do usuário, a chance do adversário identificar o mundo no qual o oráculo está operando é igual a probabilidade de termos apenas um símbolo θ_s, $1 \leq s \leq x$, tal que:

- $\theta_s \in cat_{\delta_j}$ e para todo $r \neq s$, $\theta_r \notin cat_{\gamma_j}$, ou
- $\theta_s \in cat_{\gamma_j}$ e para todo $r \neq s$, $\theta_r \notin cat_{\delta_j}$

Considere K o número total de elementos de todas as categorias. Logo, temos:

Seja θ_s um elemento do vetor z_i^j escolhido pelo usuário. A probabilidade de θ_s pertencer a cat_{δ_j} é igual a

$$\frac{|cat_{\delta_j}|}{K} \tag{4.3}$$

.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, então a probabilidade de *pelo menos um* elemento θ_s, $1 \leq s \leq x$, pertencer a cat_{δ_j} é igual a

$$x\Big(\frac{|cat_{\delta_j}|}{K}\Big) \tag{4.4}$$

A probabilidade de termos no vetor *somente* o elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} é igual a

$$\Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K-|cat_{\delta_j}|)}{K}\Big) \tag{4.5}$$

Observe, na equação acima, que

$$(x-1)\Big(\frac{(K-|cat_{\delta_j}|)}{K}\Big) \tag{4.6}$$

é a probabilidade de *não* ocorrer elementos pertencentes a cat_{δ_j} nas outras posições do vetor z_i^j , isto é, o elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} ocorre em apenas uma posição do vetor z_i^j, e nas outras posições teremos elementos de outras categorias.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, a probabilidade de termos *exatamente* um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} é igual a

$$x\Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K-|cat_{\delta_j}|)}{K}\Big) \tag{4.7}$$

Observe, na equação acima, que

$$x\Big(\frac{|cat_{\delta_j}|}{K}\Big) \tag{4.8}$$

é a probabilidade de pelo menos um elemento θ_s, $1 \leq s \leq x$, pertencer a cat_{δ_j} e

$$(x-1)\Big(\frac{(K-|cat_{\delta_j}|)}{K}\Big) \tag{4.9}$$

é a probabilidade de ocorrer elementos que não pertencem a cat_{δ_j} nas outras posições do vetor.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, então a probabilidade de *não* ocorrer objeto do conjunto cat_{γ_j} em z_i^j é igual a

$$1 - x\Big(\frac{|cat_{\gamma_j}|}{K}\Big) \tag{4.10}$$

Observe que a equação 4.10 é o complementar da equação 4.4.

Seja $P[\cdot]$, a probabilidade do evento "$\cdot$" ocorrer [Goldwasser, 2001]. Assim, $P[\cdot|j]$ é a probabilidade do evento "$\cdot$" ocorrer na parte j da senha. Nesse contexto, chamamos de $P[p_{w_0}|j]$ a probabilidade da parte j da senha p_{w_0} aparecer no vetor de apresentação e $P[\overline{p_{w_0}}|j]$, a probabilidade da parte j da senha p_{w_0} não aparecer no vetor de apresentação. Essa notação é utilizada também para qualquer outra senha.

A probabilidade de *exatamente* um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} ocorrer em $z_i^j = (\theta_1, \cdots, \theta_x)$, e *nenhum* elemento do conjunto cat_{γ_j} ocorrer é dada pelo produto das equações 4.7 e 4.10:

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{|cat_{\delta_j}|(K - |cat_{\delta_j}|)}{K^2}\Big)\Big(1 - x\Big(\frac{|cat_{\gamma_j}|}{K}\Big)\Big) \tag{4.11}$$

Observe que, na equação 4.11, temos probabilidade de, na parte j da senha p_{w_0}, ocorrer *exatamente* um objeto da categoria cat_{δ_j} dada pela equação a seguir:

$$x(x-1)\Big(\frac{|cat_{\delta_j}|(K - |cat_{\delta_j}|)}{K^2}\Big)$$

e a probabilidade de *não* ocorrer objeto da categoria cat_{γ_j} dada pela equação:

$$\Big(1 - x\Big(\frac{|cat_{\gamma_j}|}{K}\Big)\Big)$$

Para simplificar as notações da nossa análise, consideramos

$$cat = |cat_{\delta_j}| = |cat_{\gamma_j}|$$

para $j = 1, 2, \cdots, c$. E, em geral, para qualquer símbolo α da senha, consideramos também que

$$cat = |cat_{\delta_j}|$$

para $j = 1, 2, \cdots, c$. Assim,

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{cat(K - cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big) \tag{4.12}$$

Analogamente, dado $z_i^j = (\theta_1, \cdots, \theta_x)$, então a probabilidade de termos exatamente um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{γ_j} e nenhum elemento do

conjunto cat_{δ_j}, é dada por:

$$P[p_{w_1}|j] = x(x-1)\Big(\frac{|cat_{\gamma_j}|(K-|cat_{\gamma_j}|)}{K^2}\Big)\Big(1-x\Big(\frac{|cat_{\delta_j}|}{K}\Big)\Big) \tag{4.13}$$

Da mesma forma que na equação 4.12, temos a probabilidade de uma categoria de p_{w_1} na parte j da senha ocorrer:

$$P[p_{w_1}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.14}$$

Isto significa que a probabilidade de o adversário dizer que o mundo é zero e acertar ou a probabilidade de o adversário dizer que o mundo é um e acertar, isto é, a probabilidade de identificar o mundo no qual o oráculo está operando, é dada pela equação 4.12 para identificar o mundo zero e pela equação 4.14 para identificar o mundo um, assim, temos:

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{|cat_{\delta_j}|(K-|cat_{\delta_j}|)}{K^2}\Big)\Big(1-x\Big(\frac{|cat_{\gamma_j}|}{K}\Big)\Big) \tag{4.15}$$

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.16}$$

e

$$P[p_{w_1}|j] = x(x-1)\Big(\frac{|cat_{\gamma_j}|(K-|cat_{\gamma_j}|)}{K^2}\Big)\Big(1-x\Big(\frac{|cat_{\delta_j}|}{K}\Big)\Big) \tag{4.17}$$

$$P[p_{w_1}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.18}$$

A probabilidade de o adversário dizer que o mundo é zero e errar ou dizer que o mundo é um e errar, isto é, de não identificar o mundo no qual o oráculo está operando, é dada pelo complemento das equações 4.16 e 4.18, respectivamente:

$$P[\overline{p_{w_0}}|j] = 1 - x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.19}$$

$$P[\overline{p_{w_1}}|j] = 1 - x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.20}$$

No exemplo que estamos utilizando como aplicação, cada categoria possui 8 objetos e, como são 64 categorias, temos um total de 512 objetos. Assim, $|cat_{\delta_j}| = 8$, $|cat_{\gamma_j}| = 8$ e $K = 512$. Dado que no exemplo usamos vetores contendo 7 elementos,

então $z_i^j = (\theta_1, \cdots, \theta_7)$, com $x = 7$. Logo, a probabilidade de θ_s pertencer a cat_{δ_j} é dada pela equação 4.3, onde:

$$\begin{aligned} \frac{|cat_{\delta_j}|}{K} &= \frac{cat}{K} \\ \frac{cat}{K} &= \frac{8}{512} = 0.02 \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de pelo menos um elemento θ_s, $1 \leq s \leq 7$, pertencer a cat_{δ_j} é dada pela equação 4.4, onde:

$$\begin{aligned} x\Big(\frac{|cat_{\delta_j}|}{K}\Big) &= x\Big(\frac{cat}{K}\Big) \\ x\Big(\frac{cat}{K}\Big) &= 7\Big(\frac{8}{512}\Big) = 0.11 \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de termos somente o elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} é dada pela equação 4.5, onde:

$$\begin{aligned} \Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K - |cat_{\delta_j}|)}{K}\Big) &= \Big(\frac{8}{512}\Big)(7-1)\Big(\frac{(512-8)}{512}\Big) \\ \Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K - |cat_{\delta_j}|)}{K}\Big) &= 0.09 \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de termos exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} é calculada pela equação 4.7, como se segue:

$$\begin{aligned} x\Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K - |cat_{\delta_j}|)}{K}\Big) &= 7\Big(\frac{8}{512}\Big)(7-1)\Big(\frac{(512-8)}{512}\Big) \\ x\Big(\frac{|cat_{\delta_j}|}{K}\Big)(x-1)\Big(\frac{(K - |cat_{\delta_j}|)}{K}\Big) &= 0.65 \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de não ocorrer objeto do conjunto cat_{γ_j} em z_i^j é dada pela equação 4.10, onde:

$$\begin{aligned} 1 - x\Big(\frac{|cat_{\gamma_j}|}{K}\Big) &= 1 - 7\Big(\frac{8}{512}\Big) \\ 1 - x\Big(\frac{|cat_{\gamma_j}|}{K}\Big) &= 0.89 \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} ocorrer e nenhum elemento do conjunto cat_{γ_j} ocorrer é dado pela equação 4.12, onde:

$$\begin{aligned} P[p_{w_0}|j] &= x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \\ P[p_{w_0}|j] &= (0.65)(0.89) = 0.58 \end{aligned}$$

A probabilidade de o adversário não identificar o mundo no qual o oráculo está é dada pelas equações 4.19 e 4.20, onde:

$$\begin{aligned} P[\overline{p_{w_0}}|j] &= 1 - 0.58 = 0.42 \\ P[\overline{p_{w_1}}|j] &= 1 - 0.58 = 0.42 \end{aligned}$$

A análise anterior foi feita para uma parte do acesso do usuário. Considere agora, o caso em que o adversário observa a seqüência de acessos do usuário, do primeiro ao último vetor de apresentação.

Tendo visto a primeira parte do acesso, conforme equação 4.12, a probabilidade do adversário identificar na senha p_{w_0} que a parte da senha é p_1 é dada por:

$$P[p_{w_0}|j=1] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.21}$$

Equação 4.12 com $j = 1$. No caso do exemplo usado como aplicação, a probabilidade é igual a 0.58.

Se o adversário observa a primeira e a segunda parte do acesso do usuário, então a probabilidade de ele identificar a senha é dada pelo produto da probabilidade de identificar na senha p_{w_0} a parte 1, conforme equação 4.21, e da probabilidade de identificar na senha a parte 2, como na equação a seguir:

$$P[p_{w_0}|j=2] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big) \tag{4.22}$$

Equação 4.12 com $j = 2$. Como as equações 4.21 e 4.22 são iguais, temos que a probabilidade de identificar a senha p_{w_0}, observando a primeira e a segunda parte do acesso do usuário, é dada por:

$$P[p_{w_0}|j=1,2] = \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1-x\Big(\frac{cat}{K}\Big)\Big)\Big)^2$$

Isto significa que a probabilidade de identificar a senha é o produto das probabilidades de identificação de cada parte da senha. Na aplicação desenvolvida como exemplo, a probabilidade de identificação das partes p_1 e p_2 da senha é calculada a seguir:

$$P[p_{w_0}|j = 1, 2] = (0.58)^2 = 0.33$$

Se o adversário observa a primeira, a segunda e a terceira parte do acesso do usuário, então a probabilidade de ele identificar a senha é dada por:

$$P[p_{w_0}|j = 1, 2, 3] = \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)\Big)^3 \tag{4.23}$$

Na aplicação desenvolvida como exemplo, a probabilidade de identificação das partes p_1, p_2 e p_3 da senha é:

$$[p_{w_0}|j = 1, 2, 3] = (0.58)^3 = 0.19$$

Logo, a probabilidade de o adversário identificar a senha observando todas as partes da senha escolhidas pelo usuário, sabendo que a senha do usuário é dividida em c partes, é dada por:

$$P[p_{w_0}|j = 1, 2, \cdots, c] = \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)\Big)^c \tag{4.24}$$

Analogamente, se a senha do usuário for p_{w_1}, a probabilidade de identificação da senha é dada por:

$$P[p_{w_1}|j = 1, 2, \cdots, c] = \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)\Big)^c \tag{4.25}$$

Na aplicação desenvolvida como exemplo, a probabilidade de identificação de todas as partes da senha é igual a

$$[p_{w_0}|j = 1, 2, 3, 4] = (0.58)^4 = 0.11$$

Partindo desse princípio e usando as equações 4.19 e 4.20, respectivamente, temos que a probabilidade de não identificar a senha, observando todas as partes da senha escolhidas pelo usuário é dada por:

$$P[\overline{p_{w_0}}|j = 1, \cdots, c] = 1 - \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)\Big)^c \tag{4.26}$$

ou

$$P[\overline{p_{w_1}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)\right)^c \qquad (4.27)$$

Na aplicação desenvolvida como exemplo, a probabilidade de não identificação de todas as partes da senha é igual a

$$\begin{aligned} P[\overline{p_{w_0}}|j = 1, \cdots, 4] &= 1 - 0.11 \\ P[\overline{p_{w_0}}|j = 1, \cdots, 4] &= 0.89 \end{aligned}$$

ou

$$P[\overline{p_{w_1}}|j = 1, \cdots, 4] = 0.89$$

O valor obtido nas equações anteriores é muito próximo de 1, isto significa que a probabilidade de não identificação da senha é grande, isto é, o adversário tem uma grande probabilidade de não identificar a senha do usuário.

Modelo com Três Senhas

Considere um terminal de usuário ligado a um servidor e um adversário observando o terminal. O adversário sabe que a senha do usuário é p_{w_0}, p_{w_1} ou p_{w_2}, tais que $p_{w_0} = (\delta_1, \delta_2, \cdots, \delta_c)$, $p_{w_1} = (\gamma_1, \gamma_2, \cdots, \gamma_c)$ e $p_{w_2} = (\sigma_1, \sigma_2, \cdots, \sigma_c)$ e apenas uma delas é utilizada para autenticação.

Nesse contexto, dada a tripla de senhas $(p_{w_0}, p_{w_1}, p_{w_2})$, o servidor já sabe qual é a senha correta para autenticar o usuário e, nesse caso, definimos:

- Mundo 0: o servidor estará no mundo zero, se ele usa a senha p_{w_0} para autenticação e envia ao usuário vetores de apresentação associados à senha p_{w_0}.
- Mundo 1: o servidor estará no mundo um, se ele usa a senha p_{w_1} para autenticação e envia ao usuário vetores de apresentação associados à senha p_{w_1}.
- Mundo 2: o servidor estará no mundo dois, se ele usa a senha p_{w_2} para autenticação e envia ao usuário vetores de apresentação associados à senha p_{w_2}.

O problema para o adversário é descobrir em qual mundo o oráculo está, isto é, se o servidor utiliza a senha p_{w_0}, a senha p_{w_1} ou a senha p_{w_2} e, portanto, descobrir qual é a senha do usuário.

Dado que o adversário conhece os vetores

$$p_{w_0} = (\delta_1, \delta_2, \cdots, \delta_c)$$
$$p_{w_1} = (\gamma_1, \gamma_2, \cdots, \gamma_c)$$
$$p_{w_2} = (\sigma_1, \sigma_2, \cdots, \sigma_c)$$

ele também conhece as categorias associadas às partes δ_j, γ_j e σ_j tal que $1 \leq j \leq c$.

Suponha que as categorias associadas a δ_j, γ_j e σ_j sejam os conjuntos cat_{δ_j}, cat_{γ_j} e cat_{σ_j}, respectivamente. Analisamos a seguir em quais situações o adversário consegue identificar o mundo no qual o oráculo está operando.

O adversário está observando as escolhas do usuário. Assim, se um objeto do conjunto cat_{δ_j} aparecer no vetor escolhido pelo usuário e nenhum objeto dos conjuntos cat_{γ_j} e cat_{σ_j} aparecer, o adversário identifica que o oráculo está no mundo zero. Da mesma forma, se um objeto do conjunto cat_{γ_j} aparecer no vetor escolhido pelo usuário e nenhum objeto dos conjuntos cat_{δ_j} e cat_{σ_j} aparecer, o adversário identifica que o oráculo está no mundo um. Usando o mesmo raciocício, concluímos que o adversário identifica que o oráculo está no mundo dois, se um objeto do conjunto cat_{σ_j} aparecer no vetor escolhido pelo usuário e nenhum objeto dos conjuntos cat_{δ_j} e cat_{γ_j} aparecer.

Isto é, se no conjunto de vetores de apresentação

$$z_1^j$$
$$z_2^j$$
$$\vdots$$
$$z_t^j$$

o usuário escolher um vetor z_i^j, com $1 \leq i \leq t$, no qual aparece apenas um objeto do conjunto cat_{δ_j} e nenhum objeto dos conjuntos cat_{γ_j} e cat_{σ_j}, então o adversário sabe que o oráculo está no mundo zero. Mas, se aparecer apenas um objeto do conjunto cat_{γ_j} e nenhum objeto dos conjuntos cat_{δ_j} e cat_{σ_j}, então o adversário está no mundo um. Analogamente, se aparecer apenas um objeto do conjunto cat_{σ_j} e nenhum objeto dos conjuntos cat_{δ_j} e cat_{γ_j}, então o adversário está no mundo dois.

Suponha que o vetor escolhido pelo usuário seja:

$$z_i^j = (\theta_1, \cdots, \theta_x)$$

Observando essa escolha do usuário, a chance do adversário identificar o mundo no qual o oráculo está operando é igual a probabilidade de termos apenas um objeto θ_s, $1 \leq s \leq x$, tal que:

- $\theta_s \in cat_{\delta_j}$ e para todo $r \neq s$, $\theta_r \notin cat_{\gamma_j}$ e $\theta_r \notin cat_{\sigma_j}$, ou
- $\theta_s \in cat_{\gamma_j}$ e para todo $r \neq s$, $\theta_r \notin cat_{\delta_j}$ e $\theta_r \notin cat_{\sigma_j}$, ou
- $\theta_s \in cat_{\sigma_j}$ e para todo $r \neq s$, $\theta_r \notin cat_{\delta_j}$ e $\theta_r \notin cat_{\gamma_j}$

Considere K o número total de elementos de todas as categorias. Logo, temos:

Seja θ_s um elemento do vetor z_i^j. A probabilidade de θ_s pertencer a cat_{δ_j} é dada pela equação 4.3 e a probabilidade de θ_s pertencer a cat_{γ_j} ou de θ_s pertencer a cat_{σ_j} é, respectivamente:

$$\begin{aligned} \frac{|cat_{\gamma_j}|}{K} &= \frac{cat}{K} \\ \frac{|cat_{\sigma_j}|}{K} &= \frac{cat}{K} \end{aligned}$$

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, então a probabilidade de *pelo menos um* elemento θ_s, $1 \leq s \leq x$, pertencer a cat_{δ_j} é dada pela equação 4.4. A probabilidade de termos *somente* o elemento θ_s pertencente a cat_{δ_j}, $1 \leq s \leq x$, e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} é dada pela equação 4.5.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, então a probabilidade de termos *exatamente* um elemento θ_s pertencente a cat_{δ_j}, $1 \leq s \leq x$, e os outros elementos θ_r, $r \neq s$ não pertencerem a cat_{δ_j} é dada pela equação 4.7.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, a probabilidade de *não* ocorrer objeto do conjunto cat_{γ_j} em z_i^j é dada pela equação 4.10. Como vimos anteriormente,

$$cat = |cat_{\delta_j}| = |cat_{\gamma_j}| = |cat_{\sigma_j}|$$

então a probabilidade de *não* ocorrer objeto do conjunto cat_{σ_j} em z_i^j é também dada pela equação 4.10.

Dado $z_i^j = (\theta_1, \cdots, \theta_x)$, analogamente à equação 4.14, a probabilidade de *exatamente* um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} ocorrer e *nenhum* elemento dos conjuntos cat_{γ_j} e cat_{σ_j} ocorrer é dada por

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big) \tag{4.28}$$

Da equação 4.28, podemos obter:

$$P[p_{w_0}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2 \tag{4.29}$$

Seguindo o mesmo raciocínio, a probabilidade de *exatamente* um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{γ_j} ocorrer e *nenhum* elemento dos conjuntos cat_{δ_j} e cat_{σ_j} ocorrer, ou, a probabilidade de *exatamente* um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{σ_j} ocorrer e *nenhum* elemento dos conjuntos cat_{δ_j} e cat_{γ_j} ocorrer é dada, respectivamente, por:

$$P[p_{w_1}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2 \tag{4.30}$$

$$P[p_{w_2}|j] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2 \tag{4.31}$$

A probabilidade de o adversário não identificar o mundo no qual o oráculo está é o complemento da equação 4.29, isto é,

$$P[\overline{p_{w_0}}|j] = 1 - \Big(x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2\Big) \tag{4.32}$$

As equações acima representam a probabilidade de ser a senha p_{w_0} e o adversário achar que o mundo é um ou dois. Essas equações são análogas no caso em que a senha é p_{w_1} ou p_{w_2}.

Usando os valores dos parâmetros da aplicação desenvolvida como exemplo, temos:

Dado que no exemplo usamos vetores contendo 7 elementos, então $z_i^j = (\theta_1, \cdots, \theta_7)$, com $x = 7$. Logo, a probabilidade de θ_s pertencer a cat_{δ_j} dada pela equação 4.3 é igual a 0.02. Além disso, a probabilidade de pelo menos um elemento θ_s, $1 \leq s \leq 7$, pertencer a cat_{δ_j} dada pela equação 4.4 é igual a 0.11.

Dado $z_i^j = (\theta_1, \cdots, \theta_7)$, então a probabilidade de termos somente o elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$ não pertencerem a cat_{δ_j} dado pela equação 4.5 é igual a 0.09.

A probabilidade de termos no vetor exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} e os outros elementos θ_r, $r \neq s$, não pertencerem a cat_{δ_j} calculada pela equação 4.7 é igual a 0.65

A probabilidade de não ocorrer objeto do conjunto cat_{γ_j} e não ocorrer objeto do conjunto cat_{σ_j} em z_i^j dada pela equação 4.10 é igual a 0.89. Visto que as categorias usadas na aplicação têm o mesmo número de objetos, os valores das probabilidades da equação 4.10 são iguais.

Assim, a probabilidade de exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} ocorrer e nenhum elemento dos conjuntos cat_{γ_j} e cat_{σ_j} ocorrer dada pela equação 4.29 é igual a 0.51.

Portanto, a probabilidade de o adversário não identificar o mundo no qual o oráculo está é dada pela equação 4.32, onde:

$$\begin{aligned} P[\overline{p_{w_0}}|j] &= 1 - 0.51 = 0.49 \\ P[\overline{p_{w_1}}|j] &= 1 - 0.51 = 0.49 \\ P[\overline{p_{w_2}}|j] &= 1 - 0.51 = 0.49 \end{aligned}$$

A análise anterior foi feita para uma parte do acesso do usuário. Considere agora, o caso em que o adversário observa a seqüência de acessos do usuário, do primeiro ao último vetor de apresentação.

Tendo visto a primeira parte do acesso, conforme equação 4.29, a probabilidade do adversário identificar que a parte da senha do usuário é p_1 é dada por:

$$P[p_{w_0}|j=1] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2 \tag{4.33}$$

Equação 4.29 com $j = 1$. No caso do exemplo usado como aplicação, a probabilidade é igual a 0.51.

Se o adversário observa a primeira e a segunda parte do acesso do usuário, então a probabilidade de ele identificar a senha é dada pelo produto da probabilidade de identificar na senha p_{w_0} a parte 1, conforme equação 4.33, e da probabilidade de identificar na senha a parte 2, como na equação a seguir:

$$P[p_{w_0}|j=2] = x(x-1)\Big(\frac{cat(K-cat)}{K^2}\Big)\Big(1 - x\Big(\frac{cat}{K}\Big)\Big)^2 \tag{4.34}$$

Equação 4.29 com $j = 2$. Como as equações 4.33 e 4.34 são iguais, temos que a probabilidade de identificar a senha p_{w_0}, observando a primeira e a segunda parte do acesso do usuário, é dada por:

$$P[p_{w_0}|j = 1, 2] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^2 \tag{4.35}$$

Isto significa que a probabilidade de identificar a senha é o produto das probabilidades de identificação de cada parte da senha. Na aplicação desenvolvida como exemplo, a probabilidade de identificação das partes p_1 e p_2 da senha é:

$$P[p_{w_0}|j = 1, 2] \quad = \quad (0.51)^2 = 0.26$$

Se o adversário observa a primeira, a segunda e a terceira parte do acesso do usuário, então a probabilidade de ele identificar a senha é dada por:

$$P[p_{w_0}|j = 1, 2, 3] \quad = \quad \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^3$$

Na aplicação desenvolvida como exemplo, a probabilidade de identificação das partes p_1, p_2 e p_3 da senha é:

$$P[p_{w_0}|j = 1, 2, 3] = (0.51)^3 = 0.13$$

Logo, a probabilidade de o adversário identificar a senha observando todas as partes da senha escolhidas pelo usuário, sabendo que a senha do usuário é dividida em c partes, é dada por:

$$P[p_{w_0}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \tag{4.36}$$

Analogamente, se a senha do usuário for p_{w_1} ou p_{w_2} probabilidade de identificação da senha é dada, respectivamente, por:

$$P[p_{w_1}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \tag{4.37}$$

$$P[p_{w_2}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \tag{4.38}$$

Na aplicação desenvolvida como exemplo, a probabilidade de identificação de todas as partes da senha é

$$\begin{aligned} P[p_{w_0}|j = 1, 2, 3, 4] &= (0.51)^4 \\ P[p_{w_0}|j = 1, 2, 3, 4] &= 0.07 \end{aligned}$$

Partindo desse princípio e usando a equação 4.19, temos que a probabilidade de não identificar a senha, observando todas as partes da senha escolhidas pelo usuário é dada por:

$$P[\overline{p_{w_0}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \quad (4.39)$$

ou

$$P[\overline{p_{w_1}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \quad (4.40)$$

ou

$$P[\overline{p_{w_2}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^c \quad (4.41)$$

Na nossa aplicação, a probabilidade de não identificação de todas as partes da senha é igual a

$$\begin{aligned} P[\overline{p_{w_0}}|j = 1, \cdots, 4] &= 1 - 0.07 \\ P[\overline{p_{w_0}}|j = 1, \cdots, 4] &= 0.93 \end{aligned}$$

Em qualquer um dos casos listados anteriormente a probabilidade é a mesma, isto é, a probabilidade de não identificação de todas as partes da senha é igual a 0.93 para p_{w_1} e p_{w_2}.

O valor obtido na equação anterior é muito próximo de 1, isto significa que a probabilidade de não identificação da senha é grande, isto é, o adversário tem uma grande probabilidade de não identificar a senha do usuário.

Em um segundo acesso ao terminal, do primeiro ao último vetor de apresentação, a probabilidade de exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} ocorrer e nenhum elemento dos conjuntos cat_{γ_j} e cat_{σ_j} ocorrer é dada pelo produto do resultado da equação 4.36 no primeiro acesso e pelo resultado da equação 4.36 no segundo acesso, isto é, é a probabilidade de exatamente um elemento θ_s ocorrer no primeiro acesso completo e a probabilidade de exatamente um elemento θ_s ocorrer no segundo acesso completo, que é o produto das probabilidades dos dois acessos, conforme mostramos a seguir:

$$P[p_{w_0}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{2c} \quad (4.42)$$

Analogamente para as senhas p_{w_1} e p_{w_2} respectivamente, temos:

$$P[p_{w_1}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{2c} \tag{4.43}$$

$$P[p_{w_2}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{2c} \tag{4.44}$$

Na aplicação desenvolvida como exemplo, o valor da probabilidade para cada uma das senhas é:

$$\begin{aligned} P[p_{w_0}|j=1,\cdots,4] &= (0.51)^{2(4)} \\ P[p_{w_0}|j=1,\cdots,4] &= 0.005 \end{aligned}$$

Em um terceiro acesso ao terminal, a probabilidade de termos exatamente um elemento θ_s, $1 \leq s \leq 7$, pertencente a cat_{δ_j} ocorrer e nenhum elemento dos conjuntos cat_{γ_j} e cat_{σ_j} ocorrer é dado por:

$$P[p_{w_0}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{3c} \tag{4.45}$$

Analogamente para as senhas p_{w_1} e p_{w_2} respectivamente, temos:

$$P[p_{w_1}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{3c} \tag{4.46}$$

$$P[p_{w_2}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{3c} \tag{4.47}$$

Na aplicação desenvolvida como exemplo, o valor da probabilidade para cada uma das senhas é:

$$\begin{aligned} P[p_{w_0}|j=1,\cdots,4] &= (0.51)^{3(4)} \\ P[p_{w_0}|j=1,\cdots,4] &= 3.28 \times 10^{-4} \end{aligned}$$

Logo, em l acessos completos, a probabilidade de termos exatamente um elemento θ_s, $1 \leq s \leq x$, pertencente a cat_{δ_j} e nenhum elemento dos conjuntos cat_{γ_j} e cat_{σ_j} é dada por:

$$P[p_{w_0}|j=1,\cdots,c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1-x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.48}$$

Analogamente para as senhas p_{w_1} e p_{w_2} respectivamente, temos:

$$P[p_{w_1}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.49}$$

$$P[p_{w_2}|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.50}$$

Portanto, a probabilidade de o adversário não identificar o mundo no qual o oráculo está observando l acessos é igual a

$$P[\overline{p_{w_0}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.51}$$

$$P[\overline{p_{w_1}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.52}$$

$$P[\overline{p_{w_2}}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^2\right)^{lc} \tag{4.53}$$

Observe que as equações são as mesmas para p_{w_0}, p_{w_1} e p_{w_2} e, como temos a mesma quantidade de elementos em cada categoria, o resultado dessas equações também será o mesmo.

Na aplicação desenvolvida como exemplo, as probabilidades acima são:

$$\begin{aligned} P[\overline{p_{w_0}}|j = 1, 2, 3, 4] &= 1 - 3.28 \times 10^{-4} \\ P[\overline{p_{w_0}}|j = 1, 2, 3, 4] &= 0.99 \end{aligned}$$

$$\begin{aligned} P[\overline{p_{w_0}}|j = 1, 2, 3, 4] &= 1 - 3.28 \times 10^{-4} \\ P[\overline{p_{w_0}}|j = 1, 2, 3, 4] &= 0.99 \end{aligned}$$

$$\begin{aligned} P[\overline{p_{w_1}}|j = 1, 2, 3, 4] &= 1 - 3.28 \times 10^{-4} \\ P[\overline{p_{w_1}}|j = 1, 2, 3, 4] &= 0.99 \end{aligned}$$

$$\begin{aligned} P[\overline{p_{w_1}}|j = 1, 2, 3, 4] &= 1 - 3.28 \times 10^{-4} \\ P[\overline{p_{w_1}}|j = 1, 2, 3, 4] &= 0.99 \end{aligned}$$

$$P[\overline{p_{w_2}}|j = 1, 2, 3, 4] = 1 - 3.28 \times 10^{-4}$$

$$P[\overline{p_{w_2}}|j = 1, 2, 3, 4] = 0.99$$

$$P[\overline{p_{w_2}}|j = 1, 2, 3, 4] = 1 - 3.28 \times 10^{-4}$$
$$P[\overline{p_{w_2}}|j = 1, 2, 3, 4] = 0.99$$

O valor resultante é praticamente 1, portanto, a probabilidade de identificar o mundo no qual o oráculo está operando, dado que o adversário conhece três senhas, é muito pequena, isto é, é bem próxima de zero.

Modelo Geral

A probabilidade de o adversário identificar o mundo no qual o oráculo está, isto é, a probabilidade de identificação da senha de um usuário, em termos gerais, é dada por:

$$P[p_w|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^y\right)^{lc} \tag{4.54}$$

onde,
y é o número de senhas que o adversário conhece;
l é o número de acessos que o adversário observa;
c é o número de partes na qual a senha é dividida, isto é, o número de categorias relacionadas à senha do usuário.

A probabilidade de o adversário não identificar o mundo no qual o oráculo está é o complemento da equação 4.56, isto é,

$$P[\overline{p_w}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^y\right)^{lc} \tag{4.55}$$

Vimos que, no ataque do adversário descrito anteriormente, quanto mais senhas usadas pelo servidor o adversário conhece, mais difícil fica a identificação da senha de determinado usuário apenas observando o acesso completo. Na análise de duas senhas, o adversário tem aproximadamente a probabilidade de 0.10 de identificar a senha e a chance de não identificar é maior, cerca de 0.89. Na análise de três senhas a chance de identificação diminui em relação a de duas senhas, passa a ser 0.06, enquanto que a chance de não identificação aumenta para 0.93.

A probabilidade de 0.06 nos indica que, de cada 20 vítimas, 1 é atacada com sucesso, isso para um sistema de segurança é muito ruim. Esse valor deve ser bem menor para que o sistema seja confiável.

A partir desses valores, observamos que, em comparação com sistemas de criptografia utilizados, o sistema tem um baixo nível de segurança, mas são valores que nos garantem alguma segurança, por menor que seja. Esse sistema não tem a pretensão de ser um sistema criptográfico perfeito, é um esquema de segurança que pode ser modificado para usar outros códigos da Teoria de Códigos, que evitem o excesso de processamento, já que o Código Linear é de tempo exponencial, e garantam maior segurança.

4.5 Ataque de Força Bruta

Em um esquema de decodificação por força bruta é necessário comparar a senha do usuário com todas as 2^k palavras-código e selecionar a mais próxima. Isso é fácil para códigos pequenos. Mas, se k é grande, isso é praticamente impossível [Costello, 1983]. No caso do sistema desenvolvido, $k = 32$, o que resulta em $\{0,1\}^{32}$ que é um código razoavelmente grande, e selecionar a palavra-código mais próxima demoraria um certo tempo de processamento.

Considere a situação da seção anterior com o adversário sabendo que as senhas utilizadas pelo servidor é uma das seguintes senhas:

$$p_{w_0}, p_{w_1}, \cdots, p_{w_{2^k-1}}$$

Nesse caso, a probabilidade de o adversário identificar a senha do usuário em l acessos é dada pela equação:

$$P[p_w|j = 1, \cdots, c] = \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^{2^k-1}\right)^{lc} \tag{4.56}$$

A probabilidade de não identificar a senha do usuário, nesse contexto, é dada pela equação:

$$P[\overline{p_w}|j = 1, \cdots, c] = 1 - \left(x(x-1)\left(\frac{cat(K-cat)}{K^2}\right)\left(1 - x\left(\frac{cat}{K}\right)\right)^{2^k-1}\right)^{lc} \tag{4.57}$$

4.6 Ataque do Dicionário

Esse sistema evita o ataque do dicionário ao módulo parcialmente seguro do servidor, pois bits são armazenados e não caracteres criptografados ou não. Nesse caso, se o atacante tentar converter esses bits para caracteres, eles podem não fazer sentido e confundir quem tentar descobrir o que está por trás disso.

Mesmo que o espião converta os caracteres das palavras do dicionário e fizer comparação, ele não vai conseguir descobrir a síndrome, pois para isso seria necessário que o mesmo possuísse a matriz de verificação de paridade H para obter a síndrome das palavras consultadas. A matriz de verificação de paridade, como vimos anteriormente, fica armazenada no módulo seguro do servidor. Alem disso, uma nova codificação para os caracteres das senhas foi criada, não usamos código ASCII, nem Unicode.

4.7 Vantagens do Sistema Proposto

1. O sistema de apresentação de senhas proposto não necessita de teclado numérico nem vitual para escolha da senha, mesmo porque a senha não é usada em momento algum.

2. Se uma pessoa mal intencionada estiver observando as escolhas do usuário, vai verificar que o mesmo escolhe uma entre t opções, e que nessa opção existem x imagens diferentes. Esse procedimento é realizado c vezes, pois a síndrome é dividida em c partes. A cada acesso a figura utilizada para representar a categoria muda, o que dificulta a descoberta pelo espião, pois são x possibilidades em cada linha do vetor de apresentação. Na aplicação desenvolvida como exemplo, o usuário deve escolher uma entre 5 opções e, em cada opção, temos 7 imagens diferentes. O procedimento é realizado 4 vezes, pois a síndrome é dividida em 4 partes.

3. Para saber o que significa cada figura é necessário ter acesso aos arquivos do servidor, que é um servidor seguro, e mesmo que esse acesso ocorra, o espião encontra bits ligados aos nomes da figura, que por sua vez são recebem nomes como “A253.gif”. Nesse caso, o espião somente obtém sucesso se conseguir

acessar no servidor a tabela de identificação dos usuários, descobrir a síndrome do mesmo, e comparar os bits com o arquivos de categoria e subcategoria.

4. Para qualquer pessoa, leiga ou não, é mais fácil associar figuras do que números ou letras.

5. Os parâmetros do sistema podem ser alterados, para obter uma segurança maior, pois quanto maior o espaço e o sub-espaço vetorial, mais demorado e complicado fica para quem tentar descobrir qual o valor correto a ser utilizado.

6. Usar figuras que representam uma categoria é mais intuitivo, pois uma categoria possui 8 figuras diferentes que a representam, portanto qualquer uma das 8 figuras que aparecerem na tela é válida.

4.8 Desvantagens do Sistema Proposto

1. O sistema de apresentação de senhas desenvolvido necessita de um sistema computacional robusto, caso contrário ficará com uma execução lenta.

2. O esforço computacional para montar a matriz P é grande, principalmente para parâmetros k e n grandes, o que demanda muito tempo e carga de processamento, apesar do processamento ser *off-line*.

3. As figuras a serem utilizadas devem ser nítidas, uma imagem perfeita, com resolução alta.

4. Cada subcategoria deve possuir figuras totalmente diferentes, apenas relacionadas com a categoria, para que o usuário não confunda.

5. O sistema não atende a usuários deficientes visuais.

4.9 Resumo

Nesse capítulo fizemos a análise da segurança do sistema que foi dividida em três tipos: probabilidade de obter sucesso em algumas tentativas de acertar a senha, conhecendo apenas a identificação do usuário; probabilidade de descobrir a senha

observando muitas vezes as escolhas de um usuário; vantagem de um adversário usando o modelo de jogos com oráculos.

Capítulo 5

Conclusão

Neste capítulo apresentamos um resumo das contribuições dessa dissertação e enumeramos sugestões de trabalhos futuros.

5.1 Contribuições

Nesta dissertação foi proposto um sistema de armazenamento e apresentação de senhas que visa aperfeiçoar os sistemas existentes de modo a aumentar a segurança e garantir a confidencialidade da senha. Nesse sistema a senha não é informada explicitamente, o usuário apenas prova que a conhece.

Propomos um modelo em três níveis: Servidor Seguro, Servidor Parcialmente Seguro e Cliente. O servidor seguro possui requisitos de segurança e armazena a senha do usuário. O servidor parcialmente seguro usa conceitos da teoria de códigos para armazenar as senhas. Como vimos no capítulo 4, quanto maior o valor dos parâmetros usados no sistema, isto é, quanto maior o código linear e o arranjo, mais segurança o sistema possui. Além disso, à medida que os parâmetros do sistema mudam, o código também muda e isto pode ser feito a qualquer momento. Com a possibilidade de modificação do código, a segurança é incrementada. Afinal de contas, modificando o código, a síndrome também é modificada. E isso é feito sem a necessidade de alterar a senha previamente cadastrada. Observamos que a alteração no código é transparente ao usuário, este somente necessita ser informado se houver alteração da síndrome.

O cliente, que denominamos Máquina Hostil, é um equipamento que não usa

requisitos de segurança, nele colocamos apenas as categorias que são usadas no momento da montagem dos vetores de apresentação.

A Teoria de Códigos é, principalemente, usada para detecção e correção de erros em transmissão de dados, mas nesse sistema usamos essa teoria em outra abordagem: para codificar a senha. Podemos verificar pelas análises feitas no capítulo 4, que o sistema é seguro do ponto de vista dos tipos de ataques analisados nesse trabalho.

Na primeira análise feita, verificamos que a probabilidade de um espião identificar a senha do usuário em 3 e 5 tentativas de acesso, conhecendo apenas a identificação do usuário é, respectivamente, 0.0047848552288 e 0.007948922. Mas, como a probabilidade aumenta a cada tentativa, então para manter o sistema seguro, devemos usar o sistema de bloqueio de acesso se houver mais de três tentativas frustradas.

Na segunda análise feita, usamos a interseção entre as várias opções escolhidas pelo usuário. A segurança, no entanto, vai depender do resultado da função aleatória usada para escolha das categorias a serem mostradas ao usuário. Vimos que, nesse caso, o sistema é menos seguro, pois o espião pode inferir a senha com mais facilidade se a interseção resultar em conjuntos de categorias pequenos. Quanto menor o conjunto resultante da interseção, mais fácil inferir as categorias que o usuário utiliza. O ideal, nesse caso, é fazer alteração do código freqüentemente, assim o usuário terá novas categorias a selecionar e dificultará na interseção. Mas, em contrapartida, é ruim para o usuário ter de memorizar categorias em pouco tempo.

Na terceira análise, definimos uma fórmula geral para a segurança do sistema. Verificamos que quanto maior for o número de categorias usadas no sistema, é mais difícil para adversário descobrir a senha do usuário.

No sistema proposto a segurança depende da definição dos parâmetros usados. Quanto maior o código linear, mais seguro é o sistema. Isso significa que, para que a nossa proposta seja realmente segura, os valores n e k do código linear devem ser grandes, maiores que os valores usados na aplicação desenvolvida como exemplo, inclusive com um maior número de objetos por categoria. Porém, sabemos que um código linear grande, com muitas categorias, pode dificultar o uso para usuários com menor nível de escolaridade ou com dificuldade de uso da tecnologia. Além disso, um código muito grande exige uma carga de processamento alta. E talvez isso não seja viável para sistemas comerciais. Mas, esses problemas podem servir de motivação

para novos estudos.

5.2 Sugestões para Trabalhos Futuros

Enumeramos a seguir sugestões ou tópicos a serem estudados em trabalhos futuros.

- Um sistema que permita ao usuário definir o arranjo, sem que ele entenda de teoria dos códigos.
- Um estudo da satisfação do usuário em relação ao sistema proposto.
- Um sistema que utilize outros tipos de códigos da Teoria de Códigos.
- Utilização de outros sistemas de Chave Pública que substituam a Teoria de Códigos.
- Implementação de um sistema que vise acessibilidade, permitindo sua utilização por usuários deficientes visuais.

5.3 Resumo

Nesse capítulo foram enumeradas as contribuições desta dissertação e sugestções de trabalhos que podem ser desenvolvidos, em frentes diversificadas:

- linha teórica;
- projeto de protocolos ou modelos novos;
- implementação

Referências Bibliográficas

[Abramo, 2002] HEFEZ, A., VILLELA, M.L.T. *"Códigos corretores de erros"*, Rio de Janeiro: IMPA (2002).

[Anton, 2001] ANTON, H., RORRES, C. *"Álgebra linear com aplicações"*, Porto Alegre: Bookman, 8 ed. (2001).

[Barroso, 2007] BARROSO, V. *"Fundamentos de telecomunicações: codificação de Canal"*, *https : //dspace.ist.utl.pt/bitstream/*2295/51256/1/$FT32 - codifica\%C3\%A7\%C3\%A30Canal.pdf$, acessado em 13 de setembro de 2007, às 9h24m.

[Bussab, 2002] BUSSAB, W. O. *"Estatística Básica"*, São Paulo: Saraiva, 5 ed. (2002).

[Cert, 2007] CERT. *"Cartilha de Segurança para Internet"*, *http : //cartilha.cert.br/conceitos/sec*2.*html*$\sharp$*sec*2, acessado em 20 de setembro de 2007, às 15h.

[Costello, 1983] COSTELLO JR., DANIEL J., LIN, S. *"Error Control Coding-Fundamentals and Applications"*, Prentice Hall (1983).

[Domingues, 1982] DOMINGUES, H. H., IEZZI, G. *"Álgebra moderna"*, 2 ed., São Paulo: Atual (1982).

[Edwards, 2000] JR. EDWARDS, C.H., PENNEY, D.E. *"Introdução à álgebra linear"*, Rio de Janeiro: LTC (2000).

[Febraban, 2007] FEBRABAN. *"Segurança de dados"*, *http : //www.febraban.org.br*, acessado em 20 de setembro de 2007, às 16h.

[Fernandez, 2005] FERNANDEZ, E.M.G. *"Códigos corretores de erros"* (2005).

[Filho, 1998] FILHO, J.G.S. *"Informação, codificação e segurança de dados"*, Brasília: ENE (1998).

[Goldwasser, 2001] GOLDWASSER, S., BELLARE, M. *"Lecture Notes on Cryptography"*, Lecture Notes, pp 14, Agosto (2001).

[Hamming, 1950] HAMMING, R.W. *"Error detecting and error correcting codes"*, Bell Syst. Tech. J., 29, pp 147-160, Abril (1950).

[Hefez, 1994] HEFEZ, A. *"Introdução à teoria dos códigos"*, UNICAMP (1994).

[Lay, 1999] LAY, C.D. *"Álgebra Linear e suas aplicações"*, 2 ed., Rio de Janeiro: LTC (1999).

[Lint, 1992] LINT, J.H.V. *"Introduction to Coding Theory"*, Third Edition, Springer (1992).

[Lipschutz, 1994] LIPSCHUTZ, S. *"Álgebra linear: teoria e problemas"*, 3 ed., São Paulo: Makron Books (1994).

[Koblitz, 1994] KOBLITZ, N. *"A course in number theory and cryptography"*, 2 ed., New York, NY, USA: Springer-Verlag (1994).

[Morgan, 2000] MORGAN, M. *"Java 2 para programadores profissionais"*, Rio de Janeiro: Ciência Moderna (2000).

[Pass,2006] PASS, R., CHONG,S. *"Introduction to Cryptography"*, *http* : $//www.cs.cornell.edu/courses/cs687/2006fa/lectures/lecture12.pdf$, acessado em 28 de janeiro de 2008, às 22h.

[Paula Filho, 2003] PAULA FILHO, W.P. *"Engenharia de Software: fundamentos, métodos e padrões"*, Rio de Janeiro: LTC (2003).

[Rossi, 2004] SANTOS, R.R. *"Programando em Java 2 - teoria e aplicações"*, Rio de Janeiro: Axcel Books (2004).

[Schneier, 2004] SCHNEIER, B. *"Customers, Passwords, and Websites"*, IEEE Security & Privacy (2004).

[Shannon, 1948] SHANNON, C.E. *"A Mathematical Theory of Communication"*, Bell Syst. Tech. J. vol. 27, pp. 379-423, Outubro (1948).

[Silberschatz, 2004] SILBERSCHATZ, A. et al. *"Sistemas Operacionais com Java"*, Rio de Janeiro: Elsevier (2004).

[Steins, 1987] STEINBRUCH, A. *"Álgebra linear"*, 2 ed., São Paulo: McGraw-Hill (1987).

[Stallings, 1998] STALLINGS, W. *"Cryptography and network security: principles and practice"*, 2 ed., New Jersey: Prentice-Hall (1998).

[Tanenbaum, 2007] A.S. *"Sistemas Operacionais Modernos"*, 2 ed., São Paulo: Prentice Hall (2003).

[Tanenbaum, 2003] A.S. *"Redes de computadores"*, 4 ed., Rio de Janeiro: Elsevier (2003).

Apêndice A

Código-fonte da Aplicação

A.1 Servidor Parcialmente Seguro

```
// A Classe Aleatório gera posições aleatórias para
//os vetores de apresentação

package sis;
 import java.util.List;
 import java.lang.*;
 import javax.swing.*;
 import java.awt.*;


public class aleatorio extends JFrame{

 String vt2 = "";
 String vt3 = "";
 String vt4 = "";
 String vt5 = "";
 String vt6 = "";
 GridBagLayout gbl;
 GridBagConstraints Regras;
 public JButton B1;
 public JButton B2;
```

```
public JButton B3;
public JButton B4;
public JButton B5;
public JButton B6;

 public String aleatorio (String Sen)
   {
     String vt = new String();
     String[] vet = new String[Constantes.alfab];
     Integer[] vet1 = new Integer[Constantes.alfab-1];
     String[] vetM = new String[Constantes.alfab-1];
     String[] vet2 = new String[Constantes.compsenha+1];
     String[] vet3 = new String[Constantes.compsenha+1];
     String[] vet4 = new String[Constantes.compsenha+1];
     String[] vet5 = new String[Constantes.compsenha+1];
     String[] vet6 = new String[Constantes.compsenha+1];
     String[] vetS = new String[Constantes.compsenha];
     int[] vetPos= new int[Constantes.compsenha];

     vet[0]="A";
     vet[1]="B";
     vet[2]="C";
     vet[3]="D";
     vet[4]="E";
     vet[5]="F";
     vet[6]="G";
     vet[7]="H";
     vet[8]="I";
     vet[9]="J";
     vet[10]="K";
     vet[11]="L";
     vet[12]="M";
     vet[13]="N";
     vet[14]="O";
```

```
vet[15]="P";
vet[16]="Q";
vet[17]="R";
vet[18]="S";
vet[19]="T";
vet[20]="U";
vet[21]="V";
vet[22]="W";
vet[23]="X";
vet[24]="Y";
vet[25]="Z";
for (int c=0;c<4;c++) {
  gera_numeros nro = new gera_numeros();
  List numeros = nro.geranumeros(0,24);
  for (int i = 0; i < 25; i++)
      vet1[i] = (Integer) numeros.get(i);
  for (int x=0;x<25;x++)
      vetM[x]="";
  int i=0;
  int j=0;
  vetM[vet1[i].intValue()]=Sen.toUpperCase();
  vetPos[c]=vet1[i].intValue();
  while (i<25) {
    if (vet[j]!=Sen.toUpperCase()){
       vetM[vet1[i].intValue()]=vet[j];
       i++; }
    j++; }
  for ( j=0;j<5;j++)
      vet2[j]= vetM[j];
  for (j=5; j<10;j++)
      vet3[j-5]= vetM[j];
  for (j =10; j<15;j++)
      vet4[j-10]= vetM[j];
  for (j =15; j<20;j++)
```

```
            vet5[j-15]= vetM[j];
        for (j =20; j<25;j++)
            vet6[j-20]= vetM[j];
        vt2 = "";
        vt3 = "";
        vt4 = "";
        vt5 = "";
        vt6 = "";
        for (j =0; j<5;j++){
            vt = vt +vet2[j]; }
        vt = vt +" ";
        for (j =0; j<5;j++){
            vt = vt+ vet3[j];  }
        vt = vt +" ";
        for (j =0; j<5;j++){
            vt = vt+ vet4[j];   }
        vt = vt +" ";
        for (j =0; j<5;j++){
            vt = vt+ vet5[j];}
        vt = vt +" ";
        for (j =0; j<5;j++){
            vt = vt+ vet6[j];   }
        return vt;
      }
    return vt;
    }
}


// Classe que consulta o número de seqüência das categorias do
// usuário no banco de dados

package sis;
 import java.sql.Connection;
```

```
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;
 import java.sql.Statement;
 import javax.swing.JOptionPane;

public class consseqcat {

  static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
  static final String DB_URL = "jdbc:mysql://localhost/sas";
  static final String USERN= "root";
  static final String PASS= "drilica";

   public int consseqcat (String senha)
   {
     int sequ=0;
     String text1="SELECT seq FROM categ where codigo = ";
     // Cria a conexão com o banco de dados
     Connection connection = null;
     Statement statement = null;
     try
       {
        Class.forName( JDBC_DRIVER);
        connection = DriverManager.getConnection(DB_URL,USERN,PASS);
        statement =connection.createStatement();
        ResultSet rs1 = statement.executeQuery(text1+"'"+senha+"'");
        ResultSetMetaData rsmd1 = rs1.getMetaData();
        if (!rs1.next())
         {
          JOptionPane.showMessageDialog(null,"Categoria não existe!");
         }
        else { sequ =rs1.getInt("seq");}
          statement.close();
```

```
        }  // fim do try
        catch (SQLException sqlException)
          {
           String st = "Erro!"+"\nMensagem: "+sqlException.getMessage();
           JOptionPane.showMessageDialog(null,st);
           sqlException.printStackTrace();
           System.exit( 1 );
          } // fim do catch
        catch (ClassNotFoundException classNotFound)
         {
          classNotFound.printStackTrace();
          System.exit( 1 );
         } // fim do catch
        finally // instrução e conexão são fechadas adequadamente
         {
          try
           {
            statement.close();
            connection.close();
           } // fim do try
          catch ( Exception exception )
           {
             exception.printStackTrace();
             System.exit( 1 );
           } // fim do catch
         } // fim do finally
  return sequ;
   } // fim do método consulta
}

// Classe de constantes utilizadas em todo o sistema.

package sis;
```

```
public final class Constantes {

  static final int k=32;
  static final int n=56;
  static final int m=n-k;
  static final int alfab = 26;
// comprimento da mensagem
  static final int c=6;
  static final int compusuario = 40;
  static final int compsenha = 4;
// categorias
  static final int ncat = 64;
// numero de posições por linha no vetor de apresentacao
  static final int x = 7;
// subcategorias
  static final int cat = 8;
// Numero de figuras;
  static final int nfig = 512;
}

// Classe que consulta a identificação e síndrome
// do usuário no banco de dados

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionListener;
 import java.awt.event.ActionEvent;
 import javax.swing.*;
 import java.sql.Connection;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;
```

```
public class consulta {
    // driver JDBC e URL de banco de dados
       static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
       static final String DB_URL = "jdbc:mysql://localhost/sas";
       static final String USR= "root";
       static final String PASS= "drilica";

   public String consulta (String usuario)
   {
      String senha = "";
      String txt1 ="SELECT * FROM usupw where id = ";
      String txt2 ="Digite o usuário novamente";
      Connection connection = null;
      Statement statement = null;
      ResultSet rs = null;
      try
        {
          Class.forName( JDBC_DRIVER);
          connection = DriverManager.getConnection(DB_URL,USR,PASS);
          statement =connection.createStatement();
          rs = statement.executeQuery(txt1+"'"+usuario+"'");
          ResultSetMetaData rsmd = rs.getMetaData();
          while (!rs.next())
           {
            JOptionPane.showMessageDialog(null,"Usuário não existe");
            senha = "";
            usuario = JOptionPane.showInputDialog(txt2);
            rs = statement.executeQuery(txt1+"'"+usuario+"'");
           }
           senha = rs.getString("sind");
           statement.close();
          }  // fim do try
           catch (SQLException sqlException)
```

```
            {
              String st = "Erro!"+"\n"+sqlException.getMessage();
              JOptionPane.showMessageDialog(null,st);
              sqlException.printStackTrace();
              System.exit( 1 );
            } // fim do catch
          catch (ClassNotFoundException classNotFound)
            {
              classNotFound.printStackTrace();
              System.exit( 1 );
            } // fim do catch
          finally //instrução e conexão são fechadas adequadamente
            {
             try
              {
                statement.close();
                connection.close();
              } // fim do try
             catch ( Exception exception )
               {
                 exception.printStackTrace();
                 System.exit( 1 );
               } // fim do catch
            } // fim do finally
   return senha;
  } // fim do método consulta
} // fim da classe consulta


// Classe que faz a conversão da senha para o código binário
// O Código usado não é ASCII.

package sis;
 import javax.swing.JOptionPane;
```

```
public class Converte {

    public int[] troca(String senha) {

    nome = JOptionPane.showInputDialog(null,nome);
    if (nome==null)
    {
     nome = "Nome não informado!";
     JOptionPane.showMessageDialog(null,nome,"Erro",2);
     System.exit(0);
    }
    int compi = Constantes.c+1;
    if (nome.length()>compi)
    {
     nome="Nome maior que c caracteres";
     JOptionPane.showMessageDialog(null,nome,"Erro",0);
     System.exit(0);
    }
    String vet[]= new String[Constantes.alfab];
    String vetm[] =  new String[Constantes.alfab];
    String nsenha[]= new String[Constantes.compsenha];
    String tsenha[]= new String[Constantes.compsenha];
    int nsen[] = new int[Constantes.n];
    vet[0] ="A";
    vet[1] ="B";
    vet[2] ="C";
    vet[3] ="D";
    vet[4] ="E";
    vet[5] ="F";
    vet[6] ="G";
    vet[7] ="H";
    vet[8] ="I";
    vet[9] ="J";
```

```
vet[10]="K";
vet[11]="L";
vet[12]="M";
vet[13]="N";
vet[14]="O";
vet[15]="P";
vet[16]="Q";
vet[17]="R";
vet[18]="S";
vet[19]="T";
vet[20]="U";
vet[21]="V";
vet[22]="W";
vet[23]="X";
vet[24]="Y";
vet[25]="Z";
vetm[0] ="01000001";
vetm[1] ="01000010";
vetm[2] ="01000011";
vetm[3] ="01000100";
vetm[4] ="01000101";
vetm[5] ="01000110";
vetm[6] ="01000111";
vetm[7] ="01001000";
vetm[8] ="01001001";
vetm[9] ="01001010";
vetm[10]="01001011";
vetm[11]="01001100";
vetm[12]="01001101";
vetm[13]="01001110";
vetm[14]="01001111";
vetm[15]="01010000";
vetm[16]="01010001";
vetm[17]="01010010";
```

```
    vetm[18]="01010011";
    vetm[19]="01010100";
    vetm[20]="01010101";
    vetm[21]="01010110";
    vetm[22]="01010111";
    vetm[23]="01011000";
    vetm[24]="01011001";
    vetm[25]="01011010";
    int tam=0;
    for(int cont=0;cont<(Constantes.compsenha);cont++)
     {
       tam = tam+1;
       for(int conta2 = 0;conta2<26;conta2++)
       {
          if (senha.substring(cont,tam).equals(vet[conta2]))
             { nsenha[cont]= vetm[conta2];
               break;
             }
       }
     }
    int con = 0,j;
    for (int cont=0;cont<Constantes.compsenha;cont++)
      {
        for(int i =0;i<8;i++)
          {
            j=i+1;
            nsen[con]=Integer.parseInt(nsenha[cont].substring(i,j));
            con++;
          }
      }
    return nsen;
 }
}
```

```
// Classe que monta os botões da tela principal do sistema

package sis;
 import java.awt.event.ActionListener;
 import java.util.ArrayList;
 import java.util.List;
 import javax.swing.JButton;

public class Criadora {

  private static JButton btn;
  private static List<String> listaRotulos = new ArrayList<String>();
  private static List<JButton> listaBotoes = new ArrayList<JButton>();
  private static int contador;

  public static JButton criaBotão(String Rotulo, ActionListener liste,
      String ActionCommand) {

      btn = new JButton(Rotulo);
      btn.setActionCommand(ActionCommand);
      btn.addActionListener(liste);
      listaBotoes.add(btn);
      listaRotulos.add(btn.getText());
      return btn;
   }

  public static String  mudaRotulo(String ActionCommand) {
       for (JButton b : listaBotoes) {
         if (b.getActionCommand() == ActionCommand) {
            if(b.getText() == listaRotulos.get(contador)){
                 b.setText(listaRotulos.get(contador + 1));
                 return b.getActionCommand();
             }
```

```
            else{
                  b.setText(listaRotulos.get(contador));
                  contador += 1;
                  return b.getActionCommand();
                }
          }
      }
    return null;
  }
}

// Classe que exclui usuários do sistema

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionListener;
 import java.awt.event.ActionEvent;
 import javax.swing.*;
 import java.sql.Connection;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;

public class exclui_user {
    // driver JDBC e URL de banco de dados
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/sas";
   static final String USERN= "root";
   static final String PASS= "drilica";

  public exclui_user()
   {
```

```
String usuario = "";
String senha = "";
String tx1="Digite o nome do Usuário";
String txt2 ="Erro: Você não digitou o Nome do usuário.";
String txt3 ="DELETE FROM usupw where id = ";
String txt4 ="Tem certeza de que deseja remover o usuário - ";
int flag1 = 1;
int flag2 = 1;
int flag3 = 1;
int ret;
while (flag1 == 1)
 {
   if (flag2 == 1)
     {usuario = JOptionPane.showInputDialog(txt1);
      usuario = usuario.toUpperCase();}
   else
     {flag2 =0;}
     if (usuario != null){
        if (usuario.length() == 0)
           {JOptionPane.showMessageDialog(null,txt2);}
        if (flag2 == 0)
           {flag1 = 0;}
   Connection connection = null;
   Statement statement = null;
   try
     {
      Class.forName( JDBC_DRIVER);
      connection = DriverManager.getConnection(DB_URL,USERN,PASS);
      statement =connection.createStatement();
      statement.executeUpdate(txt3+"'"+usuario+"'");
      JOptionPane.showMessageDialog(null,txt4+usuario);
      statement.close();
      flag1 = 0;
     }  // fim do try
```

```
        catch (SQLException sqlException)
          {
           String st = "Erro!"+"\n"+sqlException.getMessage();
           JOptionPane.showMessageDialog(null,st);
           sqlException.printStackTrace();
           System.exit( 1 );
          } // fim do catch
        catch (ClassNotFoundException classNotFound)
          {
            classNotFound.printStackTrace();
            System.exit( 1 );
          } // fim do catch
        finally //instrução e conexão são fechadas adequadamente
         {
           try
             {
              statement.close();
              connection.close();
             } // fim do try
           catch ( Exception exception )
             {
               exception.printStackTrace();
               System.exit( 1 );
             } // fim do catch
           } // fim do finally

         }
         else
           {flag1 = 0;}
      } // fim do while
    } // fim do método pega_user
} // fim da classe pega_user
```

```
// Classe que cria a janela principal do aplicativo

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionEvent;
 import java.awt.event.ActionListener;
 import java.io.IOException;
 import java.io.PrintStream;
 import java.net.Socket;
 import java.io.BufferedReader;
 import java.io.InputStreamReader;
 import javax.swing.*;
 import java.io.ObjectInputStream;
 import java.io.ObjectOutputStream;
 import java.awt.*;


public class frmPrincipalBotoes extends JFrame implements
ActionListener {

   public String clic = "";
   private int contadorDeCliques;
   private static final long serialVersionUID = 1L;

   public frmPrincipalBotoes(String Mens)

      setTitle("Escolha a sua opção:");
      String B1 = Mens.substring(0,4);
      String B2 = Mens.substring(6,10);
      String B3 = Mens.substring(12,16);
      String B4 = Mens.substring(18,22);
      String B5 = Mens.substring(24,28);
      setLayout(new FlowLayout(FlowLayout.CENTER));
      add(Criadora.criaBotão(B1, this, "1"));
```

```
    add(Criadora.criaBotão(B2, this, "2"));
    add(Criadora.criaBotão(B3, this, "3"));
    add(Criadora.criaBotão(B4, this, "4"));
    add(Criadora.criaBotão(B5, this, "5"));
    pack();
    setLocationRelativeTo(null);
}

public void actionPerformed(ActionEvent e) {
    if(contadorDeCliques < 1){
        clic = clic + Criadora.mudaRotulo(e.getActionCommand());
        contadorDeCliques +=1;
      }
 }

 public static void main(String[] args) {

    // Declaro o socket cliente
    Socket s = null;
    // Declaro a Stream de saida de dados
    PrintStream ps = null;
    // PARA LER A MENSAGEM DO SERVIDOR
    // Declaro o leitor para a entrada de dados
     BufferedReader entrada=null;
     InputStreamReader isr = null;
    // FIM DA DECLARAÇÃO
    String txt ="Digite sua Identificação: ";
    String ID = JOptionPane.showInputDialog(null,txt);
     try{
      //Cria socket com recurso desejado na porta especificada
       s = new Socket("localhost",7000);
      //Cria a Stream de saida de dados
       ps = new PrintStream(s.getOutputStream());
       isr = new InputStreamReader(s.getInputStream());
```

```
        //Enviando uma mensagem ao servidor
        //Imprime uma linha para a stream de saída de dados
         ps.println(ID);
        // Para pegar 4 mensagens do SAS
         for(int se=0; se<4;se++)
           {
            //RECEBENDO A MENSAGEM DO SERVIDOR
            entrada = new BufferedReader(isr);
            //Aguarda algum dado e imprime a linha recebida
            String ent = entrada.readLine();
            //Colocando na tela
            frmPrincipalBotoes janela = new frmPrincipalBotoes(ent);
            janela.setDefaultCloseOperation(3);
            janela.setVisible(true);
            String p = janela.clicados;
            while (p == "")
              {
                p = janela.clicados;
              }
            janela.dispose();
            ps.println(p);
           //enviando a opcao para o servidor
         }
        //Trata possíveis exceções
       }catch(IOException e){
         System.out.println("Erro ao criar/enviar dados pelo socket");
     }finally{
         try{
            //Encerra o socket cliente
             s.close();
        }catch(IOException e){}
     }
   }
}
```

```
// Classe usada para misturar as posições do vetor randomicamente

package sis;
 import java.util.*;

 public class gera_numeros {

    public List geranumeros ( int start, int end){

      List lista = new ArrayList(end-start+1);
      for (int i=start ; i <= end; i++){ // adiciona todos os números
         lista.add(new Integer(i));
      }
      // mistura os números aletatoriamente
      Collections.shuffle(lista);
      return lista;
  }
}


//Classe que constrói a janela onde os dados serão apresentados

package sis;
 import java.awt.event.*;
 import java.awt.*;
 import javax.swing.*;

public class Janela extends JFrame {

   public Janela(String titulo, Dimension tamanho)
    {
      setTitle(titulo);
```

```
      setSize(tamanho);
      centralize();
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      getContentPane().setLayout(null);
      getContentPane().setBackground(new Color(200,230,200));
      Image Ico;
      Ico = Toolkit.getDefaultToolkit().getImage("teste.jpg");
      setIconImage(Ico);
    }

    public void centralize()
    {
      Dimension T = Toolkit.getDefaultToolkit().getScreenSize();
      Dimension J = getSize();
      if (J.height > T.height) setSize(J.width,T.height);
      if (J.width > T.width) setSize(T.width,J.height);
      setLocation((T.width - J.width)/2,(T.height - J.height)/2);
    }

    public static void main(String[] args)
    {
      Janela jan = new Janela("Janela",new Dimension(600,500));
      jan.show();
    }
}

package sis;
 import java.awt.FlowLayout;
 import javax.swing.JFrame;
 import javax.swing.JLabel;
 import javax.swing.SwingConstants;
 import javax.swing.Icon;
 import javax.swing.ImageIcon;
```

```
public class LabelFrame extends JFrame {
   private JLabel label2;
   private JLabel label3;

   public LabelFrame()
   {
     super( "SAS - Sistema de Apresentação de Senhas" );
     setLayout( new FlowLayout() );
     Icon bug = new ImageIcon( getClass().getResource("fundo2.gif"));
     label2 = new JLabel( "", bug, SwingConstants.CENTER );
     add( label2 );
     label3 = new JLabel();
     label3.setText( "Label with icon and text at bottom" );
     label3.setIcon( bug );
     label3.setHorizontalTextPosition( SwingConstants.CENTER );
     label3.setVerticalTextPosition( SwingConstants.BOTTOM );
     label3.setToolTipText("SAS - Sistema de Apresentação de Senhas");
     add( label3 );
   }
}

package sis;
 import javax.swing.JFrame;

public class LabelTest {
   public void LabelTest1()
   {
      LabelFrame labelFrame = new LabelFrame();
      labelFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      labelFrame.setSize( 1280, 770 );
      labelFrame.setVisible( true );
   }
}
```

```
//Classe que gera a matriz G

package sis;

public class Matriz_G {

  public int[][] Matrizg() {
     int[][] ident = new int[Constantes.k][Constantes.k];
     int[][] pind = new int[Constantes.k][Constantes.n];
     int[][] p = new int[Constantes.k][Constantes.m];
// BUSCANDO A MATRIZ IDENTIDADE
     Matriz_ident ft = new Matriz_ident();
     int[][] id = ft.Matrizid(Constantes.k,Constantes.k);
     for (int l = 0; l < Constantes.k; l++) {
       for (int c = 0; c < (Constantes.k); c++) {
          ident[l][c] = id[l][c];
       }
     }
// Buscando a matriz p
     Matriz_P fk = new Matriz_P();
     p = fk.Matrizp();
// gerando o pind = concatenação da p + id
     for(int l=0;l<(Constantes.k);l++)
      {
       for(int c=0;c < Constantes.m;c++)
        {
          pind[l][c]=p[l][c];
        }
      }
     for(int l=0;l<(Constantes.k);l++)
      {
       for(int c=Constantes.m;c< Constantes.n;c++)
         {
```

```
          pind[l][c] = ident[l][c - Constantes.m];
         }
       }
     return pind;
    }
}


// Classe que gera a matriz H

package sis;

public class Matriz_H {

  public int[][] Matrizh() {

    int[][] ident = new int[Constantes.m][Constantes.m];
    int[][] pind = new int[Constantes.m][Constantes.n];
    // BUSCANDO A MATRIZ IDENTIDADE
    Matriz_ident ft = new Matriz_ident();
    int[][] id = ft.Matrizid(Constantes.m,Constantes.m);
    for (int l = 0; l < Constantes.m; l++) {
       for (int c = 0; c < (Constantes.m); c++) {
          ident[l][c] = id[l][c];
       }
    }
    // Buscando a matriz p
    Matriz_P fk = new Matriz_P();
    int[][] p = fk.Matrizp();
    // Buscando a transposta da matriz P
    Transposta tr = new Transposta();
    int[][] k = tr.Transp(p, Constantes.k, Constantes.m);
    // gerando o pind = concatenação da p + id
    for(int l=0;l<(Constantes.m);l++)
```

```
     {
        for(int c=0;c < Constantes.m;c++)
         {
            pind[l][c] = ident[l][c];
         }
     }
    for(int l=0;l<(Constantes.m);l++)
     {
       for(int c=Constantes.m; c < Constantes.n;c++)
         {
           pind[l][c]=k[l][c-Constantes.m];
         }
     }
    return pind;
  }
}


// Classe que cria a matriz identidade kxk

 package sis;

public class Matriz_ident {
   public int[][] Matrizid(int ml, int mc) {
      int[][] ident = new int[ml][mc];
        for(int i=0;i<ml;i++)
        {
            for(int j=0;j<mc;j++)
            {
                if (i==j) ident[i][j]=1;
                else ident[i][j]=0;
            }
        }
      return ident;
```

```
    }
}


// Classe que gera a matriz P, criada a partir da matriz
// identidade, que juntas formarão a matriz geradora G

package sis;

import javax.swing.JOptionPane;

public class Matriz_P {

    public int[][] Matrizp() {
       int[][] p;
       p = new int[Constantes.k][Constantes.m];
       int[][] ident;
       int[][] p;
       ident = new int[Constantes.k][Constantes.k];
       p = new int[Constantes.k][Constantes.m];
       long vetorint[] = new long[Constantes.m];
       int nro = 1;
       vetorint[0]=nro;
       for (int i=1;i<Constantes.m;i++)
        {
           // Deslocamento de 1 à esquerda
           vetorint[i]=vetorint[i-1]<<1;
        }
        String bin1[] = new String[Constantes.m];
        for (int l=0;l<Constantes.m;l++)
            bin1[l]="";
        String bin2;
       //converte inteiro em binário e completa com zeros a esquerda
        for (int j = 0;j<Constantes.m ;j++)
```

```
{
   bin2= Long.toBinaryString(vetorint[j]);
   int tam = bin2.length();
   int dif = Constantes.n - tam;
   if (dif > 0)
     for (int k = 0;k<dif;k++)
      {
         bin2="0"+bin2;
      }
   bin1[j]=bin2;
   int bin1[] = new int[Constantes.m];
   String bin2[] = new String[Constantes.m];
   for (int l = 1; l<(Constantes.n);l++)
     {
       for (int c = 0; c<(Constantes.m);c++)
        {
           bin2=Integer.toBinaryString(c);
           int tam = bin2.length();
           int dif = Constantes.m - tam;
           if (dif > 0)
              for (int k = 0;k<dif;k++)
                {
                   bin2="0"+bin2;
                }
        }
     }
   for (int c = 1; c<(Constantes.k);c++)
     {
       for (int l = 0; l<(Constantes.k);l++)
        {
          p[l][0] = p[l][0]+ident[l][c];
          if (p[l][0]%2 == 0) p[l][0]=0;
          if (p[l][0]%2 == 1) p[l][0]=1;
        }
```

```
  }
for (int l = 0; l<(Constantes.k);l++)
  {
    p[l][1] = ident[l][4]+ident[l][5]+ident[l][6]+
              ident[l][7]+ident[l][10];
    if (p[l][1]%2 == 0) p[l][1]=0;
    if (p[l][1]%2 == 1) p[l][1]=1;
  }
for (int l = 0; l<(Constantes.k);l++)
  {
     p[l][2] = ident[l][2]+ident[l][3]+ident[l][6]+
               ident[l][7]+ident[l][10];
     if (p[l][2]%2 == 0) p[l][2]=0;
     if (p[l][2]%2 == 1) p[l][2]=1;
   }
for (int l = 0; l<(Constantes.k);l++)
  {
    p[l][3] = ident[l][1]+ident[l][3]+ident[l][5]+
              ident[l][7]+ident[l][9];
    if (p[l][3]%2 == 0) p[l][3]=0;
    if (p[l][3]%2 == 1) p[l][3]=1;
  }
for(int c=0;c<(Constantes.m);c++)
 {
    for(int l=0;l<Constantes.k;l++)
      {
         int t = c+2;
         if (t+c-1 >= Constantes.k) r =0;
         else r = t+c-1;
         for(int j=0;j<Constantes.k;j++)
         {
             if (j!=r)
                 {
                 p[l][c]=p[l][c]+ident[j][l];
```

```
                            System.out.print(p[l][c]);
                            }
                        }
                    }
                 }
                }
       return p;
    }
}


// Classe que faz a multiplicação entre matrizes

package sis;

public class Mult_mat {
    public int[][] Mult (int [][] hh, int lh, int ch,
        int [][] g, int lg, int cg ){

        int[][] status = new int[lg][ch];
        Transposta tr = new Transposta();
        int[][] k = tr.Transp(hh, lh, ch);
        int res = 0;
        int p = 0;
        for (int l=0; l<lg; l++)
        {
            for (int c = 0; c < lh; c ++)
            {
                for (int i=0; i<cg; i++)
                {
                    p = g[l][i]*k[i][c];
                    res = res+p;
                    if (res>1)
                        res = 0;
```

```
                }
                status[l][c] = res;
            }
        }
        return status;
    }
}


// Classe que faz a multiplicação de vetor por matriz

package sis;

public class Mult_vet {

    public int[] Multvet(int[] senha) {

// senha= 56 bits
// H = 24,56
//  sindrome = mod(senha*H',2); 1x 24
//  a matriz P tem 24 colunas e 32 linhas, fazendo P transposta
//  ela fica com 24 linhas e 32 colunas, aí é só completar as outras
//  24 colunas com a matriz identidade.

    int[] sindrome = new int[Constantes.n];
    Matriz_H hh = new Matriz_H();
    int [][] h= hh.Matrizh();
    int temp;
    for (int l=0;l<Constantes.m;l++)
    {
        temp = 0;
        for (int c=0;c<Constantes.n;c++)
        {
        temp = senha[c]*h[l][c]+temp;
```

```
        }
        if (temp%2 == 0) temp=0;
        if (temp%2 == 1) temp=1;
        sindrome[l]= temp;
    }
    return sindrome;
  }
}

// Classe usada para inserir novo usuário no sistema

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionListener;
 import java.awt.event.ActionEvent;
 import javax.swing.*;
 import java.sql.Connection;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;

 public class novo_user {
   // driver JDBC e URL de banco de dados
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/sas";
    static final String USR= "root";
    static final String PASS= "drilica";

  public novo_user()
   {
    String usu = "";
    String senha = "";
```

```
String txt = "";
String txt1 = "";
int flag1 = 1;
int flag2 = 1;
int flag3 = 1;
/* Gerando a sindrome do usuário:
 senha = 4 digitos
 converter cada digito para ASC e posteriormente para binario
 e concatena-los, tendo assim uma palavra de tamanho k
 Gerando o codigo linear da senha
 com a palavra de tamanho k (binário) * G
 Gerando a Sindrome:
 (Gera os lideres + codigo linear da senha) * H */
int ret;
while (flag1 == 1)
{
  if (flag2 == 1){
      txt = "Digite o nome do Usuário";
      usu = JOptionPane.showInputDialog(txt);
      usu = usuario.toUpperCase();}
  else
     {flag2 =0;}
  if (usu != null){
     if (flag3 == 1) {
       txt = "Digite nova senha: ";
       senha = JOptionPane.showInputDialog(txt);
       senha = senha.toUpperCase();}
     if (usuario.length() == 0){
        txt = "Erro: Você não digitou o Nome do usuário."
        JOptionPane.showMessageDialog(null,txt);}
     if (senha != null){
        if (senha.length() < Constantes.compsenha) {
           txt = "Erro: A senha deve ter ";
           int comp = Constantes.compsenha;
```

```
            txt1 = " dígitos";
            JOptionPane.showMessageDialog(null,txt+comp+txt1);
             }
         else
            {flag3 = 0;}
      if (flag2 == 0 & flag3 == 0)
         {flag1 = 0;}
   // gera a senha binária
    Converte sen = new Converte();
    int[] pass = sen.troca(senha);
  // multiplica a senha pela matriz g e gera a palavra-codigo
    Mult_vet v1 = new Mult_vet();
    int[] v = v1.Multvet(pass);
  // Calcula a sindrome
    VetStr sind = new VetStr();
    String sindu = sind.VetStr(v);
    Connection connection = null;
    Statement statement = null;
    ResultSet res = null;
    try
      {
        Class.forName( JDBC_DRIVER);
        connection = DriverManager.getConnection(DB_URL,USR,PASS);
        statement =connection.createStatement();
        txt = "insert into usupw(id,sind) values(";
        statement.executeUpdate(txt+"'"+usu+"',"+"'"+sindu+"')");
        int tam=Constantes.m/Constantes.c;
        int ini=0,fim=Constantes.c;
        String vet[] = new String[tam];
        for (int i =0; i<tam;i++)
         {
          vet[i]= sindu.substring(ini,fim);
          ini = fim;
          fim = fim+Constantes.c;
```

```
     }
    String cat = "";
    for (int i =0; i<tam;i++)
      {
       txt = "select nome from categ where codigo = ";
       res = statement.executeQuery(txt+"'"+vet[i]+"'");
       if (!res.next()) {
         JOptionPane.showMessageDialog(null,"Tabela vazia");
       }
       else
         {cat=cat+res.getString(1)+ " - "; }
     }
    statement.close();
    txt = "\nUsuário cadastrado!\n";
    txt1 = "\nPartes da senha: \n";
    JOptionPane.showMessageDialog(null,+txt1+"\n "+cat);
    flag1 = 0;
  }  // fim do try
catch (SQLException sqlException)
{
  String st ="Erro!"+"\nMensagem: "+sqlException.getMessage();
  JOptionPane.showMessageDialog(null,st);
  sqlException.printStackTrace();
  System.exit( 1 );
} // fim do catch
catch (ClassNotFoundException classNotFound)
{
   classNotFound.printStackTrace();
   System.exit( 1 );
} // fim do catch
finally //a instrução e conexão são fechadas adequadamente
  {
    try
      {
```

```
                statement.close();
                connection.close();
              } // fim do try
            catch ( Exception exception )
               {
                exception.printStackTrace();
                System.exit( 1 );
               } // fim do catch
            } // fim do finally
          }
        else
           {flag1 = 0;}
     }
     else
       {flag1 = 0;}
   } // fim do while
 } // fim do método novo_user
} // fim da classe novo_user


// Classe que busca os dados do usuário no Banco de Dados

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionListener;
 import java.awt.event.ActionEvent;
 import javax.swing.*;
 import java.sql.Connection;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;
```

```
public class pega_user {
  // driver JDBC e URL de banco de dados
     static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
     static final String DB_URL = "jdbc:mysql://localhost/sas";
     static final String USR= "root";
     static final String PASS= "drilica";

 public pega_user()
 {
    String usu = "";
    String senha = "";
    int flag1 = 1;
    int flag2 = 1;
    int flag3 = 1;
    int ret;
    String txt = "";
    String tx1 = "";
    while (flag1 == 1)
    {
      if (flag2 == 1) {
         txt = "Digite o nome do Usuário";
         usu = JOptionPane.showInputDialog(txt);
         usu = usuario.toUpperCase();}
      else
        {flag2 =0;}
      if (usu != null){
         if (flag3 == 1){
            txt = "Digite a nova senha: ";
            senha = JOptionPane.showInputDialog(txt);
            senha = senha.toUpperCase();}
         if (usuario.length() == 0){
            txt = "Erro: Você não digitou o Nome do usuário."
            JOptionPane.showMessageDialog(null,txt);}
         if (senha != null){
```

```
    if (senha.length() < Constantes.compsenha) {
        txt = "Erro: A senha deve ter ";
        int comp = Constantes.compsenha;
        txt1 = " dígitos.";
        JOptionPane.showMessageDialog(null,txt+comp+txt1);}
    else
        {flag3 = 0;}
if (flag2 == 0 & flag3 == 0)
          {flag1 = 0;}
Converte ss = new Converte();
int[] senha2 = ss.troca(senha);
VetStr sind = new VetStr();
Mult_vet v1 = new Mult_vet();
int[] v = v1.Multvet(senha2);
String sindu2 = sind.VetStr(v);
Connection connection = null;
Statement statement = null;
ResultSet res = null;
try
  {
   Class.forName( JDBC_DRIVER);
   connection=DriverManager.getConnection(DB_URL,USR,PASS);
   statement=connection.createStatement();
   txt = "SELECT * FROM usupw where id = ";
   ResultSet rs = statement.executeQuery(txt+"'"+usu+"'");
   txt = "UPDATE usupw SET sind = ";
   txt1 =" where id = ";
   statement.executeUpdate(txt+"'"+sindu2+"'"+
                           txt1+"'"+usu+"'");
   int tam=Constantes.m/Constantes.c;
   int ini=0,fim=Constantes.c;
   String vet[] = new String[tam];
   for (int i =0; i<tam;i++)
    {
```

```
   vet[i]= sindu2.substring(ini,fim);
   ini = fim;
   fim = fim+Constantes.c;
  }
 String cat = "";
 for (int i =0; i<tam;i++)
  {
   txt = "select nome from categ where codigo = ";
   res = statement.executeQuery(txt+"'"+vet[i]+"'");
    if (!res.next()) {
      JOptionPane.showMessageDialog(null,"Tabela vazia");
    }
    else {
      cat=cat+res.getString(1)+ " - "; }
   }
  flag1 = 0;
  statement.close();
  txt = "\nSenha alterada!\n";
  txt1 = "\nPartes da nova senha: \n";
  JOptionPane.showMessageDialog(null,txt+txt1+"\n "+cat);
 }  // fim do try
catch (SQLException sqlException)
  {
   String st = "Erro!"+"\n "+sqlException.getMessage();
   JOptionPane.showMessageDialog(null,st);
   sqlException.printStackTrace();
   System.exit( 1 );
  } // fim do catch
catch (ClassNotFoundException classNotFound)
 {
   classNotFound.printStackTrace();
   System.exit( 1 );
 } // fim do catch
finally //instrução e conexão são fechadas adequadamente
```

```
              {
                try
                 {
                  statement.close();
                  connection.close();
                 } // fim do try
                catch ( Exception exception )
                 {
                   exception.printStackTrace();
                   System.exit( 1 );
                 } // fim do catch
               } // fim do finally
             }
            else
             {flag1 = 0;}
        } else
           {flag1 = 0;}
       } // fim do while
 } // fim do método pega_user
} // fim da classe pega_user


// Classe que busca os dados da categoria do usuário no Banco de
// Dados

package sis;
 import java.awt.FlowLayout;
 import java.awt.event.ActionListener;
 import java.awt.event.ActionEvent;
 import javax.swing.*;
 import java.sql.Connection;
 import java.sql.Statement;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
```

```
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;

public class pegacateg {
  // driver JDBC e URL de banco de dados
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/sas";
   static final String USR= "root";
   static final String PASS= "drilica";

 public pegacateg()
 {
   String usu = "";
   String senha = "";
   String txt = "";
   String txt1 = "";
   int flag1 = 1;
   int flag2 = 1;
   int flag3 = 1;
   int ret;
   while (flag1 == 1) {
     if (flag2 == 1){
       txt = "Digite o nome do Usuário";
       usu = JOptionPane.showInputDialog(txt);
       usu = usuario.toUpperCase();}
     else
       {flag2 =0;}
     if (usu != null){
        if (flag3 == 1) {
            txt = "Digite a nova senha: ";
            senha = JOptionPane.showInputDialog(txt);
            senha = senha.toUpperCase();}
     if (usuario.length() == 0) {
        txt = "Erro: Você não digitou o Nome do usuário.";
```

```
        JOptionPane.showMessageDialog(null,txt);}
    if (senha != null){
        if (senha.length() < Constantes.compsenha){
           txt = "Erro: A senha deve ter ";
           txt1 = " dígitos.";
           int comp = Constantes.compsenha;
           JOptionPane.showMessageDialog(null,txt+comp+txt1);}
        else
           {flag3 = 0;}
        if (flag2 == 0 & flag3 == 0)
             {flag1 = 0;}
        Converte ss = new Converte();
        int[] senha2 = ss.troca(senha);
        VetStr sind = new VetStr();
        Mult_vet v1 = new Mult_vet();
        int[] v = v1.Multvet(senha2);
        String sindu2 = sind.VetStr(v);
        Connection connection = null;
        Statement statement = null;
        ResultSet rs = null;
        try
          {
            Class.forName( JDBC_DRIVER);
            connection=DriverManager.getConnection(DB_URL,USR,PASS);
            statement =connection.createStatement();
            txt = "SELECT * FROM usupw where usuario = "
            rs = statement.executeQuery(+"'"+usu+"'");
            ResultSetMetaData rsmd = rs.getMetaData();
            txt = "UPDATE usupw SET senha = ";
            txt1 = " where usuario = ";
            statement.executeUpdate(txt+"'"+sindu2+"'"+
                                    txt1+"'"+usu+"'");
            flag1 = 0;
            statement.close();
```

```
          }  // fim do try
         catch (SQLException sqlException) {
            String st="Erro!"+"\n "+sqlException.getMessage();
            JOptionPane.showMessageDialog(null,st);
            sqlException.printStackTrace();
            System.exit( 1 );
          } // fim do catch
          catch (ClassNotFoundException classNotFound) {
            classNotFound.printStackTrace();
            System.exit( 1 );
           } // fim do catch
          finally //instrução e conexão são fechadas adequadamente
            {
               try
                {
                   statement.close();
                   connection.close();
                } // fim do try
               catch ( Exception exception ) {
                   exception.printStackTrace();
                   System.exit( 1 );
               } // fim do catch
            } // fim do finally
          }
         else
            {flag1 = 0;}
      }
    else
       {flag1 = 0;}
  } // fim do while
 } // fim do método pega_categ
} // fim da classe pega_categ
```

```
// Classe principal do aplicativo

package sis;

 import java.io.IOException;
 import java.io.PrintStream;
 import java.net.Socket;
 import java.io.BufferedReader;
 import java.io.InputStreamReader;
 import javax.swing.*;
 import java.io.ObjectInputStream;
 import java.io.ObjectOutputStream;
 import java.awt.*;

public class primaria extends JFrame implements ActionListener {

  public String clicados = "";
  private int contadorDeCliques;
  private static final long serialVersionUID = 1L;
  Dimension tela = Toolkit.getDefaultToolkit().getScreenSize();

  public primaria(String Mens) {
    if (Mens == "")
      { }
    else
      {
       setTitle("Menu Principal");
       String B1 = Mens.substring(0,20);
       String B2 = Mens.substring(21,41);
       String B3 = Mens.substring(42,62);
       String B4 = Mens.substring(63,83);
       String B5 = Mens.substring(84,104);
       setLayout(new FlowLayout(FlowLayout.CENTER));
       add(Criadora.criaBotão(B1, this, "1"));
```

```
      add(Criadora.criaBotão(B2, this, "2"));
      add(Criadora.criaBotão(B3, this, "3"));
      add(Criadora.criaBotão(B4, this, "4"));
      add(Criadora.criaBotão(B5, this, "5"));
      pack();
      setLocationRelativeTo(null);
     }
}

public void actionPerformed(ActionEvent e) {
   if(contadorDeCliques < 1){
      clicados = Criadora.mudaRotulo(e.getActionCommand());
      contadorDeCliques +=1;
   }
}

public String ttela(String a) {
    LabelFrame labelFrame = new LabelFrame();
    labelFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    labelFrame.setSize( 1280, 770 );
    labelFrame.setVisible( true );
    return "";
}

public static void main(String[] args) {

   primaria principal = new primaria("");
   String po = principal.ttela("");
    String z = "";
    while (z == ""){
       String ent1 = "Aplicativo  'Cadastra Usuário   "
       String ent2 = "'Altera Usuário  'Exclui Usuário  'Sair   '";
       primaria janela = new primaria(ent1+ent2);
       janela.setSize( 800, 150 );
```

```
        janela.setDefaultCloseOperation(3);
        janela.setVisible(true);
        String p = janela.clicados;
        while (p=="")
           {p = janela.clicados;
           }
        janela.dispose();
        if (p == "1"){
           System.out.println("Opção 1 - Aplicativo Servidor");
           sas ff = new sas();
        }
        if (p == "2"){
           System.out.println("Opção 2 - Cadastrar Usuário" );
           novo_user novo_user = new novo_user();
        }
        if (p == "3"){
            System.out.println("Opção 3 - Alterar dados de Usuário");
            pega_user pega_user = new pega_user();
        }
        if (p == "4"){
            System.out.println("Opção 4 - Excluindo Usuário");
            exclui_user exclui_user = new exclui_user();
        }
        if (p == "5"){
            System.out.println("Obrigado por usar nossos serviços.");
            z = "1";
            janela.dispose();
            principal.dispose();
            System.exit(0);
         }
     }
  }
}
```

```
// Classe que implementa o aplicativo do servidor

import javax.swing.*; import java.io.IOException; import
java.net.DatagramPacket; import java.net.DatagramSocket; import
java.net.InetAddress; import java.util.List;


    public class sas {

        public sas() {

                String vetorapres = "";

  //            System.out.println("Iniciando o sistema servidor");

                //Declaro o ServerSocket
                DatagramSocket socket = null;

                String[] vetID = new String[4];
                String vetcatc[] = new String[Constantes.ncat+1];
                String vetcatn[] = new String[Constantes.ncat+1];
                String vetsubcatc[] = new String[Constantes.nfig+1];
                String vetsubcatn[] = new String[Constantes.nfig+1];
                int[] vetorcat = new int[37];

                String[][] matapr = new String[5][37];
                for(int se=1;se<37;se++)
                {
                    vetorcat[se] = 0;
                }
                for(int se=1;se<(Constantes.ncat+1);se++)
                {
                    vetcatc[se]="";
```

```
                    vetcatn[se]="";
                }
                for(int se=1;se<(Constantes.nfig+1);se++)
                {
                    vetsubcatc[se]="";
                    vetsubcatn[se]="";
                }
                int l = 0;
                for(int se=0; se<(Constantes.m/Constantes.c);se++)
                   {
                       vetID[se]="";
                   }
                try{
//Recebe a senha do cliente
                       socket = new DatagramSocket(5000);
                       byte data[] = new byte[512];

                       DatagramPacket receivePacket = new DatagramPacket (data,
                       socket.receive(receivePacket);
                       String ent = new String(receivePacket.getData(),0,receive
//                     JOptionPane.showMessageDialog(null, ent, "Senha do Usuári
// Consultando o banco de dados e retornando a senha
                       consulta fras = new consulta();
                       String senha = fras.consulta (ent);
// com senha, busca no BD a categoria referente aos 8 primeiros digitos.

//                     System.out.println("Recebi "+ent);

                       int tamanho = Constantes.m/Constantes.c;
                       String vetsenha[] =new String[tamanho];
                       int inicio = 0;

// quebra a sindrome para buscar a categoria - Ok
                       for (int i = 0;i<tamanho;i++)
```

```
                        {
                            int fim = inicio+Constantes.c;
                            vetsenha[i]=senha.substring(inicio,fim);
                            inicio=fim;
                        }
                        int[] seq = new int[tamanho];

//vetor com as sequencias da senha

                        for (int i = 0;i<tamanho;i++)
                        {
                         consseqcat cons = new consseqcat();
                         seq[i] = cons.consseqcat(vetsenha[i]);
                        }
// com senha, busca no BD a categoria referente aos8 primeiros digitos.
                      // pegando de seis em seis digitos para coletar as sequencia
                        int vetseq[] = new int[Constantes.ncat];
                        VetCateg fas = new VetCateg();
                        vetcatc = fas.Categc();
                         vetcatn = fas.Categn();

                         for (int i=0;i<Constantes.ncat;i++)
                         {
                             vetseq[i]=i;
                         }
                         int vetsubcats[]=new int[Constantes.nfig];
                         VetSubCat fes = new VetSubCat();
                         vetsubcatc = fes.SubCategc();
                         vetsubcatn = fes.SubCategn();
                         vetsubcats = fes.SubCategs();
                         int se;
                         int tam=0;
                         int in=0;
    // quebrando a sindrome em 4 vetores
```

```
                        for(se=0; se<(Constantes.m/Constantes.c);se++)
                        {
                           tam = in+(Constantes.c);
                           vetID[se]=senha.substring(in,tam);
                           in = in + Constantes.c;
                        }

    // pegando a categoria do usuario por parte

                        int v1[] = new int[65];
                        int v2[] = new int[65];
                        int v3[] = new int[65];
                        String v4[] = new String[65];
                        int v5[] = new int[65];
                        String vacat[] = new String[36];

                        for(se=0;se<(Constantes.m/Constantes.c);se++)
                        {
                           consseqcat tt = new consseqcat();
                           int sequencia = tt.consseqcat(vetID[se]);

                       // criando a lista com os aleatorios de 0 - 63
                           // buscando o vetor com os números das categorias
                           gera_numeros nro = new gera_numeros();
                           List numeros = nro.geranumeros(0,63);

                           // pegando apenas os 35 primeiros dos 64 valores - ca
                           for (int sei = 0; sei < 35;sei++)
                           {
                               v1[sei]= (Integer) numeros.get(sei);
//                             System.out.println("v1["+sei+"]= "+v1[sei]);
                           }
                           // verificando se a categoria da sindrome pertence a
                           int band = 0;
```

```
                int lp = 0;
                for (int sei = 0; sei < 35;sei++)
                {
//                  System.out.println("v1["+sei+"]= "+v1[sei]+"   ve
                    if ((Integer.parseInt(vetcatc[v1[sei]])) == (Inte
                    {
                        band = 1;
//                      System.out.println("=======================
                    }
                }
                // se band = 0 a parte da sindrome nao faz parte do v
                if (band == 0)
                {
//                  System.out.println("======================= Nao
                    for (int sei = 0; sei < 64;sei++)
                    {
                        if ((Integer.parseInt(vetcatc[sei]))== (Integ
                        {
                            lp = sei;
//                          System.out.println("posição da sindrome:
                        }
                    }
                    v1[7]= lp;
                }
                int ppos = 0;
                // fazendo uma nova rodada para o aleatorio, para mis
                List nros7 = nro.geranumeros(0,34);
                for (int sei = 0; sei < 35;sei++)
                {
//                  System.out.println("v1["+sei+"]= "+v1[sei]);
                    ppos = (Integer) nros7.get(sei);
                    v2[sei] = v1[ppos];
                }
//
```

```
// pega o sub-vetor que contem a parte da sindrome
// inserir as novas categorias
// verificar se elas ja nao fazem parte do arranjo
//   se nao: fazer o re-arranjo do subvetor
//   se sim: substituir a categoria na outra parte por uma que nao exista
//
// colhendo o vetor de apresentacao e a matriz dos vetores de apresentação
                            for (int sei2 = 0; sei2 < 35; sei2++)
                            {
                                int inic = v2[sei2]*8;
                                int finic = inic + 7;
                                // fazendo um aleatorio para cada uma das subcate
                                List nros1 = nro.geranumeros(inic,finic);
//                              System.out.println("v2["+sei2+"]= "+v2[sei2]+"

                                vetorapres = vetorapres+vetsubcatn[(Integer) nros
                                matapr[se][sei2]=vetsubcatc[(Integer) nros1.get(3
                            }
                        }

                        String dado="";
                        String ent1;
                        int inic=0;
                        int taman=490;
                        String[]opcoes = new String[5];
                        for (int ii=1;ii<5;ii++)
                               {
                                 ent1=vetorapres.substring(inic,taman);
                                 inic = taman;
                                 taman= taman+490;

                // transmitindo o vetor de apresentação

                                 byte data1[] = ent1.getBytes();
```

```
                                DatagramPacket sendPacket = new DatagramPacket(
                                socket.send(sendPacket);

//                              System.out.println("Agora vou buscar a opcao do

                // conecta
                                String mens1 = "7";
                                DatagramPacket receivePacket2 = new DatagramPac
                                int zz = 0;
                                while(zz==0)
                                {
                                    socket.receive(receivePacket2);
//                                  System.out.println("vetor : "+new String(re
                                    mens1 = new String(receivePacket2.getData()
//                                  System.out.println("mensagem recebida: "+me
                                    int zt =Integer.parseInt(mens1);
                                    if (zt>0  & zt < 6)
                                    {
                                        zz = 1;
                                    }
                                }
//                              System.out.println("Colhi do cliente : "+dado);
                                opcoes[ii]= mens1;

                            }
                        //
                        // vericando se a opcao do cliente e igual a armazenada
                        //
//                      for (int i=1;i<5;i++)
//                          {
//                             System.out.println("Opção "+i+" - "+opcoes[i]);
//                          }

//                      System.out.println("-----------------------------------
```

```
                        int[] posmat=new int[4];
                        for(int lin = 0;lin<4;lin++)
                        {
                            for(int col = 0; col < 35; col++)
                            {
                                if (vetID[lin].equals(matapr[lin][col]))
                                {
                                    posmat[lin]=(col/7)+1;
                                    break;
                                }
                            }
                        }
                        int[] opt = new int[6];
                        for(int lin = 0;lin<4;lin++)
                        {
                               if (posmat[lin]==Integer.parseInt(opcoes[lin+1]))
                                   opt[lin+1]=1;
                               else opt[lin+1]=0;

                        }
                        String resposta = "";
                        if (opt[1]==1 & opt[2]==1 & opt[3]==1 & opt[4]==1)
                        {
                           resposta = "SIM";
                        }
                        else
                        {
                            resposta = "NAO";
                        }
// Transmitindo a resposta ao usuário

                         byte data13[] = resposta.getBytes();
                         DatagramPacket sendPacket = new DatagramPacket(data13,
                                 //InetAddress.getLocalHost(),5000);
```

```
                        socket.send(sendPacket);
                       //
                       // utilize o conecta3 para enviar ao cliente a menssagem
                       //
                  }
                  catch ( IOException ioException )
                  {
                            JOptionPane.showMessageDialog(null,"Erro na conex
                            ioException.printStackTrace();
                  } // fim do catch
                  socket.close();

        }
    }


// Classe que retorna a matriz transposta

package sis;

public class Transposta {
    public int[][] Transp (int p [][],int ml,int mc) {
       int[][] pt = new int[mc+1][ml+1];
       for (int l = 0; l < ml; l++) {
          for (int c = 0; c < mc; c++) {
               pt[c][l] = p[l][c];
          }
       }
       return pt;
    }
}


// Classe usada para criar as categorias
```

```
package sis;
 import java.sql.Connection;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;
 import java.sql.Statement;
 import javax.swing.JOptionPane;

public class VetCateg {
  // driver JDBC e URL de banco de dados
  static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
  static final String DB_URL = "jdbc:mysql://localhost/sas";
  static final String USR= "root";
  static final String PASS= "drilica";

  public String[] Categc() {
      int quant = Constantes.ncat;
      int cont =0;
      String[] vetcod = new String[quant];
      String[] vetnome = new String[quant];
      Connection connection = null;
      Statement statement = null;
      ResultSet res = null;
      try
         {
           Class.forName( JDBC_DRIVER);
           connection = DriverManager.getConnection(DB_URL,USR,PASS);
           statement =connection.createStatement();
           res = statement.executeQuery("SELECT * FROM categ");
           ResultSetMetaData rsmd = res.getMetaData();
           if (!res.next()) {
              JOptionPane.showMessageDialog(null,"Tabela vazia");}
```

```
      vetcod[cont]= res.getString(2);
      vetnome[cont]= res.getString(3);
      cont++;
      while (res.next())
       {
         vetcod[cont]= res.getString(2);
         vetnome[cont]= res.getString(3);
         cont++;
       }
      statement.close();
   }  // fim do try
catch (SQLException sqlException)
   {
      String st = "Erro!"+"\n "+sqlException.getMessage();
      JOptionPane.showMessageDialog(null,st);
      sqlException.printStackTrace();
      System.exit( 1 );
   } // fim do catch
catch (ClassNotFoundException classNotFound)
   {
      classNotFound.printStackTrace();
      System.exit( 1 );
   } // fim do catch
finally //instrução e conexão são fechadas adequadamente
  {
    try
       {
          statement.close();
          connection.close();
       } // fim do try
    catch ( Exception exception )
       {
          exception.printStackTrace();
          System.exit( 1 );
```

```
            } // fim do catch
         } // fim do finally
     return vetcod;
 }

 public String[] Categn() {
     int quant = Constantes.ncat;
     int cont =0;
     String[] vetcod = new String[quant];
     String[] vetnome = new String[quant];
     Connection connection = null;
     Statement statement = null;
     ResultSet res = null;
     try
      {
        Class.forName( JDBC_DRIVER);
        connection = DriverManager.getConnection(DB_URL,USR,PASS);
        statement =connection.createStatement();
        res = statement.executeQuery("SELECT * FROM categ");
        ResultSetMetaData rsmd = res.getMetaData();
        if (!res.next())
          {
            JOptionPane.showMessageDialog(null,"Tabela vazia");
          }
        vetcod[cont]= res.getString(2);
        vetnome[cont]= res.getString(3);
        cont++;
        while (res.next())
          {
             vetcod[cont]= res.getString(2);
             vetnome[cont]= res.getString(3);
             cont++;
          }
        statement.close();
```

```
       }  // fim do try
      catch (SQLException sqlException)
       {
          String st = "Erro!"+"\n "+sqlException.getMessage();
          JOptionPane.showMessageDialog(null,st);
          sqlException.printStackTrace();
          System.exit( 1 );
       } // fim do catch
      catch (ClassNotFoundException classNotFound)
        {
           classNotFound.printStackTrace();
           System.exit( 1 );
        } // fim do catch
      finally //instrução e conexão são fechadas adequadamente
        {
          try
            {
               statement.close();
               connection.close();
            } // fim do try
          catch ( Exception exception )
            {
               exception.printStackTrace();
               System.exit( 1 );
            } // fim do catch
        } // fim do finally
    return vetnome;
  }
}

// Classe que transforma vetor em String e vice-versa

package sis;
 import javax.swing.*;
```

```
 import java.lang.*;

public class VetStr {
    public String VetStr(int[] vet) {
        String sind="";
        for(int i=0;i<Constantes.m;i++)
        {
            sind = sind+(String.valueOf(vet[i]));
        }
        return sind;
    }

    public int[] StrVet(String str) {
        int[] vet=new int[Constantes.m];
        int tam=0;
        for(int i=0;i<Constantes.m;i++)
        {
            tam=i+1;
            vet[i] = Integer.valueOf(str.substring(i,tam));
        }
        return vet;
    }
}

// Classe que retorna os dados das subcategorias

package sis;
 import java.sql.Connection;
 import java.sql.DriverManager;
 import java.sql.ResultSet;
 import java.sql.ResultSetMetaData;
 import java.sql.SQLException;
 import java.sql.Statement;
 import javax.swing.JOptionPane;
```

```
public class VetSubCat {
 // driver JDBC e URL de banco de dados
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/sas";
    static final String USR= "root";
    static final String PASS= "drilica";

 public String[] SubCategc() {
  int quant =Constantes.nfig;
  int cont =0;
  String[] vetcod = new String[quant];
  String[] vetnome = new String[quant];
  String txt = "";
  int[] vetseq = new int[quant];
  Connection connection = null;
  Statement statement = null;
  ResultSet res = null;
  try
    {
      Class.forName( JDBC_DRIVER);
      connection = DriverManager.getConnection(DB_URL,USR,PASS);
      statement =connection.createStatement();
      res = statement.executeQuery("SELECT * FROM subcateg");
      ResultSetMetaData rsmd = res.getMetaData();
      if (!res.next()) {
        JOptionPane.showMessageDialog(null,"Tabela vazia");
      }
      vetseq[cont]=res.getInt(1);
      vetcod[cont]= res.getString(2);
      vetnome[cont]= res.getString(3);
      cont++;
      while (res.next()) {
         vetseq[cont]=res.getInt(1);
```

```
        vetcod[cont]= res.getString(2);
        vetnome[cont]= res.getString(3);
        cont++;
     }
     statement.close();
   }  // fim do try
 catch (SQLException sqlException)
 {
    String st = "Erro!"+"\n "+sqlException.getMessage();
    JOptionPane.showMessageDialog(null,st);
    sqlException.printStackTrace();
    System.exit( 1 );
 } // fim do catch
 catch (ClassNotFoundException classNotFound)
 {
   classNotFound.printStackTrace();
   System.exit( 1 );
 } // fim do catch
 finally //instrução e conexão são fechadas adequadamente
   {
     try
       {
        statement.close();
        connection.close();
       } // fim do try
       catch ( Exception exception )
       {
        exception.printStackTrace();
        System.exit( 1 );
       } // fim do catch
     } // fim do finally
 return vetcod;
}
```

```
public int [] SubCategs() {
  int quant =Constantes.nfig;
  int cont =0;
  String[] vetcod = new String[quant];
  String[] vetnome = new String[quant];
  int[] vetseq = new int[quant];
  Connection connection = null;
  Statement statement = null;
  ResultSet res = null;
  try
     {
       Class.forName( JDBC_DRIVER);
       connection = DriverManager.getConnection(DB_URL,USR,PASS);
       statement =connection.createStatement();
       res = statement.executeQuery("SELECT * FROM subcateg");
       ResultSetMetaData rsmd = res.getMetaData();
       if (!res.next()) {
         JOptionPane.showMessageDialog(null,"Tabela vazia");
       }
       vetseq[cont]=res.getInt(1);
       vetcod[cont]= res.getString(2);
       vetnome[cont]= res.getString(3);
       cont++;
       while (res.next())
         {
           vetseq[cont]=res.getInt(1);
           vetcod[cont]= res.getString(2);
           vetnome[cont]= res.getString(3);
           cont++;
         }
       statement.close();
     }  // fim do try
  catch (SQLException sqlException)
     {
```

```
        String st = "Erro!"+"\n "+sqlException.getMessage();
        JOptionPane.showMessageDialog(null,st);
        sqlException.printStackTrace();
        System.exit( 1 );
      } // fim do catch
  catch (ClassNotFoundException classNotFound)
     {
       classNotFound.printStackTrace();
       System.exit( 1 );
      } // fim do catch
  finally // assegura que a instrução e conexão são fechadas adequadamente
    {
      try {
            statement.close();
            connection.close();
           } // fim do try
           catch ( Exception exception )
           {
            exception.printStackTrace();
            System.exit( 1 );
           } // fim do catch
          } // fim do finally
     return vetseq;
}

public String[] SubCategn(){
   int quant =Constantes.nfig;
   int cont =0;
   String[] vetcod = new String[quant];
   String[] vetnome = new String[quant];
   int[] vetseq = new int[quant];
   Connection connection = null;
   Statement statement = null;
   ResultSet res = null;
```

```
try
   {
     Class.forName( JDBC_DRIVER);
     connection = DriverManager.getConnection(DB_URL,USR,PASS);
     statement =connection.createStatement();
     res = statement.executeQuery("SELECT * FROM subcateg");
     ResultSetMetaData rsmd = res.getMetaData();
     if (!res.next()) {
        JOptionPane.showMessageDialog(null,"Tabela vazia");
     }
     vetseq[cont]=res.getInt(1);
     vetcod[cont]= res.getString(2);
     vetnome[cont]= res.getString(3);
     cont++;
     while (res.next())
       {
         vetseq[cont]=res.getInt(1);
         vetcod[cont]= res.getString(2);
         vetnome[cont]= res.getString(3);
         cont++;
       }
       statement.close();
     }  // fim do try
   catch (SQLException sqlException)
     {
        String st = "Erro!"+"\n "+sqlException.getMessage();
        JOptionPane.showMessageDialog(null,st);
        sqlException.printStackTrace();
        System.exit( 1 );
       } // fim do catch
       catch (ClassNotFoundException classNotFound)
       {
        classNotFound.printStackTrace();
        System.exit( 1 );
```

```
          } // fim do catch
          finally //instrução e conexão são fechadas adequadamente
          {
           try
           {
            statement.close();
            connection.close();
           } // fim do try
           catch ( Exception exception )
           {
            exception.printStackTrace();
            System.exit( 1 );
           } // fim do catch
          } // fim do finally
     return vetnome;
 }
}
```

A.2 Código-fonte do Cliente

```
// Classe que monta os vetores de apresentação

public class ButtonFrame extends JFrame {
   public int conex;
   private int retorna;
   private JButton B1; // botão apenas com texto
   private JButton B2; // botão apenas com texto
   private JButton B3; // botão apenas com texto
   private JButton B4; // botão apenas com texto
   private JButton B5; // botão apenas com texto
   private JButton IconB; // botão com ícones
   private JLabel label1;
```

```
  private JLabel label2, label3,label4,label5, label6;
  public String resposta;

 // ButtonFrame adiciona JButtons ao JFrame
public int ButtonFrame( int conexao, String v[]){
   conex =conexao;
   //super( "SIS - Sistema de Segurança" );
   retorna = 1;
   setLayout( new FlowLayout(FlowLayout.LEFT) );
   label1 = new JLabel( "Escolha o número que possui suas imagens");
   label1.setVerticalTextPosition(SwingConstants.BOTTOM);
   label1.setHorizontalTextPosition(SwingConstants.CENTER);
   add(label1);
   label2 = new JLabel("_____________________________________");
   label2.setVerticalTextPosition(SwingConstants.BOTTOM);
   label2.setHorizontalTextPosition(SwingConstants.CENTER);
   add(label2);
   B1 = new JButton( " 1 " ); // botão com texto
   add( B1 ); // adiciona plainJButton ao JFrame
   Icon bug0 = new ImageIcon( getClass().getResource( v[0]) );
   IconB = new JButton( bug0); // configura imagem
   add( IconB ); // adiciona IconB ao JFrame
   Icon bug1 = new ImageIcon( getClass().getResource( v[1]) );
   IconB = new JButton( bug1); // configura imagem
   add( IconB ); // adiciona IconB ao JFrame
   Icon bug2 = new ImageIcon( getClass().getResource( v[2]) );
   IconB = new JButton( bug2); // configura imagem
   add( IconB ); // adiciona IconB ao JFrame
   Icon bug3 = new ImageIcon( getClass().getResource( v[3]) );
   IconB = new JButton( bug3); // configura imagem
   add( IconB ); // adiciona IconB ao JFrame
   Icon bug4 = new ImageIcon( getClass().getResource( v[4]) );
   IconB = new JButton( bug4); // configura imagem
   add( IconB ); // adiciona IconB ao JFrame
```

```
Icon bug5 = new ImageIcon( getClass().getResource( v[5]) );
IconB = new JButton( bug5); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug6 = new ImageIcon( getClass().getResource( v[6]) );
IconB = new JButton( bug6); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
label3 = new JLabel("_____________________________________");
label3.setVerticalTextPosition(SwingConstants.BOTTOM);
label3.setHorizontalTextPosition(SwingConstants.CENTER);
add(label3);
//  segunda sessão
B2 = new JButton( " 2 " ); // botão com texto
add( B2 ); // adiciona plainJButton ao JFrame
Icon bug10 = new ImageIcon( getClass().getResource( v[7]) );
IconB = new JButton( bug10); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug11 = new ImageIcon( getClass().getResource( v[8]) );
IconB = new JButton( bug11); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug12 = new ImageIcon( getClass().getResource( v[9]) );
IconB = new JButton( bug12); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug13 = new ImageIcon( getClass().getResource( v[10]) );
IconB = new JButton( bug13); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug14 = new ImageIcon( getClass().getResource( v[11]) );
IconB = new JButton( bug14); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug15 = new ImageIcon( getClass().getResource( v[12]) );
IconB = new JButton( bug15); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug16 = new ImageIcon( getClass().getResource( v[13]) );
IconB = new JButton( bug16); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
```

```
label4 = new JLabel("_____________________________________");
label4.setVerticalTextPosition(SwingConstants.BOTTOM);
label4.setHorizontalTextPosition(SwingConstants.CENTER);
add(label4);
//  3 sessão
B3 = new JButton( " 3 " ); // botão com texto
add( B3 ); // adiciona plainJButton ao JFrame
Icon bug20 = new ImageIcon( getClass().getResource( v[14]) );
IconB = new JButton( bug20); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug21 = new ImageIcon( getClass().getResource( v[15]) );
IconB = new JButton( bug21); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug22 = new ImageIcon( getClass().getResource( v[16]) );
IconB = new JButton( bug22); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug23 = new ImageIcon( getClass().getResource( v[17]) );
IconB = new JButton( bug23); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug24 = new ImageIcon( getClass().getResource( v[18]) );
IconB = new JButton( bug24); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug25 = new ImageIcon( getClass().getResource( v[19]) );
IconB = new JButton( bug25); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug26 = new ImageIcon( getClass().getResource( v[20]) );
IconB = new JButton( bug26); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
label5 = new JLabel("_____________________________________");
label5.setVerticalTextPosition(SwingConstants.BOTTOM);
label5.setHorizontalTextPosition(SwingConstants.CENTER);
add(label5);
//  4 sessão
B4 = new JButton( " 4 " ); // botão com texto
```

```
add( B4 ); // adiciona plainJButton ao JFrame
Icon bug30 = new ImageIcon( getClass().getResource( v[21]) );
IconB = new JButton( bug30); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug31 = new ImageIcon( getClass().getResource( v[22]) );
IconB = new JButton( bug31); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug32 = new ImageIcon( getClass().getResource( v[23]) );
IconB = new JButton( bug32); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug33 = new ImageIcon( getClass().getResource( v[24]) );
IconB = new JButton( bug33); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug34 = new ImageIcon( getClass().getResource( v[25]) );
IconB = new JButton( bug34); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug35 = new ImageIcon( getClass().getResource( v[26]) );
IconB = new JButton( bug35); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug36 = new ImageIcon( getClass().getResource( v[27]) );
IconB = new JButton( bug36); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
label6 = new JLabel("_____________________________________");
label6.setVerticalTextPosition(SwingConstants.BOTTOM);
label6.setHorizontalTextPosition(SwingConstants.CENTER);
add(label6);
//  5 sessão
B5 = new JButton( " 5 " ); // botão com texto
add( B5 ); // adiciona plainJButton ao JFrame
Icon bug40 = new ImageIcon( getClass().getResource( v[28]) );
IconB = new JButton( bug40); // configura imagem
add( IconB ); // adiciona IconB ao JFrame
Icon bug41 = new ImageIcon( getClass().getResource( v[29]) );
IconB = new JButton( bug41); // configura imagem
```

```
  add( IconB ); // adiciona IconB ao JFrame
  Icon bug42 = new ImageIcon( getClass().getResource( v[30]) );
  IconB = new JButton( bug42); // configura imagem
  add( IconB ); // adiciona IconB ao JFrame
  Icon bug43 = new ImageIcon( getClass().getResource( v[31]) );
  IconB = new JButton( bug43); // configura imagem
  add( IconB ); // adiciona IconB ao JFrame
  Icon bug44 = new ImageIcon( getClass().getResource( v[32]) );
  IconB = new JButton( bug44); // configura imagem
  add( IconB ); // adiciona IconB ao JFrame
  Icon bug45 = new ImageIcon( getClass().getResource( v[33]) );
  IconB = new JButton( bug45); // configura imagem
  add( IconB ); // adiciona IconB ao JFrame
  Icon bug46 = new ImageIcon( getClass().getResource( v[34]) );
  IconB = new JButton( bug46); // configura imagem
  add( IconB ); // adiciona IconB ao JFrame
  // cria novo ButtonHandler para tratamento de evento de botão
  ButtonHandler handler = new ButtonHandler();
  resposta = handler.resposta;
  //IconB.addActionListener( handler );
  B1.addActionListener( handler );
  B2.addActionListener( handler );
  B3.addActionListener( handler );
  B4.addActionListener( handler );
  B5.addActionListener( handler );
  return retorna;
} // fim do construtor ButtonFrame

 // classe interna para tratamento de evento de botão

 private class ButtonHandler implements ActionListener
 {
    public String resposta;
    // trata evento de botão
```

```
      public void actionPerformed( ActionEvent event )
      {
          conecta cone = new conecta();
          resposta = event.getActionCommand();
          String a ="";
         //Declaro o ServerSocket
          if (resposta == " 1 ")
          {a="1";
           dispose();
           }
          if (resposta == " 2 ")
             { a="2";
               dispose();
             }
             if (resposta == " 3 ")
             { a="3";
              dispose();
             }
             if (resposta == " 4 ")
             { a="4";
               dispose();
             }
             if (resposta == " 5 ")
             { a= "5";
               dispose();
             }
             ButtonFrame.this.retorna=1;
             int conex = ButtonFrame.this.conex;
             cone.conecta(conex,a);
               } // fim do método actionPerformed
   } // fim da classe ButtonHandler private interna
} // fim da classe ButtonFrame

// Classe principal do Cliente
```

```
public class Cliente {
   public int conexao;
   public String clicados = "";
   private int contadorDeCliques;
   private static final long serialVersionUID = 1L;

   public Cliente(int conex, String Mens) {
     conexao = conex;
     String v[]=new String [36];
     int tam = 0;
     int in = 0;
     for(int i = 0;i<35;i++)
     {
       tam = in+14;
       v[i]=Mens.substring(in,tam)+".GIF";
       in = in + 14;
     }
     int ret = 0;
     ButtonFrame buttonFrame = new ButtonFrame(); // cria ButtonFrame
     ret = buttonFrame.ButtonFrame(conex,v);
     buttonFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
     buttonFrame.setSize( 730, 520 ); //configura o tamanho do frame
     int largura = buttonFrame.getWidth();
     int altura = buttonFrame.getHeight();
     buttonFrame.setLocation(largura/3,largura/4);
     buttonFrame.setVisible( true );
     JPanel panel = new JPanel();
   }

public static void main(String[] args) {
 String ent= "";
 String t = "";
 String opcao = "FIM";
```

```
//FIM DA DECLARAÇÃO
  LabelTest principal = new LabelTest();
  principal.LabelTest1();

  while (opcao=="FIM")
   {
// Pegando o nome do cliente
    String ID = JOptionPane.showInputDialog(null,"Digite sua Identificação:");
    DatagramSocket socket = null;

    try{
        socket = new DatagramSocket();
        byte data[] = ID.getBytes();
        byte[] ip = {(byte)192, (byte)168, 13, 81};
        InetAddress addr = InetAddress.getByAddress(ip);
        DatagramPacket sendPacket = new DatagramPacket(data, data.length, addr,50
        socket.send(sendPacket);
       }catch(IOException e){
         JOptionPane.showMessageDialog(null,"Erro na conexão \n");
        }
       String ent1= "";
       byte data18[] = new byte[512];
       byte data[] = new byte[512];
       for (int i=1;i<5;i++)
        {
// Pegando parte do vetor de apresentação.
         try{
             DatagramPacket receivePacketzt = new DatagramPacket (data18, data18.
             socket.receive(receivePacketzt);
             String mens12 = new String(receivePacketzt.getData(),0,receivePacket
// Envia a parte do vetor de apresentação a classe Cliente para a apresentação
             Cliente janela = new Cliente(mens12);
// Laço de espera
```

```
            }catch(IOException e){
              System.out.println("Erro de conexão");
              JOptionPane.showMessageDialog(null,"Erro na conexão \n");
            }
          }

// Pegando a resposta final do servidor, autorizando ou nao o cliente de efetuar
        try{
            DatagramPacket receivePacket2 = new DatagramPacket (data, data.length
            socket.receive(receivePacket2);
            String respfinal = new String(receivePacket2.getData(),0,receivePacke
            if (respfinal.equals("SIM"))
              { respfinal = "Operação Autorizada";
                t = "SIM";}
                else
                { respfinal = "Operação não autorizada, sua senha não confere";
                  t = "Não";}
                  JOptionPane.showMessageDialog(null, respfinal, "Resposta do Ser
                  if (t== "SIM" )
                  { opcao = "FIM";}
                  else
                  { opcao = "Não";}
              }catch(IOException e){
                JOptionPane.showMessageDialog(null,"Erro na conexão \n");
            }
       }
    }
}


// Classe que monta a tela principal do cliente

 public class LabelFrame extends JFrame {
   private JLabel label2;
```

```
   private JLabel label3;

  public LabelFrame(){
    super( "SAS - Sistema de Apresentação de Senhas" );
    setLayout( new FlowLayout() );
    Icon bug = new ImageIcon( getClass().getResource("fundo2.gif"));
    label2 = new JLabel( "", bug, SwingConstants.CENTER);
    add( label2 );
    label3 = new JLabel();
    label3.setText( "Label with icon and text at bottom");
    label3.setIcon( bug );
    label3.setHorizontalTextPosition( SwingConstants.CENTER);
    label3.setVerticalTextPosition( SwingConstants.BOTTOM);
    label3.setToolTipText("SAS - Sistema de Apresentação de Senhas");
    add( label3 );
  }
}

// Classe que monta a janela e a deixa visível

public class LabelTest {
   public void LabelTest1()
   {
      LabelFrame labelFrame = new LabelFrame();
      labelFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      labelFrame.setSize( 1280, 770 );
      labelFrame.setVisible( true );
   }
}
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo