VII Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais
Agosto de 2007
Rio de Janeiro, RJ

Minicursos

Editora
Sociedade Brasileira de Computação (SBC)

Organizadores
Luci Pirmez (UFRJ)
Flávia Coimbra Delicato (UFRN)
Luiz Fernando Rust da Costa Carmo (UFRJ)

Realização
Núcleo de Computação Eletrônica da
Universidade Federal do Rio de Janeiro (NCE/UFRJ)

Promoção
Sociedade Brasileira de Computação (SBC)

Organizadores

Luci Pirmez

Flávia Coimbra Delicato

Luiz Fernando Rust da Costa Carmo

Minicursos do VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2007)

Porto Alegre

Sociedade Brasileira de Computação – SBC

2022

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (7. : 27-31 ago. 2007 : Rio de Janeiro)
Minicursos do VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2007) [recurso eletrônico] / organização: Luci Pirmez ; Flávia Coimbra Delicato ; Luiz Fernando Rust da Costa Carmo. Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2022.
199 p. ; PDF

Inclui bibliografia
ISBN 978-85-7669-503-5 (e-book)

1. Ciência da Computação. 2. Segurança da informação. 3. Sistemas computacionais. I. Pirmez, Luci. II. Delicato, Flávia Coimbra. III. Carmo, Luiz Fernando Rust da Costa. IV. Universidade Federal do Rio de Janeiro. V. Sociedade Brasileira de Computação. VI. Título.

CDU 004(063)

Ficha catalográfica elaborada por Jéssica Paola Macedo Müller – CRB-10/2662

Biblioteca Digital da SBC – SBC OpenLib

Índice para catálogo sistemático:

1. Ciência e tecnologia informáticas : Computação : Processamento de dados – Publicação de conferências, congressos, simpósios etc... 004(063)

Sumário

Capítulo 1

Forense Computacional: fundamentos, tecnologias e desafios atuais

Evandro Pereira[1], Leonardo Lemes Fagundes[2], Paulo Neukamp[2], Glauco Ludwig[1], Marlom Konrath[2]

Universidade do Vale do Rio dos Sinos - Unisinos

[1]{evandrodvp, llemes, glaucol}@unisinos.br; [2]{pneukamp, marlomk}@gmail.com

Abstract

This chapter presents a very current theme that has received substantial attention from both academic and industrial communities: the computer forensics. The scientific and systematic inspection of computational environments, with the goal of figuring out and reconstruct events, still has lot of open research topics. On industry, the interest on this subject is supported by an increasing amount of criminal investigations having digital data as its main evidences. In this chapter, we present basic notions of computer forensics and tools that can be used to collect, keep and analyze evidences. The anatomy of some malicious codes and case studies are also used as a complement on the subject and as a way of increasing its comprehension.

Resumo

Este capítulo trata de um tema bastante atual e que tem recebido significativa atenção tanto da comunidade científica quanto da indústria: a forense computacional. A inspeção científica e sistemática em ambientes computacionais com o objetivo de tentar reconstituir eventos apresenta ainda vários tópicos de pesquisa em aberto. Na indústria, o interesse justifica-se pela grande quantidade de investigações criminais, cujas principais evidências estão armazenadas em formato digital. São apresentados neste capítulo noções de forense computacional e ferramentas que podem ser utilizadas para auxiliar na coleta, manutenção e análise de evidências. A anatomia de alguns códigos maliciosos e estudos de caso complementam o tema e auxiliam no entendimento do assunto.

1.1 Introdução

O *cyber crime* diferencia-se dos crimes tradicionais em função do seu modo de operação, pois envolve a utilização de dispositivos eletrônicos, de computadores e da Internet para a execução de ação ou omissão, típica, antijurídica e culpável [Chawki 2005]. Entretanto, tanto quanto os autores dos atos ilícitos convencionais – aqueles cometidos sem o uso de computadores – os responsáveis por crimes virtuais devem ser identificados, julgados e penalizados. Contudo, essa é uma tarefa complexa devido à possibilidade de anonimato dos contraventores e ao fato de que as evidências do crime podem estar distribuídas em diversos servidores espalhados pela Internet, possivelmente em computadores localizados em regiões distantes daquelas onde as vítimas se encontram.

Conforme estatísticas apresentadas no *Internet Crime Report*, um relatório anual elaborado conjuntamente pelo *National White Collar Crime Center* (NWCCC) e pelo *Federal Bureau of Investigation* (FBI), que reúne diversas informações sobre tendências e padrões observados nos crimes praticados via Internet, somente em 2006, aproximadamente U$ 200.000.000,00 (duzentos milhões de dólares) foram perdidos em conseqüência de diferentes tipos de fraudes [NWCCC and FBI 2006]. Entre os crimes apresentados pode-se citar o caso *Nigerian Letter Fraud*, em que uma vítima é induzida a auxiliar um agente de governo estrangeiro a movimentar grandes somas de dinheiro para fora do seu país em troca de uma generosa comissão. Após fornecer os dados da conta bancária, ao invés de ter a comissão depositada, as vítimas são roubadas. A pesquisa indica ainda que, assim como neste golpe, aproximadamente 76% dos casos reportados tiveram como meio de comunicação com a vítima o correio eletrônico.

Entre as ocorrências mais comuns estão a calúnia, difamação e injúria[1] via e-mail, o roubo de informações confidenciais e a remoção de arquivos. Essas ações são motivadas pelo interesse de causar constrangimento ou algum tipo de perda à vítima e, normalmente, são protagonizadas por colaboradores insatisfeitos ou por concorrentes de um determinado segmento de mercado. Além disso, crimes como pedofilia, fraudes e o tráfico de drogas via Internet também são atos ilícitos constantemente realizados com o apoio de computadores. Institutos de pesquisa como o WebSense indicam que, em 2007, o crime organizado se unirá à *crackers* para comprar, vender e negociar *commodities*, como *kits* de ferramentas prontas para ataques virtuais e golpes utilizando vulnerabilidades recém descobertas [Bessa 2006].

Com a finalidade de auxiliar na investigação de tais crimes, se faz necessário o uso da Forense Computacional. De acordo com [Palmer and Corporation 2001] a Forense Computacional pode ser definida como a inspeção científica e sistemática em ambientes computacionais, com a finalidade de angariar evidências derivadas de fontes digitais, tendo como objetivo, promover a reconstituição dos eventos encontrados (podendo assim, determinar se o ambiente em análise foi utilizado na realização de atividades ilegais ou não autorizadas).

---

[1] **Calúnia:** consiste em atribuir, falsamente, a alguém a responsabilidade pela prática de um fato determinado definido como crime. **Difamação:** consiste em atribuir a alguém fato determinado ofensivo à sua reputação. **Injúria:** consiste em atribuir a alguém qualidade negativa, que ofenda sua dignidade ou decoro.

Este capítulo trata de um tema bastante atual e que tem recebido significativa atenção tanto da comunidade científica quanto da indústria: a forense computacional. No âmbito acadêmico, o interesse deve-se à quantidade de questões de pesquisa em aberto, fruto do constante surgimento de novas tecnologias, de inúmeras vulnerabilidades e de ameaças cada vez mais sofisticadas [Richard and Roussev 2006]. Já no contexto da indústria, tal interesse justifica-se pela grande quantidade de investigações criminais, cujas principais evidências estão armazenadas em formato digital, e pela carência de profissionais especializados para realizar o processo de investigação de maneira precisa a fim de que o resultado obtido possa ser aceito por juízes nos tribunais, tanto como peça de acusação quanto de defesa [Viotto 2007].

O presente capítulo está organizado da seguinte forma. A Seção 1.2 apresenta análises referentes à anatomia de diversos tipos de códigos maliciosos (*malwares*) e um cenário de ataque, cujas evidências deixadas por *worms* e *rootkits* serão detalhadamente descritas. Na Seção 1.3 é realizada uma introdução à Forense Computacional. Já na Seção 1.4 são realizadas demonstrações sobre a utilização de um conjunto de ferramentas que oferecem suporte a cada uma das etapas do processo de investigação forense, bem como são mencionadas técnicas anti-forense normalmente empregadas por invasores no intuito de ocultar as evidências das ações realizadas. A Seção 1.5 apresenta quatro estudos de caso que consolidam os conceitos estudados nos capítulos anteriores e permitem ao leitor um primeiro contato com a prática forense. A Seção 1.6 demonstra os desafios atuais em forense digital e este trabalho se encerra apresentando as considerações finais na Seção 1.7.

1.2 Códigos Maliciosos

Segundo [Skoudis and Zeltser 2003], códigos maliciosos são conjuntos de instruções executadas em um computador e que fazem o sistema realizar algo que um atacante deseja. Esta definição é bastante genérica, buscando assim abordar todas as categorias de *software* que são comumente consideradas quando se fala de *malware*.

Nos Estados Unidos, a maioria das ações legais contra *malware* é tomada pela *Federal Trade Commission* (FTC), sendo que o seu foco principal é companhias que produzem ou distribuem *spyware* [Payton 2006]. Como na maioria dos casos estas companhias estão fora dos EUA, a FTC publicou em junho de 2005 uma recomendação para o congresso de mudança na lei para permitir cooperação com órgãos internacionais [FTC 2005]. No entanto, este tipo de iniciativa ainda é bastante incomum, o que facilita a ação de atacantes que se utilizam de servidores geograficamente dispersos.

Esta seção aborda as principais categorias de *malware* existentes. A classificação aqui apresentada é fortemente influenciada por [Skoudis and Zeltser 2003].

1.2.1 Vírus

Os primeiros códigos com capacidade de se auto-replicarem de que se têm notícia surgiram em 1962, nos laboratórios Bell, com o jogo Darwin, onde programas “lutavam” entre si para sobreviverem [Aleph-Null 1971]. No entanto, a utilização do termo vírus para referenciar-se a programas com capacidade de se auto-replicarem só surgiu em 1984, com o artigo “Computer Viruses – Theory and Experiments” de Fred

Cohen [Cohen 1987]. Nele, o autor demonstra ainda que não há algoritmo capaz de detectar todos os possíveis vírus de computador.

Embora recentemente novas definições estejam sendo propostas [Bonfante et al. 2007, Case and Moelius 2007], a definição de vírus aqui apresentada segue o modelo utilizado por [Adleman 1990] em que um vírus é considerado uma função computável que infecta qualquer programa. Um programa infectado pode realizar então três ações disparadas conforme as entradas: (a) executar no programa hospedeiro e propagar a infecção, (b) danificar o sistema ou (c) imitar o programa hospedeiro.

Uma das características de um vírus é a necessidade de anexar-se a um "programa hospedeiro" para funcionar [Skoudis and Zeltser 2003]. O alvo neste caso pode ser um arquivo executável, o setor de inicialização, um documento que suporte macros ou um arquivo de script. Os vírus propagam-se normalmente através de mídias removíveis, e-mails, downloads e diretórios compartilhados. Os tipos abaixo apresentados seguem a classificação apresentada em [Kaspersky 2007]:

Os chamados "vírus acompanhantes" (*companion viruses*) são aqueles nos quais o vírus não infecta um arquivo executável, mas utiliza o mesmo nome de um arquivo existente, com uma extensão diferente, que é preferida pelo Sistema Operacional [Sophos 2001]. Na Figura 1 um exemplo desta técnica é mostrada. Se o usuário clicar em Iniciar => Executar e digitar "notepad", sem aspas, o arquivo notepad.com é quem será executado.

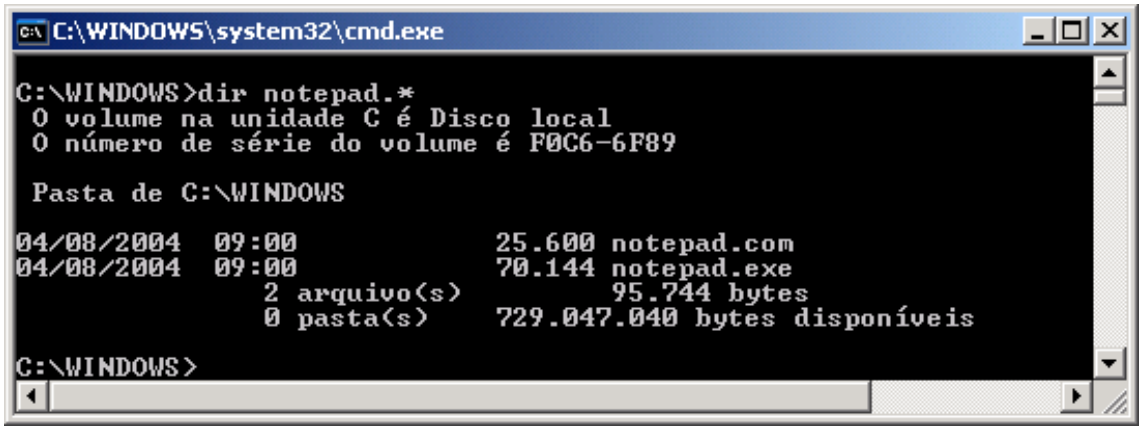


Figura 1. Exemplo de Vírus Acompanhante

No caso de vírus que infectam um arquivo executável existente, modificando o seu código, duas técnicas de infecção são possíveis: (a) infectar o início do arquivo e (b) infectar o final do arquivo. Na primeira técnica, o vírus irá adicionar seu código antes do início do código do programa hospedeiro. Um exemplo de aplicação de tal procedimento é o vírus Nimda. Já ao infectar o final do arquivo, como o próprio nome diz, o código do vírus é adicionado no final do arquivo. Para ser executado, parte do início do arquivo original é sobrescrita com instruções que realizam um salto de execução para o início do código do vírus. Ao final deste último, a parte sobrescrita do arquivo original é novamente adicionada e um segundo salto desviará o fluxo de execução para o código original. A maioria dos vírus utiliza esta técnica, pois a mesma

possui implementação mais fácil, embora vírus mais robustos geralmente utilizem mais de uma técnica de infecção.

Além de arquivos executáveis, alguns vírus podem instalar-se nos primeiros setores lidos durante a inicialização de um Sistema Operacional. Estes setores são tipicamente os primeiros setores de um disco rígido e é aonde a máquina irá inicialmente procurar pelo código a ser executado quando a mesma é ligada. O setor mestre de inicialização (também chamado Master Boot Record ou MBR) é o setor mais comumente utilizado, embora os primeiros setores de uma partição também possam ser utilizados. Um dos vírus mais conhecidos que infectava a MBR foi o Michelangelo [IBM 2007]. No entanto, infectar a MBR a partir do Windows NT 4 tornou-se inócuo, pois este SO e seus derivados acessam diretamente o hardware [Symantec 2007a].

Vírus de macro, por sua vez, são possíveis graças à capacidade de alguns *softwares* de interpretar código presente dentro de arquivos de dados. Os exemplos mais clássicos desta classe são os vírus de macro do Microsoft Word. Por possuir a capacidade de interpretar código em uma linguagem denominada Visual Basic for Applications, vários vírus desta categoria surgiram no Word no final da década passada. Um dos mais conhecidos foi o vírus Melissa [CERT 1999], que era capaz de enviar e-mails para a lista de contados do usuário, contendo como anexo um documento infectado, gerado pelo vírus ou do próprio usuário.

Por último, podem existir vírus com outros alvos em específico, como o vírus de scripts, que se espalham através de arquivos que permitem a execução de códigos em scripts [Kaspersky 2007]. O vírus PHP.Pirus, por exemplo, foi o primeiro vírus escrito na linguagem PHP [Symantec 2007b] e pode residir em servidores que possuem suporte a esta linguagem.

Em [Subramanya and Lakshminarasimhan 2001] os autores classificam os vírus em 5 gerações:

1. Vírus Simples: não faziam nada muito significativo além de replicar-se. Freqüentemente consumiam toda a memória por replicarem-se indiscriminadamente.
2. Vírus com auto-reconhecimento: detectam um sistema já infectado através de alguma assinatura, não gerando duplicidade de infecção.
3. Vírus invisíveis (*stealth*): interceptam chamadas do sistema para tentar esconder a sua presença.
4. Vírus com arsenal (*armored*): empregam técnicas para dificultar a análise de seu código e podem realizar ataques diretos a antivírus presentes no sistema.
5. Vírus polimórficos: também chamados de auto-mutantes, este tipo de vírus infecta novos alvos com versões modificadas ou criptografadas de si mesmo.

1.2.2 Backdoor

Segundo [Skoudis and Zeltser 2003], um *software* que permite uma “entrada pelos fundos” (*backdoor*) é um programa que habilita um atacante a furar os controles normais de segurança de um sistema, ganhando acesso ao mesmo através de um caminho alternativo. Segundo [Zhang and Paxson 2000] normalmente um *backdoor*

opera sobre o protocolo Telnet, Rlogin ou SSH e tipicamente fornece uma das seguintes funcionalidades ao atacante:

1. Aumento dos Privilégios Locais (*Local Escalation of Privileges*): permite que um usuário normal execute programas com privilégios de superusuário.
2. Execução Remota de Comandos: permite que o atacante envie comandos para a máquina alvo e obtenha as respostas geradas pela execução dos mesmos.
3. Acesso Remoto à Linha de Comando: permite que o atacante utilize um *shell* remoto na máquina alvo, de onde poderá realizar qualquer operação como se estivesse utilizando o teclado em frente à máquina real.
4. Controle Remoto da Interface Gráfica: permite ao atacante observar e interferir na interface gráfica à qual o usuário local está conectado, fornecendo assim acesso pleno à máquina.

Um dos *backdoors* mais conhecidos é o netcat [Giacobbi 2007], também chamado de "canivete suíço". Este apelido se deve ao fato de o netcat poder executar praticamente qualquer comando em uma máquina e desviar a entrada e/ou saída padrão para uma conexão de rede. Assim, pode-se programar este *software* para ficar escutando uma porta TCP ou UDP e esperando por conexões em uma máquina alvo. Ao estabelecer uma conexão, o netcat executa um comando e passa a desviar toda a saída para a conexão estabelecida. Na outra ponta, um atacante executa o programa numa versão cliente e passa a desviar a entrada padrão para o programa executando na máquina alvo e obtendo os resultados gerados nesta.

A Figura 2 ilustra um exemplo de como o netcat pode ser utilizado para fornecer acesso remoto à linha de comando. Outros exemplos de programas que podem ser utilizados como *backdoors* são Virtual Network Computing (VNC) [Cambridge 2007] e Loki [Phrack 2007]. O primeiro permite que se capture e manipule a interface gráfica do usuário e o segundo é um *backdoor* que utiliza o protocolo ICMP, não necessitando portanto, abrir portas TCP ou UDP.

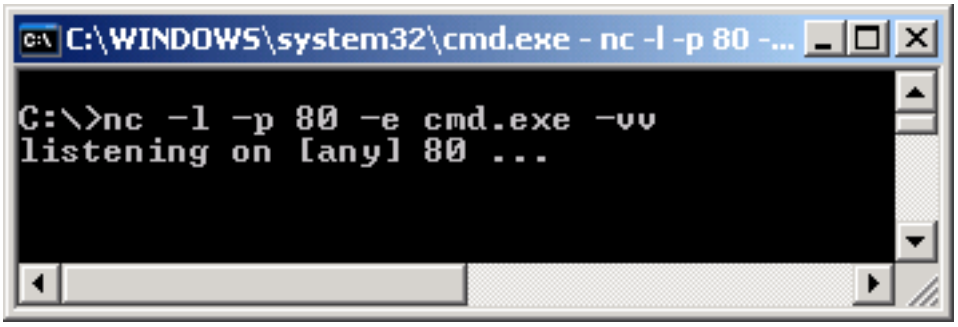


Figura 2. Exemplo de utilização do Netcat

1.2.3 Cavalos de Tróia

Segundo [Kolter and Maloof 2006], um cavalo de tróia (*Trojan Horse*) é um programa que se mascara ou aparenta possuir um propósito benigno, mas que arbitrariamente realiza funções maliciosas. Normalmente, um cavalo de tróia não é capaz de replicar-se

automaticamente [Haagman and Ghavalas 2005]. Segundo [Newman 2006], quando um cavalo de tróia é ativado os resultados podem variar, mas freqüentemente estes programas criam *backdoors* (veja Seção 1.2.2) permitindo acesso remoto ou vazamento de informações.

Para não ser descoberto, é comum que cavalos de tróia tentem esconder seus rastros, bem como disfarçar-se de programas legítimos. Para disfarçar-se de programa legítimo um cavalo de tróia muda seu nome para nomes comuns de serem encontrados no SO hospedeiro, como explorer.exe, iexplore.exe, svchost.exe, csrss.exe*, services.exe*, smss.exe*, spoolsv.exe*, System*, System Idle Process* ou winlogon.exe* no Windows ou init, cron ou httpd em sistemas *nix.

A mostra um exemplo de como se pode renomear um programa para escondê-lo entre os processos normais do sistema. Nela, primeiramente o nome do arquivo executável netcat.exe é trocado para explorer.exe. Em seguida o programa é executado e pode-se notar que seu nome aparece na lista de tarefas do Windows e que este programa passa a escutar a porta 80.

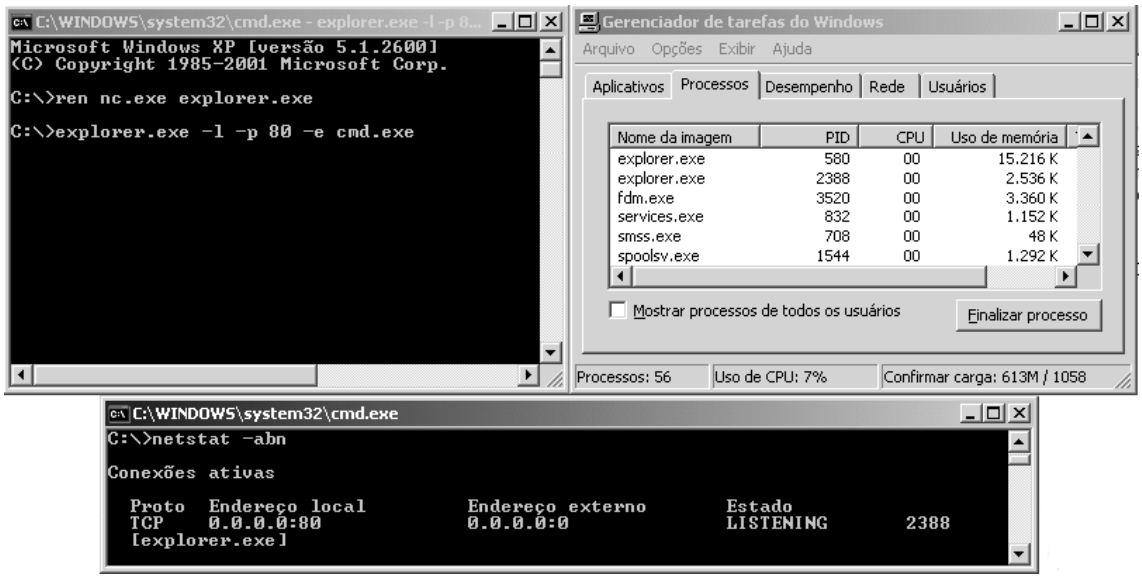


Figura 3. Exemplo de como disfarçar um Cavalo de Tróia

Segundo [Skoudis and Zeltser 2003] outra prática comum é um programa cavalo de tróia ser combinado com um programa executável normal em um único executável. Programas que juntam dois ou mais executáveis são chamados de *wrappers* e o uso destes *softwares* habilita um atacante a mesclar praticamente qualquer cavalo de tróia com qualquer programa legítimo. Como a maioria dos cavalos de tróia não irá gerar saídas para a tela, o programa combinado aparenta ser apenas o outro, escondendo sua atividade do usuário.

Um caso famoso de um cavalo de tróia foi reportado em 2002, quando um atacante invadiu o site oficial do *software tcpdump* e adicionou códigos nos scripts de instalação que permitiriam um atacante executar remotamente comandos em uma

máquina Linux [CERT 2002b]. Em ambiente Windows, programas comumente utilizados como cavalos de tróia são sub7 [Crapanzano 2003] e BO2K [Dildog 2007].

1.2.4 Spyware

De acordo com [Payton 2006], um *spyware* é um *software* que auxilia a coleta de informações sobre uma pessoa ou organização sem o seu conhecimento, que pode enviar tais dados para outra identidade sem o seu consentimento ou que toma controle do computador sem que o usuário saiba disso. De todos os tipos de *malware* existentes *spyware* é o mais comumente encontrado. Segundo [Weiss 2005], uma pesquisa conduzida pela Dell em setembro de 2004 estimou que aproximadamente 90% dos PCs com Windows possuíam no mínimo um *spyware*. Este tipo de *software* foi responsável por metade das falhas em ambientes Windows reportados por usuários da Microsoft [Shukla and Nah 2005]. Outro estudo apontou uma média de 25 *spywares* por PC [Sipior et al. 2005].

Segundo [Solove and Rotenberg 2003], Xupiter e Gator eram as empresas líder na disseminação de programas tidos como *spywares* em 2003. Estimava-se que neste período aproximadamente 35 milhões de PCs nos EUA possuíam algum *software* da Gator Companion. Atualmente não há um consenso quanto às classificações e tipos de *spyware* existentes. Algumas proposições englobam inclusive categorias de *software* que são nesta seção apresentados em separado, como *keyloggers* por exemplo. Em [Payton 2006], *spywares* são classificados quanto ao seu tipo em:

- *Advertising displays*: códigos que mostram anúncios de vários tipos no PC do usuário.
- *Automatic download software*: instalam outros *softwares* sem o conhecimento e consentimento do usuário.
- *Autonomous spyware*: programas que são executados fora de um navegador web, normalmente sendo executados na inicialização e permanecendo indetectáveis pelo usuário.
- *Tracking software*: programas que monitoram os hábitos de um usuário ou os dados fornecidos por ele em páginas web e enviam estas informações pela Internet para algum servidor remoto.

Para reduzir a presença deste tipo de código malicioso, diversas companhias desenvolveram produtos *anti-spyware*. Entre as soluções mais famosas estão: *Spybot Search and Destroy* [Kolla 2007], *Ad-Aware* [Lavasoft 2007], *Pest Patrol* [Etrust 2007] e *Microsoft Windows Defender* [Microsoft 2007a]. Estas ferramentas geralmente já possuem ferramentas para automaticamente eliminar os *spywares* encontrados. Recentemente os *softwares* de antivírus começaram também a detectar e remover este tipo de código malicioso.

1.2.5 Worms

Os *worms* são caracterizados como programas que se auto-propagam por meio de uma rede de computadores explorando vulnerabilidades em serviços usados em larga escala (como programas de e-mail, programas de mensagens instantâneas e compartilhamentos

de rede) [Zou et al. 2005]. Um *worm* diferencia-se de um vírus pela sua característica de auto-replicação, ou seja, não necessita de intervenção humana para se disseminar.

A primeira implementação de um *worm* foi realizada em 1978 por John Shock e Jon Hupp, pesquisadores da Xerox. Na época, a intenção dos autores foi desenvolver um *software* capaz de encontrar processadores ociosos disponíveis na rede da Xerox e, assim, designar tarefas para esses processadores computarem. Contudo, a partir de 1980, o conceito de *worms* passou a ser utilizado por atacantes para espalhar *malwares* de forma rápida e abrangente. Atualmente, os *worms* são classificados como uma das maiores, ameaças virtuais, chegando a atingir 65% dos incidentes de segurança reportados ao Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) no período de Janeiro a Março de 2007 [CERT.br 2007].

Comumente, os *worms* executam uma varredura em busca de sistemas e serviços vulneráveis. Quando máquinas-alvo são identificadas, os *worms* exploram as vulnerabilidades encontradas e se propagam infectando essas máquinas. Após, a procura por novos alvos recomeça [Klaus and Nelson 2001]. Um *worm* pode ter as mais diversas finalidades, como simplesmente espalhar-se consumindo largura de banda, causar ataques de negação de serviço, apagar arquivos, enviar arquivos por e-mail e, principalmente, instalar outros *malwares* como *keyloggers*, *rootkits* e *backdoors*.

No decorrer dos últimos anos, vários *worms* tornaram-se conhecidos por suas habilidades de rápida disseminação e pelos danos que causaram. Como exemplo, pode-se citar o *Code Red Worm* [Tham 2001]. O *Code Red* foi detectado em 12 de julho de 2001 pela empresa *eEye Digital Security* [eEye 2007], explorando uma vulnerabilidade já conhecida [CERT 2002a] em servidores *Internet Information Service* (IIS) da *Microsoft*. A vulnerabilidade havia sido reportada um mês antes de o *Code Red* entrar em ação, indicando o despreparo de muitos administradores de redes. De acordo com o CERT, há uma estimativa de que mais de 250.000 servidores foram infectados em apenas 9 horas.

A vulnerabilidade explorada pelo *Code Red* está baseada no fato de que quando um servidor ISS é instalado, diversas extensões do ISAPI (*Internet Services Application Programming Interface*) são instaladas automaticamente. O ISAPI permite aos programadores estender as potencialidades de um servidor ISS utilizando bibliotecas DLLs. A biblioteca “idq.dll”, utilizada pelo serviço de indexação do ISS, continha erros de programação, pois não realizava a checagem de strings longas de entrada. Assim, permitindo que atacantes ocasionassem um buffer overflow ao enviar dados a essa DLL. Esta vulnerabilidade em específico dava acesso remoto de super-usuário ao sistema. Todas as versões do ISS que acompanhavam por padrão os sistemas Windows NT 4.0, Windows 2000 e Windows XP beta estavam vulneráveis [CERT 2002a].

As operações realizadas pelo Code Red podem ser classificadas em 6 passos [Tham 2001]:

1. O *worm* tenta conectar-se na porta TCP 80 de máquinas selecionados randomicamente, assumindo que um servidor web será encontrado. No caso de obter conexão, o atacante envia uma requisição HTTP GET para o alvo. A presença da seguinte string em um *log* de um servidor web pode indicar o comprometimento do servidor por parte do Code Red:

/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9
090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u90
90%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a

2. Uma vez executado, o *worm* faz uma busca pelo arquivo c:\notworm. Caso esse arquivo seja encontrado, a execução do Code Red é cancelada. Caso contrário, o *worm* gera 100 threads;
3. Se a data do sistema invadido foi anterior ao dia 20 do mês, 99 threads são usadas para a tentativa de invadir mais sistemas;
4. A centésima thread é responsável por modificar a página principal do servidor web (caso a linguagem padrão do sistema o seja o inglês), a qual passará a exibir a mensagem: "*HELLO! Welcome to http://www.worm.com! Hacked by Chinese!*". Essas alterações são realizadas sem modificar o arquivo da página no disco físico, mas sim, na memória;
5. Caso a data do sistema for entre os dias 20 e 28, o *worm* tenta então disparar um ataque de negação de serviço ao domínio www.whitehouse.com;
6. Caso a data seja superior ao dia 28, a execução do *worm* é cancelada.

Ao longo dos tempos, algumas variantes do *Code Red* acabaram sendo desenvolvidas (ex: *Code Red II, Code Blue, Code Green*). Mais detalhes sobre as mesmas podem ser encontradas em [Tham 2001].

1.2.6 Key Loggers

Os *keyloggers* são definidos como um tipo de spyware cuja finalidade é capturar tudo o que for digitado em um determinado computador [CERT.br 2007]. As intenções no uso de um *keylogger* podem ser as mais diversas, como, por exemplo, monitorar as atividades dos funcionários de uma determinada empresa, capturar conversas em programas de mensagens instantâneas e, inclusive, capturar informações e senhas bancárias. As informações obtidas podem então ser enviadas pela Internet para o gerente da empresa ou para um atacante, tudo sem o consentimento da vítima que faz uso do computador infectado.

De acordo com [SecurityFocus 2007], os *keyloggers* podem ser classificados em três diferentes tipos:

Hardware keylogger: trata-se de um dispositivo físico posicionado entre o teclado e o computador da vítima. Apesar de serem rápidos (por possuírem um hardware dedicado para executar sua função) e por não serem detectados por mecanismos como anti-vírus e anti-spyware, os mesmos podem ser detectados

visualmente por uma pessoa. Além disso, possuem espaço de armazenamento limitado e necessitam de acesso físico a máquina vítima para serem instalados;

Sofware keylogger usando um mecanismo de *hooking*: um *hook* trata-se de uma rotina que tem como objetivo "ficar no meio do caminho" do tratamento normal da execução de informações do Sistema Operacional (SO). Para isso, os programadores utilizam funções disponibilizadas pela API (*Aplication Program Interface*) do SO. Essas funções são responsáveis por capturar as mensagens do sistema (assim como as teclas que são pressionadas) antes que as mesmas sejam tratadas pelas devidas rotinas de tratamento. *Keyloggers* desse tipo normalmente possuem um módulo executável, que dispara a execução do aplicativo, e uma biblioteca que contém as rotinas para a captura das informações desejadas. Esses *keyloggers* podem ser instalados remotamente, no entanto, são os mais lentos e facilmente detectáveis por programas como anti-vírus e anti-spywares;

Kernel keylogger: este tipo de *keylogger* trabalha no nível do kernel e usa suas próprias rotinas para receber os dados diretamente dos dispositivos de entrada (no caso, o teclado). É o método mais difícil de ser desenvolvido (por exigir um elevado conhecimento de programação) e também de ser detectado (por substituir as rotinas padrão do SO e serem inicializados como parte do próprio sistema). Pelo fato de trabalharem no núcleo do sistema, não são capazes de capturar informações que são trocadas diretamente no nível de aplicações (ex: operações de copiar e colar e operações de autocompletar).

Atualmente, um dos *keyloggers* que mais se destaca é o *Perfect Keylogger* (BPK), da *Blazing Tools Software* [BlazingTools 2007]. O BPK trata-se de um *keylogger* baseado em *hooking* que pode ser facilmente instalado em sistemas operacionais Windows. Por possuir um processo de instalação bastante simplificado e uma interface gráfica amigável, qualquer usuário, mesmo sem experiência em segurança de computadores, é capaz de utilizar o programa e monitorar atividades alheias.

A versão completa do BPK pode ser obtida por um preço acessível (U$ 34,95) via o site do seu fabricante. Uma versão de avaliação do produto também é disponibilizada, o que facilita ainda mais a disseminação do seu uso. A Figura 4 mostra a interface de configurações gerais do BPK. Entre suas principais funcionalidades pode-se citar:

- capturar tudo o que for digitado no computador;
- trabalhar em modo invisível (ocultando o processo em execução do gerenciador de tarefas, tornando o programa invisível à lista de startup e removendo o programa do menu de inicialização e da lista de desinstalação de programas do Windows);
- renomear os arquivos que compõem o aplicativo para que tenham um nome qualquer (dificultando a busca no sistema de arquivos pelo nome original do *keylogger*);
- possibilitar que os processos de instalação, atualização e desinstalação possam ser realizados remotamente;
- registrar em arquivos de log os sites que foram visitados;

- enviar as informações capturadas para um determinado e-mail.

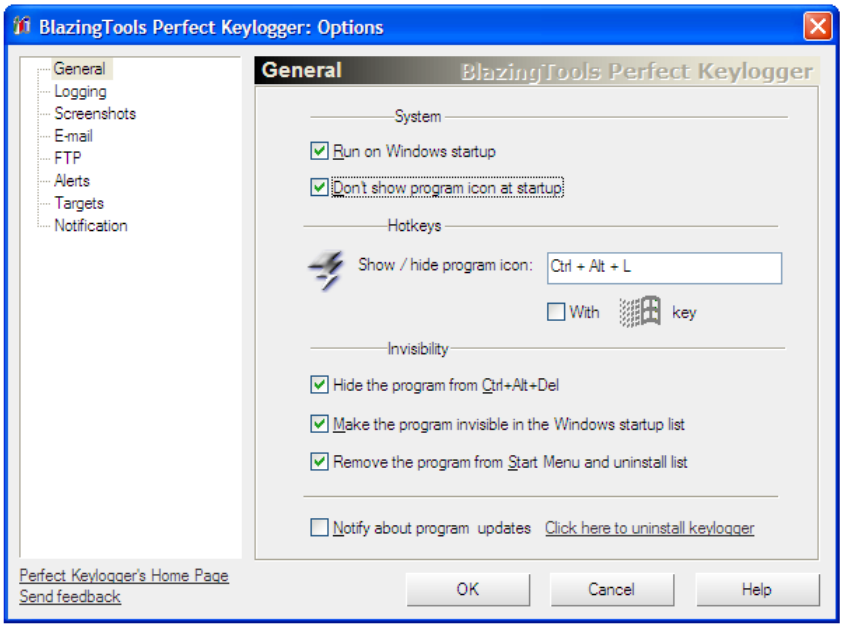


Figura 4. Tela de configurações gerais do *Perfect Keylogger*

Pelo fato do BPK ter a possibilidade de rodar em modo background e ficar invisível ao gerenciador de tarefas do Windows, é preciso do auxílio de uma ferramenta como a *SysInternals' Process Explorer* [Microsoft 2007b] para poder visualizar seu processo em execução. A Figura 5 apresenta o processo spyware executando em *background* (representado por bkp.exe).

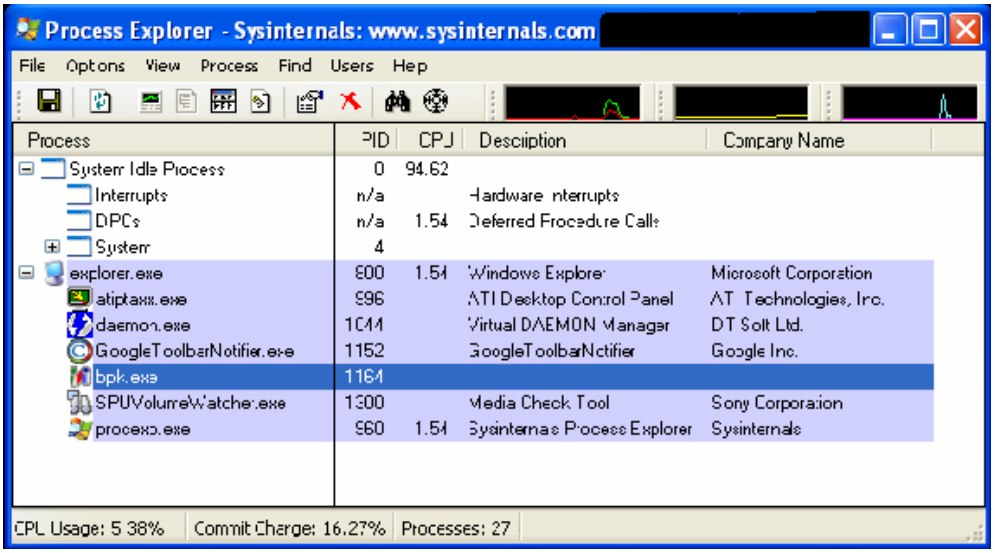


Figura 5. A ferramenta *SysInternals' Process Explorer* exibindo a execução do BPK em *modo background*

Nos últimos anos, foi desenvolvido também o conceito de *screenlogger* [CERT.br 2007]. Um *screenlogger* tem a finalidade de obter uma cópia (*screenshot*) da tela do computador da vítima assim que um clicar do mouse for efetuado. Os *screenloggers* foram uma alternativa utilizada por atacantes para que os mesmos pudessem capturar senhas bancárias quando organizações passaram a utilizar o conceito de teclados virtuais (modelo de teclado que permite que as senhas sejam informadas utilizando um mouse). Essa funcionalidade também está incluída nas versões mais recentes do BPK.

1.2.7 Rootkits

Um *rootkit* pode ser definido como um programa - ou um conjunto de programas - usado por um atacante para que o mesmo consiga ocultar sua presença em um determinado sistema e, ainda, para permitir acesso futuro a esse sistema (ex: por meio da instalação de *backdoors*) [Klaus and Nelson 2001]. Percebe-se que o termo *rootkit* refere-se a um conjunto de ferramentas utilizadas pelo atacante não para obter privilégios de super-usuário (como no caso do uso de um *exploit* que dá acesso *root* a um sistema), mas sim, para manter esses privilégios ocultos em seus acessos futuros [Microsoft 2007b]. Os *rootkits* podem ser classificas em duas categorias [Klaus and Nelson 2001]: os *rootkits* tradicionais e os *rootkits* baseados em LKMs (*Loadable Kernel Modules*).

Os *rootkits* tradicionais começaram a ser desenvolvidos em meados de 1994 e são caracterizados por versões modificadas de comandos do sistema, como *ls* (usado para listar arquivos), *ps* (usado para listar processos), *ifconfig* (usado para configurar dispositivos de rede) e *netstat* (usado para exibir conexões de rede). Esses comandos passaram a ser programados para ocultarem do administrador do sistema os processos, os arquivos e as conexões utilizadas pelo atacante. No caso do *ifconfig*, por exemplo, o programa original é modificado e substituído por uma versão maliciosa que oculta o fato de uma determinada interface de rede estar sendo executada em modo promíscuo[2], dando a ilusão ao administrador da máquina de que tudo está ocorrendo normalmente em seu sistema.

Os *rootkits* dessa geração podem ser neutralizados facilmente. Para isso, é preciso fazer uso de programas específicos para monitorar o sistema original e armazenar as informações obtidas (como tamanho do arquivo e data de criação) em bases de dados. Após, caso algum arquivo seja alterado, o programa identifica essa alteração e aponta a possibilidade do sistema estar comprometido.

Os *rootkits* baseados em LKM, por sua vez, começaram a ser publicados a partir de 1997. Esses códigos maliciosos funcionam alterando as chamadas do sistema (*system calls*). Os módulos do kernel são componentes que podem ser carregados de forma dinâmica, modificando a funcionalidade de um sistema mesmo sem a necessidade de uma reinicialização. Esse tipo de *rootkit* normalmente altera também as chamadas do sistema que permitem listar os módulos de kernel instalados. O processo de detecção desses *malwares* é muito mais difícil comparado ao processo de detecção dos *rootkits*

[2] Uma interface de rede sendo executada em modo promíscuo passa a aceitar pacotes de maneira passiva, mesmo que esses pacotes não sejam endereçadas para essa interface.

tradicionais, pois os comandos do sistema continuam inalterados e o próprio kernel responderá às requisições.

1.2.8 Bots

Segundo [Holz 2005] há três atributos que caracterizam um *bot* (nome derivado de {Robot}): (a) a existência de um controle remoto, (b) a implementação de vários comando e (c) um mecanismo de espalhamento, que permite ao *bot* espalhar-se ainda mais. De acordo com [Ramachandran and Feamster 2006], acredita-se que a maior parte dos *spams* é enviado por *botnes*, ora partindo diretamente destes, ora os mesmos sendo utilizados como *relay*.

A família de *bots* mais conhecida é provavelmente a família *Agobot* (também conhecida como Gaobot). O código foi escrito em *C++* com suporte multi-plataforma. Segundo a Sophos [Sophos 2007] há mais de mil variantes do Agobot conhecidas. Ao ser iniciado, o *bot* tenta conectar-se com alguns endereços previamente conhecidos e realizar um teste de velocidade, o que torna fácil a contabilização do número de infecções [Holz 2005].

Tipicamente um *bot* conecta-se a uma rede IRC (*Internet Relay Chat*) e fica esperando por comandos em um canal específico. Ao identificar seu mestre, o *bot* irá realizar o que lhe for ordenado através de comandos. Um conjunto de *bots* é chamado de uma *botnet*, *bot-network* ou mesmo *zombie drones*. Em [Mclaughlin 2004], os autores apontam um estudo que estimava que, em 2004, o Phatbot possuía uma rede aproximadamente 400.000 *bots*.

1.3 Forense Computacional

Nessa seção será apresentado uma breve resumo sobre a história da ciência forense, em seguida serão descritas as etapas do processo de investigação e os desafios inerentes a realização das técnicas da Forense Computacional em ambientes de produção ou que não podem ser desconectados (*live systems*).

1.3.1 Uma Breve Incursão pela História da Ciência Forense

A Forense Computacional tem como objetivo, a partir de métodos científicos e sistemáticos, reconstruir as ações executadas nos diversos ativos de tecnologia utilizados em cyber crimes. Embora as aplicações de tais métodos no contexto que envolve a tecnologia seja algo recente, o mesmo não se pode afirmar da ciência forense como um todo, pois ao longo da história podem ser observados muitos casos de aplicação de métodos científicos para fins de comprovação de fraudes e reconstrução de eventos.

Um dos primeiros casos de descoberta de fraudes a partir de experimentos científicos é relatado pelo historiador romano Virtrúvio, segundo o qual Arquimedes foi chamado pelo rei Hieron para atestar que a coroa encomendada junto a um artesão local não era composta pela quantidade de ouro combinada previamente entre as partes. Embora existisse essa suspeita, o rei Hieron não tinha evidências que lhe permitissem acusar o fraudador e, portanto, atribuiu a tarefa de investigação sobre o caso a Arquimedes que, depois de algum tempo e quase que por acaso, formulou a teoria do

peso específico dos corpos [Inman and Rudin 2000]. A partir dessa nova descoberta Arquimedes comprovou que parte da estrutura da coroa havia sido composta de prata e, portanto, não se tratava de uma peça totalmente de ouro.

Outro fato que demonstra que há muito tempo a sociedade faz uso da forense para fins da lei e para atribuir responsabilidades a determinados indivíduos, data do século VII. Nessa época, já eram utilizadas impressões digitais para determinar as identidades dos devedores. As impressões digitais dos cidadãos eram anexadas às contas que ficavam em poder dos credores. Essas contas eram legalmente reconhecidas como prova válida do débito. Essa mesma técnica também era empregada pelos chineses para identificar a autoria de documentos e de obras de arte. Em 1823, John Evangelist Purkinji, um professor de anatomia da Universidade de Breslau, Czecheslovakia, publicou o primeiro artigo sobre a impressão digital e sugeriu um sistema de classificação baseado em padrões [Inman and Rudin 2000].

Já no século XX a evolução da ciência forense pode ser observada a partir de pesquisas que conduziram, por exemplo, à identificação do tipo sanguíneo e a análise e interpretação do DNA. Durante este período foram publicados os principais estudos referentes a aplicação de métodos e técnicas utilizadas na investigação de crimes e, também, foi criado *The Federal Bureau of Investigation* (FBI) – uma referência no que tange a investigação de crimes e a utilização de técnicas forense em diversas áreas [Inman and Rudin 2000].

Atualmente, existem peritos especializados em diversas áreas ou disciplinas como por exemplo: análise de documentos, antropologia, balística, criminalística, genética, odontologia, patologia, psiquiatria, química e toxicologia. De maneira formal, afirma-se que a Forense Computacional é uma sub-área da Forense Digital voltada a análise de evidências em computadores isolados e também em computadores em rede, embora ambas possam ser definidas como a ciência que estuda a aquisição, a preservação, a recuperação e a análise de dados que estão em formato eletrônico, a Forense Digital possui um escopo mais abrangente pois engloba evidências armazenadas e processadas por qualquer tipo de dispositivo eletrônico, por exemplo celulares, máquinas fotográficas digitais e computadores [Kruse and Heiser 2001]. Na subseção seguinte serão apresentadas as etapas do processo de investigação.

1.3.2 O Processo de Investigação Forense

Conforme mencionado na subseção 1.1, a Forense computacional é empregada em diversos cenários tanto para fins legais (por exemplo: investigar casos de espionagem industrial) quanto para o exercício de ações disciplinares internas (por exemplo: uso indevido de recursos da instituição) - em ambos os casos o intuito é obter evidências relacionadas à realização desses eventos.

As evidências são peças utilizadas por advogados nos tribunais e cortes do mundo inteiro, mas para que sejam consideradas provas válidas é muito importante que o perito realize o processo de investigação de maneira cuidadosa e sistemática, para que entre outras coisas todas as evidências sejam preservadas e detalhadamente documentadas. De acordo com [Kent et al. 2006] e [Kruse and Heiser 2001] as fases de um processo de investigação são:

- Coleta dos dados: nessa fase os dados relacionados a um evento devem ser coletados e a integridade dos mesmos deve ser preservada, posteriormente, os equipamentos devem ser identificados, devidamente embalados, etiquetados e suas identificações registradas;
- Exame dos dados: nessa segunda fase são selecionadas e utilizadas ferramentas e técnicas apropriadas a cada tipo de dado coletado, a fim de identificar e extrair as informações relevantes ao caso que está sendo investigado, mas sempre com a preocupação de manter a integridade dos dados;
- Análise das informações: a terceira etapa refere-se à análise dos dados filtrados na etapa anterior, cujo objetivo é obter informações úteis e relevantes que possam responder às perguntas que deram origem à investigação;
- Interpretação dos resultados: na última fase do processo de investigação gera-se um relatório no qual deve estar descrito os procedimentos realizados e os resultados obtidos.

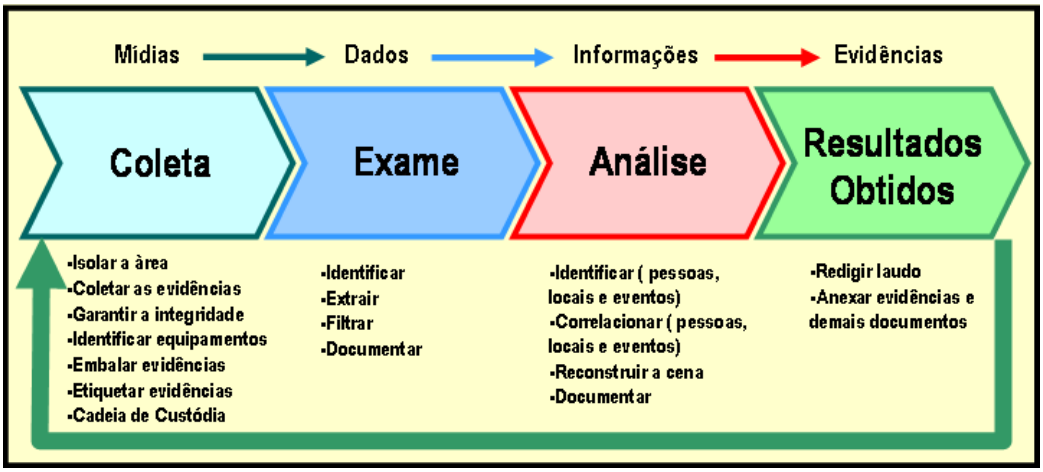


Figura 6. Fases do processo de investigação

A Figura 6 ilustra as quatro fases mencionadas acima, a seqüência em que devem ser realizadas e os procedimentos relacionados a cada uma das etapas. Além disso, é possível se observar as transformações ocorridas durante o processo forense. Por exemplo, a investigação começa a partir da apreensão dos dispositivos (computadores, meios de armazenamento e outras fontes de informações), em seguida os dados armazenados são coletados e passam pela fase de exame, essa é a primeira transformação pois a partir desse momento o perito utiliza ferramentas que lhe permitam separar apenas os dados relevantes ao caso investigado. As outras transformações que se observam estão relacionadas com as etapas de análise, da qual se obtêm a partir da correlação de eventos informações importantes, e de elaboração do relatório da investigação, que tem como resultado um laudo técnico no qual as evidências são claramente indicadas e descritas. A seguir as fases do processo forense serão descritas em detalhes.

Coleta dos Dados

Conforme menção anterior, a primeira etapa do processo forense é identificar possíveis fontes de dados. As fontes de dados mais comuns são computadores pessoais, laptops e dispositivos de armazenamento em rede. Esses sistemas normalmente possuem diversos tipos de conexões que possibilitam o acesso a outras fontes de informações, tais como: CD's e DVD's. Os equipamentos podem ainda possuir algumas portas de comunicação de vários tipos, como: *USB, Firewire, Flash card e PCMCIA* em que outras mídias e dispositivos externos de armazenamento de dados podem estar conectados [Kent et al. 2006].

Os dados também podem estar armazenados em locais fora de domínios físicos da cena investigada, como provedores de Internet, servidores FTP (*File Transfer Protocol*) e servidores coorporativos. Nesses casos, a coleta dos dados armazenados em um provedor de Internet, por exemplo, somente será possível mediante ordem judicial. Após a identificação das possíveis origens dos dados, o perito necessita adquiri-los. Para a aquisição dos dados, é utilizado um processo composto por três etapas:

1. o perito deve estabelecer a ordem (prioridade) na qual os dados devem ser coletados. Os fatores importantes na priorização dos dados são [Kent et al. 2006]:

   - **Volatilidade:** os dados voláteis representam informações que serão perdidas, caso o sistema seja desligado e, portanto, devem ser imediatamente coletados pelo perito. Por exemplo, o estado das conexões de rede e o conteúdo da memória.

   - **Esforço:** o esforço necessário para coletar dados de diferentes origens pode variar. O esforço envolve não somente o tempo gasto pelo perito, mas também o custo dos equipamentos e serviços de terceiros, caso sejam necessários. Por exemplo, coletar os dados de um roteador da rede local necessita menor esforço do que coletar os dados de um provedor de Internet.

   - **Valor estimado:** baseado na percepção do perito sobre a situação do ambiente e nas experiências anteriores semelhantes de investigação, ele deve estimar um valor relativo para cada provável fonte de dados, para, assim, poder definir a seqüência na qual as fontes de dados serão investigadas.

   Utilizando esses três fatores para cada provável fonte de dados, o perito poderá definir qual será a prioridade a ser adotada para a aquisição e quais dados serão coletados. Por exemplo, em uma investigação de invasão da rede, o perito deve preocupar-se, primeiramente, com os dados voláteis, como as conexões da rede, o estado das portas TCP e UDP e quais programas estão em execução. Em seguida, devem ser coletados os dados contidos na memória, as configurações da rede, informações sobre quais programas estão em execução para, então, iniciar a coleta dos dados não-voláteis.

2. **Copiar dados:** o processo de cópia dos dados envolve a utilização de ferramentas adequadas para a duplicação dos dados, por exemplo, o utilitário dd, encontrado na maioria das distribuições Linux, que pode ser usado para coletar os dados voláteis como os contidos na memória, e duplicação das fontes de dados não-voláteis, garantindo a integridade e segurança dos dados.
3. **Garantir e preservar a integridade dos dados:** após a coleta dos dados, o perito deve garantir e preservar a integridade dos mesmos, pois, caso isso não ocorra, eles poderão ser invalidados como provas perante a justiça. A garantia da integridade das evidências consiste na utilização de ferramentas que aplicam algum tipo de algoritmo *hash*. Esse procedimento deve ser executado nos dados originais e nas cópias, e as strings resultantes devem ser comparadas, para certificar-se de que são idênticas, garantindo, assim, a integridade dos dados.

A exemplo dos demais objetos apreendidos na cena do crime, os materiais de informática apreendidos deverão ter anotado em seu relatório de apreensão, conhecido como cadeia de custódia, o nome de todas as pessoas que estejam de posse dos mesmos e a situação envolvendo o referido material.

Exame dos dados

O exame dos dados tem a finalidade de avaliar e extrair somente as informações relevantes à investigação, o que representa uma tarefa muito trabalhosa visto a grande capacidade de armazenamento dos dispositivos atuais e a quantidade de diferentes formatos de arquivos existentes, entre eles: imagens, áudio, arquivos criptografados e compactados.

Em meio aos dados recuperados podem estar informações irrelevantes e que devem ser filtradas. Por exemplo, o arquivo de log do sistema de um servidor pode conter milhares de entradas, porém somente algumas delas podem interessar à investigação. Além disso, são muitos os formatos de arquivos que possibilitam o uso de esteganografia para ocultar dados, o que exige que o perito esteja atento e apto a identificar e recuperar esses dados.

A correta aplicação das diversas ferramentas e técnicas disponíveis, atualmente, pode reduzir muito a quantidade de dados que necessitam de um exame minucioso. A utilização de determinados filtros como palavras-chave ou tipos de arquivos nas pesquisas podem agilizar a localização das informações, tais como encontrar documentos que mencionem um determinado assunto, pessoa em particular ou ainda identificar entradas entre os registros de e-mail para um endereço específico.

Outra prática vantajosa é utilizar ferramentas e fontes de dados que possam determinar padrões para cada tipo de arquivo como texto, imagem, música, vídeos, entre outros. Por exemplo, o projeto denominado *National Software Reference Library* (NSRL), contêm uma coleção de assinaturas digitais referentes a milhares de arquivos o que pode ser usado para identificar e filtrar, por exemplo, arquivos que tenham sido manipulados por ferramentas de esteganografia [Kruse and Heiser 2001] e [Farmer and

Venema 2006]. As técnicas e ferramentas que podem ser utilizadas nesta fase da investigação serão descritas na próxima seção.

Análise das informações

Uma vez que as informações relevantes foram extraídas dos dados coletados, o perito deve concentrar suas habilidades e conhecimentos na etapa de análise e interpretação das informações. A etapa de análise tem a finalidade de identificar pessoas, locais e eventos, determinando como esses elementos estão inter-relacionados, pois, dessa maneira, será possível realizar uma descrição precisa e conclusiva da investigação [Kent et al. 2006] e [Farmer and Venema 2006].

Normalmente, nessa etapa, é necessário correlacionar informações de várias fontes de dados. Por exemplo, alguém tenta realizar um acesso não autorizado a um determinado servidor, através da análise dos eventos registrados nos arquivos de log do sistema, é possível identificar o endereço IP, utilizado pelo equipamento de onde a tentativa de acesso não autorizado. Além disso, os registros gerados pelos *firewalls*, sistemas de detecção de intrusão (tanto de rede quanto de host) e demais aplicações são extremamente importantes nesta etapa do processo.

Essa é uma fase que além de consumir muito tempo, esta muito suscetível a equívocos pois depende muito da experiência e do conhecimento dos peritos, já que são poucas as ferramentas que realizam esse tipo de análise com precisão [Casey 2006].

Interpretação dos Resultados

A interpretação dos resultados obtidos é a etapa conclusiva da investigação onde o perito constrói um laudo pericial que deve ser escrito de forma clara e concisa, elencando todas as evidências localizadas e analisadas, com base em todas as etapas anteriores da investigação.

O laudo pericial deve apresentar uma conclusão imparcial e final a respeito da investigação. Para que o laudo pericial se torne um documento de fácil interpretação por qualquer pessoa, seja ela do meio jurídico ou técnico, é indicado que o mesmo seja organizado em seções como: finalidade da investigação, autor do laudo, resumo do incidente, relação de evidências analisadas e seus detalhes, conclusão, anexos e glossário [Kent et al. 2006].

Nesse documento deve constar informações sobre a metodologia utilizada durante a realização do processo, as técnicas, os *softwares* e os equipamentos empregados, isso para que se necessário as fases da investigação possam ser reproduzidas. Na subseção 1.4.1 serão apresentadas mais informações sobre a elaboração do relatório final de uma investigação. A seção seguinte descreve as vantagens existentes e os cuidados que devem ser tomados durante a execução da investigação em dispositivos conectados a redes coorporativas e à Internet.

1.3.3 *Live Forensics*: Diagnóstico de sistemas *on-line*

Em muitos casos os profissionais de forense computacional estão mediante uma difícil tomada de decisão, desligar os equipamentos ou mantê-los operando a fim de executar

os procedimentos de uma investigação. Por exemplo, o sistema de detecção de intrusão gera alertas que indicam que o servidor web de uma organização esta sob um determinado ataque, o que pode ser um falso positivo, nesse momento a equipe de resposta a incidentes é acionada e tem que decidir entre a parada do servidor, o que pode representar a perda de dinheiro para a instituição, mas garante o tempo e as condições necessárias para que os peritos realizem as suas atividades, ou mantê-lo *on-line* o que permite ao investigador coletar dados voláteis - que são de grande importância para o entendimento e reconstrução dos eventos realizados - mas mediante qualquer descuido existe a possibilidade de haver a contaminação das evidências.

De acordo com [Carrier 2006] e [Adelstein 2006] o processo de investigação forense envolve basicamente dois tipos de técnicas: *post-mortem* e *live analysis*. A abordagem tradicional da Forense Computacional (post-mortem) tem como premissa a preservação de todas as evidências armazenadas nos discos rígidos e outras mídias, enquanto que a abordagem denominada de *live computer forensics* tem como objetivo obter o máximo de informações relacionadas ao contexto (por exemplo: estado das conexões, conteúdo da memória e dados referentes aos processos em execução), algo como uma fotografia da cena do crime. Essas técnicas quando realizadas de forma correta, claramente, se complementam e contribuem para que o resultado da investigação seja conclusivo e preciso.

Quando o processo forense é realizado nas mídias apreendidas (tais como discos rígidos, CDs e DVDs) o desafio, conforme mencionado anteriormente, é localizar entre um grande volume de dados, aqueles que são pertinentes ao caso investigado. Nesse tipo de cenário, a única e principal fonte de informação é o conteúdo gravado nos meios de armazenamento não voláteis. Esses dados podem ser obtidos a partir de ferramentas desenvolvidas ou instaladas e compiladas pela própria equipe de investigação e, portanto a priori confiáveis. Entretanto, quando se trata de cenários em que os dispositivos não foram desligados é importante que o perito certifique-se que as ferramentas utilizadas para executar os procedimentos de coleta, exame e análise dos dados não foram comprometidas a fim de gerar dados falsos ou omitir informações, como por exemplo ocultar processos em execução no sistema operacional ou não mostrar determinadas entradas do arquivo de log do servidor [Skoudis and Zeltser 2003].

Conforme [Skoudis and Zeltser 2003] e [Carrier 2006] os *rootkits* são, entre todos os códigos maliciosos, os principais causadores ou fontes de dados falsos, pois podem modificar as ferramentas (comandos) do sistema operacional e assim permanecerem com acesso ao host e não serem identificados. Os *rootkits* inserem filtros em determinadas partes do sistema operacional que impedem a visualização de algumas informações. Por exemplo, a Figura 2 mostra um filtro que impede que o arquivo passwds.txt, mesmo existindo no sistema de arquivos, seja exibido com a saída de um comando para listar o conteúdo de diretórios.

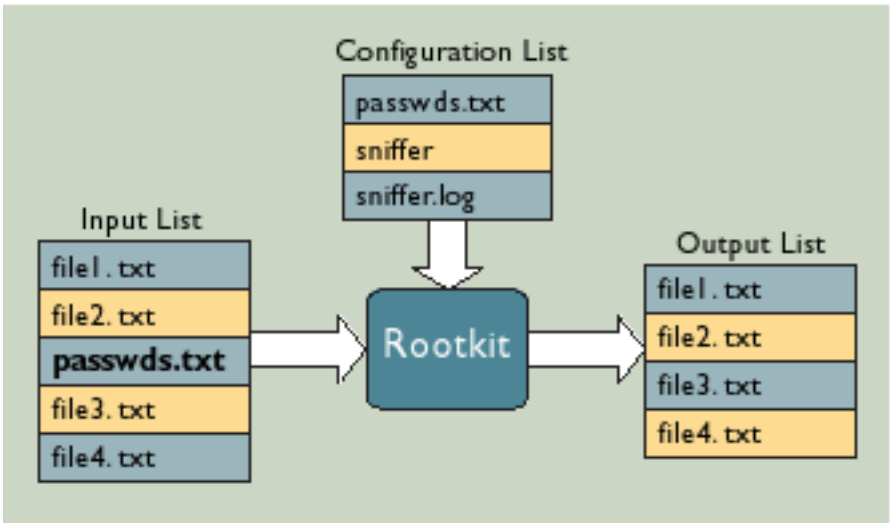


Figura 7. Exemplo de um filtro utilizado por *rootkits* [Carrier 2006]

Os *rootkits*, conforme visto na subseção 1.2.7, podem realizar modificações em diversas camadas do sistema operacional, ou seja, (a) podem atuar diretamente na aplicação, (b) redirecionar as chamadas de sistema, (c) substituir as bibliotecas compartilhadas e (d) subverter o kernel do sistema operacional [Hoglund and Butler 2005].

A fim de mitigar os riscos mencionados, sugere-se que o perito utilize um *live CD* com um conjunto de ferramentas apropriadas e que sejam confiáveis, pois isso servirá como contramedida para *rootkits* que atuam no nível da aplicação e de bibliotecas. Atualmente, existem algumas distribuições Linux voltadas a Forense Computacional que podem auxiliar no processo de investigação forense, entre elas sugere-se a utilização o projeto denominado FDTK [Neukamp 2007]. Contornar os problemas com *rootkits* que atuam no nível do kernel é um pouco mais complicado porque não há como acessar a memória ou o hardware sem passar pelo kernel. Neste caso a principal recomendação ainda é utilizar os detectores de *rootkits*, mas esse tipo de ferramenta pode interferir no sistema de alguma forma [Carrier 2006] e [Adelstein 2006].

Além disso, é importante relembrar que o perito deve (a) manter uma lista contendo a *hash* de todas as evidências coletadas, para que se necessário posteriormente possa demonstrar que nada foi alterado, e (b) ter em mente que alguns dados são mais efêmeros do que outros e, portanto, a ordem de volatilidade deve ser considerada no momento da coleta dos dados [Farmer and Venema 2006].

Em suma, além dos tipos de dados que podem ser coletados a diferença mais significativa entre *live* e *post-mortem analysis* é o grau de confiança existente nos resultados obtidos. Isso porque as ferramentas e comandos instalados no sistema investigado podem ter sido modificados para produzir dados falsos e também porque qualquer erro do perito pode contaminar ou alterar os dados existentes [Carrier 2006].

1.4 Técnicas e Ferramentas Forenses

Na seção anterior, foram apresentados alguns dos conceitos básicos sobre Forense Computacional e descritas as etapas em uma investigação forense. Essa seção retomará

este assunto, mas agora sob uma perspectiva mais técnica, com o intuito de aprofundar a compreensão do leitor sobre cada uma das etapas. Ao final, esta seção apresentará ainda uma introdução ao assunto das técnicas anti-forense.

1.4.1 Técnicas Forenses

Antes de iniciar a descrição sobre as técnicas e os procedimentos a serem adotados pelo perito em cada uma das fases do processo de investigação, é importante mencionar algumas das boas práticas que antecedem a coleta dos dados. Por exemplo [Farmer and Venema 2006]:

1. esterilizar todas as mídias que serão utilizadas ou usar mídias novas a cada investigação;
2. certificar-se de que todas as ferramentas (*softwares*) que serão utilizadas estão devidamente licenciadas e prontas para utilização;
3. verificar se todos os equipamentos e materiais necessários (por exemplo, a estação forense, as mídias para coleta dos dados, etc.) estão a disposição;
4. quando chegar ao local da investigação, o perito deve providenciar para que nada seja tocado sem seu consentimento, com o objetivo de proteger e coletar todos os tipos de evidências;
5. os investigadores devem filmar ou fotografar o ambiente e registrar detalhes sobre os equipamentos como: marca, modelo, números de série, componentes internos, periféricos, etc.
6. manter a cadeia de custódia.

Uma vez tomados esses cuidados, o perito poderá dar início a coleta de dados junto aos dispositivos eletrônicos apreendidos.

Coleta dos Dados

Uma vez que os equipamentos estejam protegidos e devidamente registrados, o perito poderá dar início à coleta dos dados. A primeira ação a ser tomada é manter o estado do equipamento, ou seja, se o equipamento estiver ligado, o mesmo não deve ser desligado e, se o equipamento estiver desligado, o mesmo não deve ser ligado, pois dessa forma não haverá modificações nas evidências.

Conforme mencionado na subseção 1.3.3, o estado no qual os equipamentos se encontram é muito importante, pois determinará a prioridade durante a coleta dos dados – que são classificados em voláteis e não-voláteis. A lista abaixo apresenta um conjunto de dados voláteis organizada pela ordem recomendada para coleta, segundo [Kent et al. 2006]:

- **Conexões de rede:** os sistemas operacionais oferecem recursos que permitem visualizar informações sobre as conexões de rede atuais. Por exemplo, os endereços IP de origem e destino, o estado das conexões e o programa associado a cada uma das portas. Além disso, a lista de sistemas de arquivos montados remotamente e o estado da interface de

rede também são dados relevantes e que podem auxiliar a análise dos dados;

- **Sessões de *Login*:** dados como a lista dos usuários atualmente conectados, o horário em que a conexão foi realizada e o endereço de rede de onde partiu essas conexões, quando correlacionados com outras informações como aquelas obtidas através das conexões de redes podem auxiliar, por exemplo, na identificação (a) dos usuários, (b) das ações realizadas e (c) do horário em que essas atividades foram executadas, o que permite a reconstrução dos fatos segundo a ordem cronológica dos eventos ocorridos;
- **Conteúdo da memória:** o espaço de troca e a memória principal, normalmente, contém os dados acessados recentemente tais como: senhas e os últimos comandos executados. Além disso, como em um sistema de arquivos, a memória pode conter resíduos de dados nos espaços livres ou que não estão em utilização, por exemplo: partes ou até mesmo arquivos inteiros que foram manipulados [Farmer and Venema 2006];
- **Processos em execução:** a lista e o estado de cada um dos processos do sistema são dados importantes, pois possibilitam identificar quais os programas que estão sendo executados;
- **Arquivos abertos:** comandos como o *lsof* presente em sistemas operacionais Linux geram uma lista contendo o nome de todos os arquivos que estão abertos no momento – essa informação pode ser um indicador para o perito do que deve ser coletado e, posteriormente analisado;
- **Configuração de rede:** as configurações da rede incluem informações como o nome da máquina, o endereço *IP* e o *MAC Address* (*Media Access Control*) de cada uma das interfaces de rede;
- **Data hora do sistema operacional:** a data e hora atual do sistema e as configurações de fuso horário – esses dados são importantes para reconstruir os eventos segundo a ordem cronológica de realização dos eventos.

Ao contrário dos dados voláteis, os dados não-voláteis são menos sensíveis à manipulação e podem ser coletados após o equipamento ser desligado, pois não sofrem alterações. Para realizar a cópia dos dados existem pelo menos dois métodos:

- **Cópia lógica (Backup):** as cópias lógicas gravam o conteúdo dos diretórios e os arquivos de um volume lógico. Não capturam outros dados que possam estar nas mídias, tais como os arquivos deletados ou fragmentos de dados armazenados nos espaços não utilizados, mas alocados por arquivos.
- **Imagem:** a imagem do disco, ou imagem *bit-a-bit* dos dados das mídias, inclui os espaços livres e os espaços não utilizados. As imagens bit-a-bit dos dados necessitam mais espaço de armazenamento e consomem

muito mais tempo para serem realizadas, porém permite ao investigador realizar as etapas de exame e análise com base em uma cenário mais próximo do real, pois possibilita por exemplo a recuperação de arquivos excluídos já que se trata de uma imagem da mídia apreendida.

A principal fonte de dados não-voláteis é o sistema de arquivos que armazena diversos tipos de dados, entre eles [Kent et al. 2006] e [Farmer and Venema 2006]:

- **Arquivos temporários:** durante a instalação e execução das aplicações são gerados arquivos temporários – que nem sempre são excluídos ao desligar os equipamentos. Esse tipo de arquivo pode conter dados relevantes como cópias de arquivos do sistema, dados sobre as aplicações e outras evidências;
- **Arquivos de Configuração:** esse tipo de arquivo fornece uma série de informações, como por exemplo: a lista dos serviços que devem ser ativados durante o processo de inicialização, a localização de arquivos de log, a relação de grupos e usuários do sistema e também os arquivos de senha e de agendamento de tarefas;
- **Arquivos de Swap:** os arquivos de swap (ou de troca) quando utilizados fornecem dados sobre aplicações, nome e senha de usuários, entre outros tipos de dados;
- **Arquivos de Dados:** são aqueles arquivos gerados por *softwares* como editores de texto, planilhas, agendas, etc.;
- **Arquivos de Hibernação:** arquivos de hibernação são criados para preservar o estado do sistema e contêm dados sobre a memória do dispositivo e os arquivos em uso – esses arquivos são utilizados para restaurar o sistema;
- **Arquivos de Log:** normalmente os sistemas operacionais registram diversos eventos relacionados ao sistema. Além disso, as aplicações também geram os seus próprios arquivos de log, nos quais são registrados dados como horário de acesso e de inicialização de serviços e transações, entre outros.

Durante a aquisição dos dados mencionados acima é muito importante manter a integridade dos atributos de tempo *mtime* (*modification time*), *atime* (*access time*) e *ctime* (*creation time*) – denominados de MAC *Times* - que estão relacionados aos arquivos e diretórios. [Farmer and Venema 2006]. Segue abaixo uma breve descrição destes atributos:

- **Modificação:** registro da data e hora em que ocorreu a última alteração no arquivo;
- **Acesso:** registro da data e hora em que ocorreu o último acesso ao arquivo;
- **Criação:** registro da data e hora em que o arquivo foi criado, entretanto, quando um arquivo é copiado de um local para outro em um sistema, o registro de criação assume a data e hora do destino e as informações de modificação permanecem inalteradas.

Essas informações são úteis para identificar o que ocorreu em um incidente, mas também são muito suscetíveis a alterações. Por exemplo, o simples acesso a um diretório altera o atributo atime e pode induzir a equívocos durante as fases seguintes.

1.5 Exame dos Dados

Após a restauração da cópia dos dados, o perito inicia o exame dos dados coletados e faz uma avaliação dos dados encontrados, incluindo os arquivos que haviam sido removidos e foram recuperados, arquivos ocultos e fragmentos de arquivos encontrados nas áreas livres ou nas áreas não utilizadas das mídias. Esse exame minucioso dos dados coletados tem como finalidade localizar, filtrar e extrair somente as informações que possam de alguma maneira, contribuir para a reconstrução dos eventos que deram origem à investigação. A seguir, serão descritos as técnicas envolvidas nesta fase do processo.

Extração dos dados

A extração manual dos dados é um processo difícil e demorado, pois exige do perito conhecimento aprofundado, principalmente, sobre o sistema de arquivos. Entretanto, existem algumas ferramentas disponíveis que podem automatizar o processo de extração dos dados, bem como na recuperação dos arquivos deletados.

Localização de arquivos

A tarefa de localização e identificação do conteúdo dos diversos tipos de arquivos, com os quais o perito irá se deparar durante a investigação, pode ser facilitada se o mesmo possuir um bom conhecimento dos diversos formatos de arquivos existentes, por exemplo, uma extensão JPG identifica um arquivo gráfico, uma extensão mp3 identifica um arquivo de áudio. Mas, os usuários podem alterar a extensão de qualquer tipo de arquivo, por exemplo, renomear um arquivo de texto para a extensão mp3. Além disso, esses arquivos podem armazenar outros dados, vide aplicação de técnicas de esteganografia em áudio, vídeo e arquivos de imagem.

Os dados armazenados nos arquivos podem ser identificados com maior precisão, utilizando ferramentas de análise de cabeçalhos. O cabeçalho de um arquivo contém assinaturas particulares que possibilitam identificar qual o tipo de dado que o arquivo contém, podendo também indicar se ele foi cifrado. Uma prática comum utilizada pelos atacantes é renomear a extensão dos arquivos, entretanto comandos como o file permite identificar o tipo de arquivos independentemente do tipo de tipo de extensão.

A criptografia está freqüentemente presente entre os desafios enfrentados pelos peritos. Os usuários podem cifrar arquivos, pastas, volumes ou partições para que outras pessoas não possam acessar o seu conteúdo sem conhecer a chave ou a senha. Em alguns casos, não é possível decifrar esses arquivos, pois, mesmo com a ajuda de ferramentas como *John the Ripper*, esta tarefa pode exigir um tempo excessivo para a descoberta da senha [Farmer and Venema 2006].

Já para identificar e localizar arquivos que tenham sido submetidos à esteganografia, normalmente, a procura se dá nos registros dos metadados, através de

histogramas. Outra evidência é a presença de programas de esteganografia armazenados no equipamento [Kent et al. 2006]. Uma vez determinada a presença de arquivos que tenham sido submetidos à esteganografia, é importante empregar técnicas de esteganoanálise a fim de recuperar os dados ocultados.

Análise dos Dados

A etapa de análise das informações, muitas vezes, ocorre paralelo à etapa de exame, pois, conforme as evidências vão sendo identificadas e extraídas dos dados, o perito tem condições de efetuar um cruzamento e correlacionamento entre as mesmas, a fim de estabelecer e recriar o(s) evento(s) que estão sendo investigado(s). A correlação das evidências tem o propósito de responder às perguntas-chave que normalmente dão origem a uma investigação: quando e como um fato ocorreu e quem é o responsável pelos mesmo.

A escolha das ferramentas a serem utilizadas nesta fase depende de cada caso. Por exemplo, para investigar ataques ou tentativas de invasão em sistemas informatizados, serão necessárias ferramentas que auxiliem na identificação da origem do ataque. Uma vez determinada a origem dos ataques, através do endereço IP utilizado no ataque por exemplo, é necessária a identificação do responsável. Esta última pode exigir a utilização de outras ferramentas. No caso do responsável pelo endereço do atacante ser um ISP (*Internet Service Provider*), será necessária a solicitação de um mandado judicial, solicitando ao ISP informações a respeito do seu cliente que utilizava o endereço IP identificado no início da investigação.

Todas as etapas e conclusões sobre as análises realizadas devem ser devidamente registradas e ao final anexadas ao laudo pericial.

Interpretação dos Dados

Durante as etapas iniciais de uma investigação, são gerados documentos específicos referentes às atividades realizadas em cada fase. Ao longo dessa documentação, será necessário identificar somente as informações que sejam especificamente relevantes à investigação e organizá-las em categorias. A seguir, serão descritos alguns procedimentos que podem ser benéficos à organização da documentação necessária para a confecção do laudo pericial [Kent et al. 2006].

- Reunir todas as documentações e anotações geradas nas etapas de coleta, exame e análise dos dados, incluindo as conclusões prévias já alcançadas;
- Identificar os fatos que fornecerão suporte às conclusões descritas no laudo pericial;
- Criar uma lista de todas as evidências analisadas, para que as mesmas sejam enumeradas no laudo pericial;
- Listar as conclusões que devem ser relatadas no laudo pericial;
- Organizar e classificar as informações recolhidas para garantir a redação de um laudo conciso e inquestionável.

Redação do Laudo

Posteriormente à organização devida de todas as informações, inicia-se a redação do laudo pericial. É imprescindível que resultado da investigação seja registrado de forma clara e concisa, evitando a utilização de termos técnicos complexos ou expressões somente conhecidas por pessoas ligadas à tecnologia. A seguir, são referidas algumas das seções e informações que podem auxiliar na redação do laudo pericial [Kent et al. 2006].

- Finalidade do relatório: Explicar claramente os objetivos do laudo;
- Autor do relatório: listar todos os autores e co-autores do relatório, incluindo suas especialidades e responsabilidades, durante a investigação, e informações para contato;
- Resumo do incidente: síntese, explicando o incidente investigado e suas conseqüências. O resumo deve ser redigido de forma que uma pessoa não-técnica, como um juiz ou um júri, compreenda como e quais os fatos que ocorreram e estão sob investigação.
- Evidências: fornecer descrições sobre o estado das evidências: como, quando e por quem elas foram adquiridas no decorrer das investigações.
- Detalhes: fornecer uma descrição detalhada de quais evidências foram analisadas, quais os métodos utilizados e quais as conclusões alcançadas, descrevendo os procedimentos e as técnicas adotados, durante a investigação.
- Conclusões: na conclusão, os resultados da investigação devem ser somente descritos, citando especificamente as evidências que comprovem as conclusões, evitando pormenores excessivos sobre como as evidências foram obtidas, pois essas informações já foram descritas na seção detalhes. A conclusão deve ser clara e não oferecer dupla interpretação.
- Anexos: todas as documentações, referentes à investigação, devem ser anexadas, ao final do laudo, tais como: diagramas da rede, formulários descritivos dos procedimentos utilizados, formulário de cadeia de custódia e informações gerais sobre as tecnologias envolvidas na investigação, para que, em caso de necessidade, possam ser consultadas. Outro detalhe significativo é referente aos anexos. Eles devem fornecer todas as informações complementares ao laudo, para que o leitor compreenda completamente o incidente investigado.

Outro aspecto relevante à redação do laudo refere-se ao glossário. O perito, sempre que possível, precisa adicionar um glossário dos termos utilizados no laudo, que poderá esclarecer muitas dúvidas que possam surgir durante a leitura do juiz e/ou dos jurados.

Concluídas todas as etapas de uma investigação, vale lembrar que, durante o decorrer do processo, o perito manterá contato com informações que podem ser sigilosas (por exemplo: segredos industriais ou de justiça). Sendo assim, é necessário que o perito entenda a importância e as suas responsabilidades no que se refere a preservação dos dados.

1.5.1 Ferramentas

Na etapa de coleta dos dados serão abordadas ferramentas utilizadas a fim de salvaguardar os dados contidos no equipamento suspeito, para posterior análise.

Entre as ferramentas mais conhecidas para coleta de dados estão o dd [OpenGroup 2007] (*Disk Definition*) e o dcfldd [DCFL 2007] (*Department of Defense Computer Forensics Lab Disk Definition*). O segundo é uma versão aprimorada do primeiro, criado pelo Laboratório Forense do Departamento de Defesa Americano, na qual foram adicionadas funcionalidades como a geração do *hash* dos dados durante a cópia dos mesmos, visualização do processo de geração da imagem e divisão de uma imagem em partes, a fim de facilitar o armazenamento e o transporte desta.

Visando facilitar a utilização destas duas ferramentas, um *frontend*, chamado *Automated Image & Restore* (AIR) [Gibson 2007], foi desenvolvido. O AIR é uma interface gráfica para os comandos dd/dcfldd que auxilia na criação ou restauração de imagens dos dados (evidências), tanto das mídias conectadas fisicamente ao equipamento, quanto imagens geradas através de uma rede. Além da criação de imagens, o AIR gera e compara automaticamente *hashes* MD5 ou SHA e produz um relatório contendo todos os comandos utilizados durante a sua execução. Uma das grandes funções deste utilitário é eliminar o risco da utilização de parâmetros errados por usuários menos capacitados. Entretanto, a utilização do AIR não elimina a necessidade do perito conhecer basicamente como os utilitários dd ou dcfldd funcionam. A Figura 8 ilustra a tela principal da ferramenta AIR.

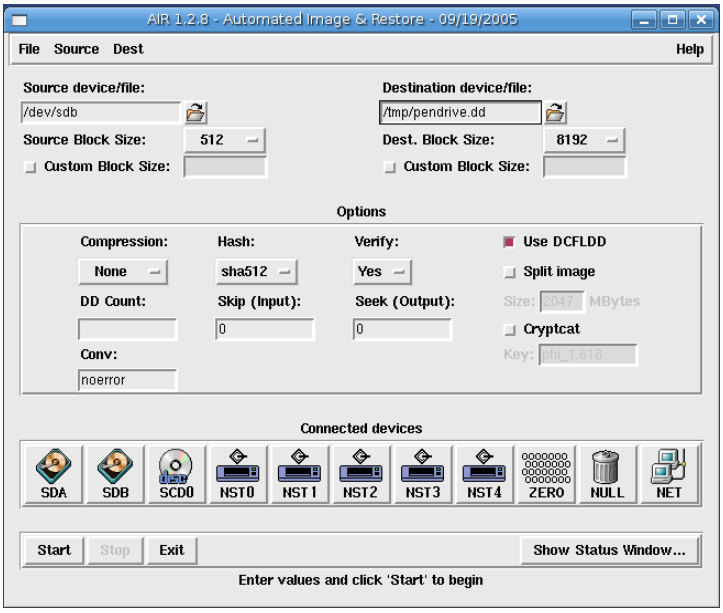


Figura 8. Imagem do utilitário AIR

Ainda dentro da etapa de coleta dos dados existe uma ferramenta, chamada *aimage*, que faz parte da biblioteca de ferramentas AFFLIB [Garfinkel 2007] (*Advanced Forensic Format Library*) e que oferece uma série de benefícios ao perito. Dentre estes benefícios estão: (a) a facilidade de utilização, (b) a automação na geração de hashes dos dados, (c) a redução de 30 a 50% no tamanho dos arquivos de imagem gerados

(através de compactação LZMA (*Lempel-Ziv-Markov Chain-Algorithm*) [Pavlov 2007] e (d) a possibilidade de extração de informações contidas nas imagens sem a necessidade de descompactá-las.

Exame dos Dados

A etapa de exame dos dados pode se tornar muito cansativa para o perito, caso o mesmo não utilize um conjunto de ferramentas adequadas que possibilite a filtragem e o foco de suas habilidades nos dados mais importantes da investigação. Diante deste cenário, o *National Institute of Standards and Technology* (NIST) [NIST 2007] mantém um projeto denominado *National Software Reference Library* (NSRL) [NSRL 2007], o qual disponibiliza uma coleção de assinaturas digitais (*hashes*) de aplicações e arquivos conhecidos. Disponíveis no formato ISO, estas coleções de assinaturas permitem que o perito elimine um conjunto de arquivos coletados e que não sofreram modificações, reduzindo assim a superfície de análise. Por exemplo, em uma investigação suponha que seja gerada a imagem de um disco com 240GB de dados. Dentre os arquivos existentes provavelmente vários deles serão parte dos arquivos do sistema operacional sendo utilizado, além de um conjunto de arquivos pertencentes aos programas instalados no equipamento. A utilização destas bases permite que o perito elimine os arquivos cujos *hashes* casem com os presentes na base de dados.

Atualmente, diversas ferramentas disponíveis, tanto proprietárias, quanto baseadas em código aberto, já permitem a utilização dos bancos de dados supracitados. Entre elas, pode-se citar o *Encase* [Guidance 2007], a *Autopsy* [Carrier 2007a , Carrier 2007c] e o *pyFLAG* [Collett and Cohen 2007]. A Figura 9 ilustra um exemplo da tela principal da ferramenta Autopsy. Como se pode notar, a interação dá-se através de uma interface web, o que dispensa a necessidade de se instalar um software específico para esta finalidade.

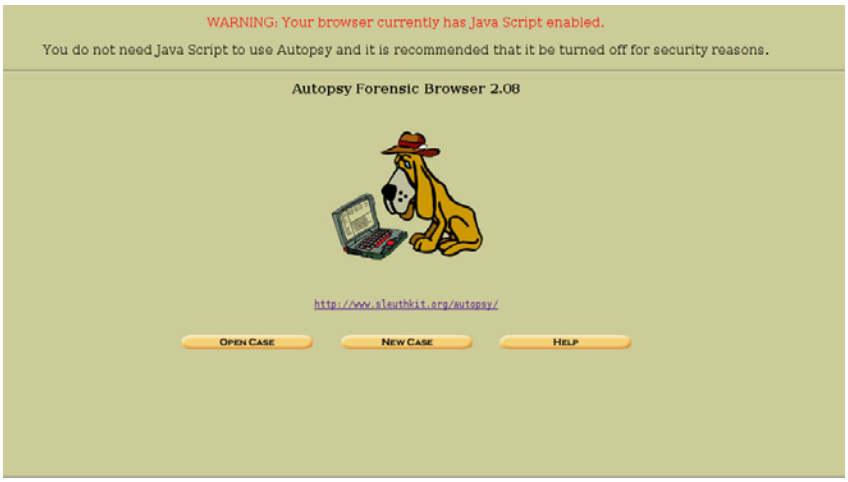


Figura 9. Imagem do utilitário *Autopsy*

Análise dos Dados

Dentre as ferramentas utilizadas na etapa de análise dos dados, é importante ressaltar os utilitários para construção da linha de tempo dos eventos. Nesta categoria, uma

ferramenta particularmente interessante é o *mactime* [Carrier 2007b], que permite que a partir das informações contidas nos metadados dos arquivos e diretórios, os mesmos possam ser classificados de acordo com a sua data de criação ou modificação, fornecendo assim uma visão cronológica dos acontecimentos.

Muitos arquivos importantes que fazem parte dos sistemas operacionais da família Windows não possuem uma clara explicação de suas estruturas dificultando assim o acesso e a compreensão do conteúdo dos mesmos. Diante desta constatação foram desenvolvidas algumas ferramentas capazes de amenizar algumas destas dificuldades e que podem ser verificadas na pagina da *Foundstone* [Foundstone 2007]. Dentre as ferramentas disponíveis neste site pode-se ressaltar: Pasco [Jones 2007b] e Galleta [Jones 2007a].

O utilitário Pasco foi criado com a finalidade de analisar os índices dos arquivos do Internet Explorer. Esta ferramenta analisa gramaticalmente as informações contidas nos arquivos index.dat, exportando os resultados em um formato de texto padrão, inteligível por humanos e que utiliza como delimitador de campos o caractere |. Dessa forma, pode-se analisar as informações do arquivo índex.dat com a ajuda de outros softwares. Após a importação destas informações o perito poderá facilmente verificar a ocorrência de situações como, por exemplo, o acesso a sites ou conteúdos proibidos, ou até mesmo a utilização dos recursos da empresa em benefício próprio durante o horário de trabalho. A Figura 10 ilustra o local onde o arquivo index.dat fica armazenado, dependendo do sistema operacional sendo utilizado.

Operating System	*File Path(s)*
Windows 95/98/Me	\Windows\Temporary Internet Files\Content.IE5\ \Windows\Cookies\ \Windows\History\History.IE5\
Windows NT	\Winnt\Profiles\<username>\Local Settings\Temporary Internet Files\Content.IE5\ \Winnt\Profiles\<username>\Cookies\ \Winnt\Profiles\<username>\Local Settings\History\History.IE5\
Windows 2K/XP	\Documents and Settings\<username>\Local Settings\Temporary Internet Files\Content.IE5\ \Documents and Settings\<username>\Cookies\ \Document and Settings\<username>\Local Settings\History\History.IE5\

Figura 10. Localização do arquivo *index.dat* do Internet Explorer

Em algumas situações o perito necessita reconstruir a seqüência das ações realizadas via web por um suspeito, na qual os arquivos *cookies* do Internet Explorer poderão fornecer valiosas informações. Neste caso, a ferramenta Galleta [Jones 2007a] é capaz de analisar os *cookies* existente em uma máquina e separar as informações úteis em campos para que possam ser manipuladas por outros programas. A Figura 11mostra onde os *cookies* são armazenados, dependendo da versão do Windows sendo utilizada.

Operating System	*Cookie File Location*
Windows 2000/XP	C:\Documents and Settings\<*username*>\Cookies
Windows 95/98/ME	C:\Windows\Cookies

Figura 11. Localização dos *Cookies*

1.5.2 Métodos e Ferramentas Anti-Forense

Os métodos anti-forenses tem como objetivo deliberadamente destruir, ocultar ou modificar as evidências existentes em um sistema a fim de dificultar o trabalho realizado pelos investigadores [Harris 2006]. A seguir uma breve descrição destes métodos.

Destruição dos Dados

Para impedir, ou pelo menos dificultar, a recuperação dos dados os atacantes utilizam ferramentas (conhecidas como *wiping tools*) para remoção dos dados, tais como: *wipe*, *secure-delete*, *pgp wipe* e *The Defiler's Toolkit*. Essa categoria de ferramentas emprega uma variedade de técnicas para sobrescrever o conteúdo dos arquivos, por exemplo, gravar dados de forma randômica e sobrepor o conteúdo dos arquivos com bytes nulos. Essas ferramentas também alteram o *inode* dos arquivos o que torna a tarefa de recuperação dos arquivos ainda muito complexa, embora seja possível [Harris 2006]. Além da destruição lógica, o atacante, em alguns casos, pode danificar fisicamente as mídias utilizadas – o que dificulta e muitas vezes impossibilita a recuperação dos dados.

Ocultação dos Dados

Os dados de um arquivo podem ser escondidos pelo menos de duas formas: (a) fragmentando um arquivo e armazenando esses fragmentos em espaços não alocados ou naqueles marcados como *bad blocks* e (b) utilizando recursos como *Alternate Data Stream* (ADS), existente em sistemas de arquivos NTFS, que possibilitam esconder arquivos que não serão visualizados por comandos de listagem de conteúdo de diretórios dentro de outros arquivos, por exemplo executáveis [Zadjmool 2004].

Além desses métodos, a aplicação de criptografia e esteganografia em arquivos de texto, imagem, vídeo e áudio representam uma barreira difícil de ser superada, pois exigem tempo e recursos nem sempre disponíveis para identificação dos dados ocultos. Por exemplo, utilizar ferramentas de esteganoanálise em uma mídia de 80GB requer muito tempo e na prática nem sempre é algo viável de se realizar. O mesmo ocorre quando se trata de arquivos criptografados [Harris 2006].

Os *rootkits*, conforme já mencionados neste capítulo, implementam métodos eficientes para ocultar informações como arquivos e dados sobre os processos em execução no sistema operacional.

Modificação dos Dados

Os métodos mais comuns para realizar a modificação dos dados são: alterar a extensão e o conteúdo do cabeçalho dos arquivos [Harris 2006]. A troca da extensão pode ser facilmente detectável por comandos dos sistema operacional como file, já a modificação do cabeçalho altera a assinatura do arquivo e, portanto, impede que as ferramentas associem o conteúdo do cabeçalho a um determinado tipo de arquivo.

Outros métodos de modificação incluem a alteração dos atributos de tempo, através de ferramentas como *touch* e *timestamp*, e ataques de colisão em *hash* do tipo MD5, que podem ser utilizados por atacantes para criar arquivos com valores de *hash* idênticos, o que permite substituir arquivos legítimos por arquivos com códigos maliciosos, entre outras ameaças [Wang and Yu 2005].

Os métodos anti-forense mencionados nesta seção são implementados em diversas ferramentas tais como *Metasploit Anti-Forensic Investigation Arsenal* (MAFIA) e *Windows Memory Forensic Toolkit* [Harris 2006].

1.6 Estudos de Caso

Com base em investigações realizadas pelos autores, quatro estudos de caso serão apresentados nesta seção. Estes casos são verídicos, entretanto, algumas informações são omitidas com o objetivo de preservar a identidade das partes envolvidas. No primeiro caso será mostrado um acesso remoto de uma empresa à outra, considerado pela denunciante como indevido (sem permissão). Nesse processo, serão explicadas passo a passo todas as etapas da forense computacional. No segundo caso será mostrada uma investigação de um possível roubo de informações, mais especificamente, de dados bancários (onde o uso de um *malware* pode ter sido utilizado). Já no terceiro e no quarto caso serão mostradas investigações para a descoberta da origem de e-mails. A diferença entre os dois últimos casos é que no terceiro não se tem o cabeçalho do e-mail para análise, mas há um computador suspeito. Já no quarto caso, o perito possui o computador da vítima para análise, logo pode analisar o cabeçalho do e-mail.

1.6.1 Acesso Indevido

O primeiro caso trata do acesso indevido a um sistema, envolvendo a empresa acusada, denominada “invasora”, a empresa que realizou a denúncia, denominada como “vítima” e um sistema, denominado “sistema X”. O sistema X possui informações fundamentais para determinado ramo empresarial. Assim, o acesso ao mesmo é controlado e existe a cobrança de uma mensalidade. Para evitar que mais de uma empresa utilize o sistema com o pagamento de apenas uma mensalidade, a autenticação é baseada no endereço IP.

Em determinado momento os responsáveis pela auditoria do sistema X verificaram que havia mais de uma empresa acessando o sistema, sendo que aquela que não possuía cadastro estava acessando através daquela que possuía. Ou seja, uma empresa estava conectando-se àquela que possuía acesso ao sistema X e a partir dela acessava o sistema. Neste momento, o acesso foi bloqueado, ocasionando grande prejuízo à empresa “invadida”, visto que ela possuía clientes que dependiam de informações acessadas no sistema X.

A empresa vítima verificou em seus *logs* o acesso remoto da empresa invasora, os imprimiu e levou às autoridades para investigação. Contudo, *logs* impressos ou capturados na empresa invadida não são consideradas evidências consistentes, visto que *logs* podem ser alterados facilmente com os devidos cuidados (datas de criação/alteração de arquivo), podendo burlar qualquer situação.

Diante dos fatos, as autoridades decidiram apreender os computadores da empresa invasora, o que pode ser considerado por muitos uma decisão arriscada, visto que a empresa dependia do uso dos mesmos para trabalhar e o backup do sistema e informações poderiam estar nelas. Tratava-se de dez computadores e havia certa urgência na conclusão da investigação. Então, foi decidido que cada perito realizaria a perícia em um conjunto de máquinas (em paralelo) e, toda informação considerada importante deveria ser compartilhada com todos para facilitar a correlação de eventos.

A metodologia aplicada na perícia foi a *post mortem forensic*, visto que os computadores foram enviados aos peritos. Caso algum perito fosse requisitado para comparecer no local da apreensão, a metodologia *live forensic* poderia ser aplicada, podendo inclusive detectar o flagrante de conexões efetuadas entre as empresas envolvidas.

Seguindo as etapas explicadas na seção 3, primeiramente todos os computadores foram fotografados, identificados e seus componentes foram descritos. Todas as mídias encontradas foram associadas ao computador em que se encontravam conectadas. Por exemplo, se dois discos rígidos (HDs) foram encontrados no computador identificado como “CPU01”, os HDs podem ser identificados como “HD01-CPU01” e “HD02-CPU01”. Cabe salientar que as fotografias devem ser tiradas também do interior dos computadores, identificando os componentes que o compõem (mesmo que a descrição também seja feita) e, como estava o estado das conexões (ex: se todos HDs estavam conectados, se um drive de CD-ROM ou DVD-ROM estava conectado, entre outros). As fotografias são importantes no sentido de documentar o passo a passo realizado pelo perito e ajudar a entender algumas situações. Por exemplo, pode-se encontrar conteúdo totalmente fora de contexto em um HD. Já em outro HD, no mesmo computador, pode-se encontrar conteúdo coerente com a investigação. Muitas vezes, analisando as fotografias realizadas do interior do computador é possível verificar que um HD estava desconectado, talvez por ser um HD antigo e tiver sido desativado.

Se tivesse sido solicitada a presença do perito no local da apreensão, o ambiente onde os equipamentos se encontravam instalados, os próprios computadores, dispositivos de conexão, cabos, mídias e tudo o que pudesse ajudar na montagem do cenário poderia ser fotografado. Além das fotografias, seria possível ainda a identificação dos computadores de acordo com a localização. Por exemplo, se foi encontrado na recepção (possivelmente seria o computador da secretária), na sala com identificação de “Gerência”, “Contabilidade”, em um armário de telecomunicações (possivelmente um servidor), etc.

Depois de realizada a identificação do material questionado, foi realizada a coleta do conteúdo de cada mídia. Para isto foi utilizada a ferramenta Encase, uma das mais utilizadas na área de forense computacional. Contudo, poderiam ser utilizadas ferramentas baseadas em *software* livre. Mas, como mencionado na Seção 1.4, não

existe, ou não existia no momento da perícia, uma distribuição Linux contendo um bom conjunto de ferramentas para forense computacional.

Com um disco de boot do Encase é possível realizar uma cópia da imagem (bit-a-bit) de uma mídia para outra, sem contaminar a mídia questionada. Para certificar-se de que a integridade não foi afetada, a opção de geração de *hash* foi ativada no momento da geração da cópia. O *hash* é gerado na mídia questionada e, depois de realizada a cópia, é gerada também nesta, para que seja possível o confronto dos dois códigos gerados e assim garantir que não houve escrita na mídia questionada. Há todo este cuidado para que no caso de um pedido de contra-prova, pela parte da defesa, em um julgamento, não seja constatada uma possível escrita na mídia questionada após a apreensão da mesma. Assim, a defesa não pode alegar que alguém possa ter colocado dados que incriminem o acusado durante a cadeia de custódia (após o desligamento dos computadores até a realização da perícia).

Depois de realizada a coleta dos dados, foi possível realizar o exame dos mesmos. Como o objetivo da perícia era verificar o acesso da empresa “invasora” à empresa “invadida” e os peritos possuíam em mãos impressões de *logs*, o primeiro passo foi procurar em diretórios onde geralmente existem *logs*, típico em sistemas Linux. No entanto, a maioria dos computadores possuía sistema operacional da família Windows. Em um primeiro momento, poderia-se pensar em não analisar estes computadores e partir apenas para os que possuíam sistema Linux. Entretanto, isto seria uma irresponsabilidade dos peritos (deduzir onde tem e onde não tem evidências). Os computadores com sistema Windows possuíam diversos e-mails comerciais, documentos, entre outros. Durante este exame, os peritos verificaram dentro da imagem gerada pelo Encase (semelhante ao Windows Explorer) o que poderia conter evidências, olhando rapidamente alguns arquivos. Contudo, a visualização de muitos tipos de arquivos é prejudicada dentro desta ferramenta, então todos diretórios e arquivos candidatos a conterem dados importantes para investigação foram selecionados e extraídos. Fora da imagem é possível analisar com mais detalhes os arquivos, utilizando ferramentas específicas, como editores de texto e planilhas eletrônicas, visualizadores de imagens, e-mails e históricos de acesso a páginas Web.

Com a análise dos dados extraídos, foi verificado que havia uma troca de e-mails entre as duas empresas. Ainda, foi encontrado um contrato onde constava um acordo comercial de acesso ao sistema X pelas duas. Com a análise mais aprofundada dos e-mails e documentos encontrados foi possível elaborar uma lista de palavras-chave, contendo nomes de funcionários, endereços de e-mail, nomes de empresas, entre outros. Estas chaves foram colocadas no Encase e uma busca foi realizada. Foram encontrados arquivos excluídos, trechos de texto encontrados em parte do disco não alocada pelo sistema de arquivos e trechos de texto encontrados em unidades de alocação onde os arquivos não utilizavam todo o espaço destinado a ela. Ou seja, havia e-mails e documentos excluídos, sendo que alguns puderam ser totalmente recuperados (pois não havia escrita de outros arquivos na mesma porção do disco). Já outros arquivos foram sobrescritos em partes do disco e, desta forma, apenas alguns trechos puderam ser recuperados.

Além das constatações já relacionadas, ainda foi possível verificar em alguns casos evidências de formatação de discos e redimensionamento de algumas partições. Quanto mais se encontrava evidências de comunicação entre as empresas, mais se

encontrava palavras-chave. Assim, mais buscas eram realizadas. Contudo, nenhum indício de acesso entre as empresas havia sido encontrado até então. Tudo indicava que as máquinas com sistema Windows eram apenas utilizadas por pessoas da área administrativa/comercial. Porém, estas informações foram úteis para mostrar que se houve acesso entre elas, tudo indicava que era um acesso lícito, pois havia inclusive um contrato entre elas. Continuando as buscas, acabou sendo encontrado um desentendimento (e-mails e documentos de texto) e a parceria teria sido rompida.

Continuando com a perícia em máquinas com sistema Linux, foi verificado que essas não possuíam e-mails ou dados comerciais, indicando que se tratavam de servidores (foram identificados serviços típicos de um provedor de acesso Internet - ramo da empresa acusada). Foram realizadas buscas por comandos e endereços IP que poderiam gerar os *logs* em formato impresso, conforme mencionado anteriormente. Para o desfecho do caso, finalmente foi encontrado um comando de acesso ao endereço IP da empresa vítima. No entanto, em nenhum momento foi encontrado algum indício de invasão propriamente dita, e sim, um acesso remoto com um usuário e senha legítimos.

Após montar a cronologia dos eventos, reunir todos os arquivos, trechos de texto e outras informações consideradas relevantes para a investigação, iniciou-se então a elaboração do laudo técnico. No laudo foi relatado tudo o que foi mencionado anteriormente, informando a metodologia adotada, como foi realizada a cópia dos dados com a garantia de integridade, a análise do conteúdo e a conclusão. Basicamente, foi informado no laudo que:

- havia um contrato de parceria entre as empresas envolvidas, constatado através de trocas de mensagens e documentos;
- foi constatado certo desentendimento entre as empresas e a parceria teria sido interrompida;
- foram verificadas seqüências de comandos compatíveis com a geração dos *logs* impressos pela empresa vítima, sendo as datas posteriores ao possível desentendimento relatado;
- não foi constatado nenhum tipo de ataque, entretanto, foram constatados acessos legítimos utilizando credenciais autênticas e válidas.

Para complementar, todos os arquivos e dados considerados relevantes para a elaboração do laudo foram gravados em CD e anexados ao mesmo para que a investigação e as partes envolvidas pudessem analisar e comparar com as informações que foram relatadas. Ainda, para garantir a integridade do CD foi gerado um código *hash* para cada arquivo, esses códigos foram gravados em um arquivo. Um novo *hash* foi gerado, no arquivo de *hashes*. Este “ *hash* dos *hashes*” foi adicionado ao laudo juntamente com a explicação de como comprovar a integridade dos arquivos. No CD foi gravado um *software* livre que gera códigos *hash* utilizando o algoritmo MD5 e instruções de como utilizá-lo. Este cuidado é tomado porque durante um processo judicial advogados de defesa podem solicitar uma cópia do laudo para analisar e, após um período determinado pela Justiça, o devolver. Uma cópia do laudo (impressa) pode ser facilmente realizada e entregue ao advogado, mas uma cópia do CD não é feita, sendo assim este pode ser levado. Uma cópia alterada pode ser facilmente criada contendo rótulo (etiqueta) falsificado e entregue novamente à Justiça. Se isto for

realizado, a qualquer momento o teste da integridade dos arquivos pode comprovar a fraude. Por este motivo, é fundamental ter uma cópia, para no caso de constatação de fraude, enviar um novo CD anexo.

1.6.2 Malware

O segundo caso trata de uma suspeita de fraude bancária. Um funcionário de confiança de uma empresa possuía os dados da conta bancária e senha da mesma. Após a conferência de um extrato bancário, foi constatada a transferência de uma grande quantia de dinheiro para outra conta. Como apenas este funcionário e o dono da empresa sabiam a senha da conta bancária e ambos garantiam que a transferência não tinha sido feito por eles, foi instaurado inquérito policial.

Após investigações, foi constatado que o dono da conta para a qual foi feita a transferência era uma pessoa com poucos recursos, semi-analfabeto, e que a conta havia sido aberto há poucos dias. Este foi interrogado e falou que um desconhecido lhe ofereceu uma pequena quantia em dinheiro para que ele abrisse uma conta em determinado banco e que entregasse o cartão eletrônico ao tal desconhecido. Depois de ter entregado o cartão, o cidadão interrogado afirma que nunca mais viu ou teve contato com o desconhecido. Apenas soube descrever algumas características físicas do mesmo, o que não ajudou muito para a polícia. Este é um típico caso de fraude bancária, onde um “laranja” com pouco grau de instrução é utilizado para abrir uma conta. Os fraudadores transferem então quantias monetárias para essa conta e, de posse do cartão da vítima, retiram o dinheiro.

Neste caso, a empresa vítima solicitou ao banco o ressarcimento do valor transferido, alegando que alguma fraude eletrônica deveria ter ocorrido (já que ninguém mais saberia a senha da conta além das duas pessoas mencionadas anteriormente). O banco concordou, entretanto, a fraude deveria ser provada. Por se tratar de suspeita de crime e não haver nenhum suspeito, a solução adotada foi apreender (ou neste caso, solicitar) o computador utilizado pela vítima na empresa. A suspeita era que o computador estaria infectado por algum *malware*, situação comum na época (atualmente ainda é).

Seguindo as etapas já descritas neste capítulo, primeiramente foi realizada a coleta dos dados com uma cópia bit-a-bit e garantia de integridade. Como o objetivo estava bem definido, começou-se a análise das caixas de e-mail. Após algum tempo de análise, foi encontrada uma mensagem de *phishing scam* solicitando ao usuário para clicar em um determinado link. Para verificar se o link ainda estava ativo, foi clicado no mesmo. O link já não estava mais ativo. No entanto, era possível verificar o nome do arquivo que seria baixado (exemplo: fotos.exe). Procurou-se então pelo arquivo fotos.exe na imagem (cópia bit-a-bit) e o mesmo foi encontrado. Alguns *softwares* antivírus foram executados e constatou-se que o arquivo continha um *keylogger*, o BPK, descrito na seção 2. Uma pesquisa foi realizada e foi constatado que o BPK foi desenvolvido para monitorar, entre outros, filhos e esposo(a), segundo o site do fabricante [BlazingTools 2007]. Aparentemente, seria um *software* para uso legítimo, muito utilizado para monitorar funcionários de uma empresa, encaminhando *logs* a um determinado e-mail ou por FTP, por exemplo. Para o caso de monitoramento de funcionários no Brasil, o uso deste *software* seria possível, conforme notícias do

Tribunal Superior do Trabalho [TST 2005] e o processo E-ED-RR - 613/2000-013-10-00.7 do Tribunal Superior do Trabalho [TST 2006].

Toda a explicação do parágrafo anterior serve para mostrar que mesmo um *software* desenvolvido para o uso legítimo pode ser utilizado de forma ilegítima, ou seja, sem o consentimento do usuário. Isto porque o BPK pode ser instalado de forma "camuflada", sem que se perceba. Então, muitos fraudadores utilizam-se desta característica para que usuários instalem tal ferramenta em seu computador e envie dados para algum destino. Para isto, o fraudador deve realizar algumas configurações, como por exemplo, capturar teclas digitadas e cliques do mouse sempre que o usuário entrar em determinado site, enviando os dados capturados para um determinado servidor de FTP a cada intervalo de tempo ou volume de dados.

Voltando ao caso da empresa infectada, depois de detectado o *keylogger*, este foi extraído para análise. Nas configurações dele, foram encontrados:

- armazenamento de *logs* (teclas digitadas) e telas (região de uma quantidade pixels ao redor do clique do mouse) a partir do acesso a seis sites de bancos pré-definidos;
- envio dos arquivos gerados para um servidor FTP, com usuário e senha pré-definidos.

A partir do momento que foi verificado que o computador estava realmente infectado e que a instalação dele foi realizada momentos depois do recebimento do e-mail segundo a data e hora analisados (indicando que o e-mail induziu à execução do *malware*), partiu-se para uma nova busca. O que os peritos estavam a procura neste momento era se os dados relativos à conta bancária realmente teriam sido enviados ao servidor de FTP previamente mencionado. Foi realizada então uma busca por palavras-chave sem a utilização de expressões regulares, pois se procuravam palavras específicas, incluindo usuário e endereço IP do servidor ig.com. Para a satisfação dos peritos, foram encontrados indícios na memória virtual, indicando conexões realizadas ao servidor de FTP mencionado e com o usuário configurado. Inclusive a senha pôde ser verificada, visto que não havia criptografia. Por se tratar de memória virtual, não foi possível extrair grande quantidade de informação útil, sem haver "sujeira" entre um comando e outro. Contudo, os fragmentos encontrados puderam indicar a conexão com o servidor e algumas datas e horários puderam ser coletadas em texto claro.

O servidor FTP encontrava-se em outro país e possuía a modalidade de contas gratuitas. Um teste foi realizado no site da empresa provedora do serviço e uma conta foi criada sem informação de dados pessoais validados, ou seja, foram digitados caracteres aleatórios com um e-mail gratuito e a conta foi criada sem dificuldades.

Todas as informações mostradas neste caso foram relatadas no laudo pericial, indicando inclusive o endereço IP do servidor FTP, caso a investigação tivesse algum acesso a *logs* do servidor para uma nova perícia. No laudo não foi possível concluir que os dados da conta foram enviados a algum destinatário, porque os arquivos de log gerados pelo BPK eram excluídos de tempos em tempos. Mesmo com uma busca por arquivos excluídos, não foram encontrados dados relativos à conta. No entanto, foi possível relatar que o computador estava infectado, que foi infectado logo após o

recebimento da mensagem com *phishing scam*, e que foram realizadas conexões com um servidor FTP previamente configurado no *keylogger*.

1.6.3 Ameaça por E-mail (sem cabeçalho)

O terceiro caso trata de uma ameaça realizada por e-mail a um diretor de uma empresa, com cópia a diversos membros da diretoria. O diretor imprimiu o e-mail e o entregou à polícia, realizando o boletim de ocorrência. Contudo, o cabeçalho do e-mail não foi impresso e o computador contendo o e-mail recebido não foi entregue para a perícia. O conteúdo do e-mail mencionava fatos ocorridos na empresa, citando que uma categoria de funcionários estava correndo perigo e o diretor, além de não tomar providências, ainda protegia os possíveis ameaçadores. Para não ficar vaga a explicação, um exemplo será dado a seguir, lembrando que o exemplo é fictício e serve apenas para ilustrar a situação.

A rodoviária de uma cidade sofre diversos atentados por vândalos, que realizam furtos, são violentos com os cidadãos que transitam por ela, agridem os vigias e ameaçam os mesmos. A polícia alega não ter gente suficiente para garantir a segurança da rodoviária 24h. Os vigias não podem portar e utilizar armas. Um dos vigias, indignado, cria uma conta de e-mail gratuita e envia um e-mail para o responsável da rodoviária, relatando que os vândalos fazem o que querem com os vigias, que ele é um incompetente e outras coisas mais (palavras de baixo calão). Por fim, o ameaça e a sua família também.

Após a análise do e-mail impresso e com depoimento do responsável pela rodoviária, deduz-se que o e-mail foi criado por um funcionário, mais especificamente, um vigia. Sabe-se que os vigias utilizam um computador com acesso à Internet, disponível para os funcionários durante o horário de descanso. A polícia decide, então, apreender este computador e enviar juntamente com o e-mail impresso, à perícia.

Estava traçado então o objetivo da perícia, ou seja, saber se aquele e-mail impresso foi originado na máquina enviada para a investigação. Depois de realizada a cópia bit-a-bit do disco rígido questionado, começou-se a busca. Por se tratar de um e-mail enviado a partir de um site (Webmail), foram realizadas duas atividades em paralelo. Foram selecionadas palavras-chave para busca na imagem. Foram selecionadas palavras com erros de grafia encontradas no e-mail impresso e, também, palavras muito específicas (que raramente seriam encontradas em outro arquivo ou trecho do disco). Enquanto essa busca era processada, foram verificados os arquivos encontrados na pasta de arquivos temporários da Internet.

Diversos acessos a sites de Webmail foram encontrados, com um total de seis usuários diferentes. No entanto, na pasta de arquivos temporários não foi encontrado o e-mail com as ameaças nem mesmo os dados da conta de e-mail que foi utilizada. Antes de continuar a análise “manual”, o processamento da busca foi concluído, mostrando algumas ocorrências para as palavras-chave selecionadas. Analisando as ocorrências, foi encontrado a maior parte do e-mail dentro do arquivo de memória virtual (plataforma Windows), indicando que algum arquivo ou mensagem na Internet foi visualizado no computador. Antes do início da mensagem havia o seguinte metadado:

<?xml:namespace prefix = o ns = "urn:schemas-microsoft-com:office:office"

Este metadado indica a utilização de um dos aplicativos do Microsoft Office. Neste caso, há o indício de que a mensagem teria sido digitada no aplicativo de edição de texto do pacote Office, antes de ser enviado por e-mail. Além destas informações, nada mais foi encontrado. Logo, não houve uma conclusão de que a mensagem foi originada na máquina analisada. Porém foram relatados os indícios encontrados, podendo ajudar na investigação.

1.6.4 Injúria por E-mail (com cabeçalho)

O quarto caso trata de uma mensagem de e-mail contendo injúria, mais especificamente, racismo. O e-mail foi enviado para a amiga da vítima, sendo que a amiga mostrou para a vítima e esta decidiu registrar ocorrência, levando consigo o e-mail impresso. O delegado decidiu então solicitar o computador da amiga para enviar à perícia. Neste caso, foi possível analisar o cabeçalho do e-mail, ilustrado na Figura 12. Alguns dados foram propositalmente alterados para evitar a identificação dos envolvidos.

Received: from email-2.ig.com.br ([10.10.1.11]) by mailserver-4.ig.com.br
(Sun Internet Mail Server sims.4.0.2000.10.12.16.25.p8)
with ESMTP id <0GB7003QEZIO16@mailserver-4.ig.com.br> for
rf.teste@sims-ms-daemon (ORCPT rfc822;rf.teste@ig.com.br); Tue,
23 Apr 2002 11:13:36 -0300 (EST)

Received: from srv-int.empresa.com.br ([210.218.115.13])
by email-2.ig.com.br (Sun Internet Mail Server sims.4.0.2000.10.12.16.25.p8)
with ESMTP id <0GB700EMYZICO0@email-2.ig.com.br> for
rf.teste@mailserver-4.ig.com.br (ORCPT rfc822;rf.teste@ig.com.br); Tue,
23 Apr 2002 11:13:25 -0300 (EST)

Received: from 01gecad ([192.168.1.22]) by srv-int.empresa.com.br
(8.9.3/8.9.3) with SMTP id LAA01485 for <rf.teste@ig.com.br>; Tue,
23 Apr 2002 11:13:03 +0000 (GMT)

Date: Tue, 23 Apr 2002 11:10:58 -0300
From: Nome da Empresa <josericardo@empresa.com.br>
Subject: QUE ABSURDO
To: Raimunda Francisca Teste <rf.teste@ig.com.br>
Reply-to: Nome da Empresa <josericardo@empresa.com.br>

Figura 12. Exemplo de um cabeçalho de e-mail

Analisando os cabeçalhos (em negrito) de baixo para cima, é possível verificar por quais servidores passou o e-mail questionado. O primeiro servidor é denominado “svr-int.empresa.com.br”, a máquina que enviou o e-mail possui nome “01gecad” e endereço IP privado 192.168.1.22. Na segunda parte do cabeçalho (de baixo para cima) é possível verificar que o servidor “email-2.ig.com.br” recebeu diretamente do servidor de e-mail da empresa e que o endereço IP do servidor de e-mail da empresa é 210.218.115.13. Após, na última parte (a do topo) é possível ver um encaminhamento de um servidor para outro no mesmo domínio “ig.com.br”.

Foi possível consultar na base de dados de domínios brasileiros [NIC.br 2007] e os dados da empresa, como endereço, telefone do responsável, entre outros, puderam ser coletados. Atualmente os dados não são acessíveis por qualquer usuário, porém na época em que foi realizada a perícia, estes dados eram acessíveis. Estes dados foram

colocados no laudo, indicando a origem do e-mail sendo a empresa descrita como detentora do domínio “empresa.com.br”, localizada na mesma cidade onde morava a vítima e o acusado, e, segundo informações que a investigação passou para os peritos posteriormente, esta seria a empresa onde o acusado trabalhava.

1.7 Desafios Atuais em Forense Computacional

Essa seção apresenta as oportunidades de pesquisa identificadas pelos autores durante o processo de revisão bibliográfica que antecedeu a proposta do minicurso em questão. As questões de pesquisa mencionadas estão organizadas de acordo com o processo de investigação apresentado na Seção 1.3.

1.7.1 Coleta dos Dados

Conforme discutido ao longo das seções anteriores o sucesso de uma investigação depende muito da coleta e da preservação da integridade dos dados, portanto esta etapa do processo forense deve ser realizada de maneira sistemática e tão logo as fontes de evidências sejam localizadas. As principais fontes de evidências existentes são: os discos rígidos e a memória física dos *hosts*, os *logs* dos diversos serviços em execução e o tráfego da rede capturado.

Embora existam ferramentas especializadas para coletar e preservar esses tipos de fontes de dados, essa é uma etapa que ainda envolve muito tempo até ser finalizada, pois além do deslocamento do perito até o local da cena do crime é necessário considerar o tempo gasto para coletar os dados de cada um dos hosts apreendidos, o que pode demandar muitas horas, principalmente, se considerarmos que as estações e os servidores podem estar ligados, o que requer uma série de cuidados extras para que não haja a contaminação dos dados coletados (vide seção 1.3.3).

Para facilitar e agilizar a aquisição das evidências em *hosts* que estejam conectados a rede, foram desenvolvidas ferramentas para forense remota que a partir de uma console central são capazes de coletar simultaneamente dados de diversos *hosts*. Entretanto, este tipo de aplicação não é capaz de realizar a cópia completa da memória ou inspecionar as áreas de memória alocadas por um determinado processo [Casey 2006]. Sendo assim, com este tipo de ferramenta não é possível, por exemplo, adquirir evidências localizadas em arquivos abertos ou referentes a processos em execução.

Ainda no que diz respeito a coleta de informações em sistemas ativos, [Carrier 2006] afirma que os *rootkits* representam a principal ameaça e indica como contramedida o uso de *hardware* especializado para realizar a cópia do conteúdo da memória. Por exemplo, o sistema *Tribble* [Carrier 2004] é um cartão PCI que pode realizar a cópia da memória física usando requisições DMA, sem a mediação do *kernel*. Portanto, ainda que o *kernel* do *host* apreendido esteja comprometido, é possível obter a imagem da memória para ser examinada em uma estação forense confiável.

Além das questões recém mencionadas outro tópico que tem recebido a atenção dos pesquisadores refere-se a padronização dos formatos utilizados para armazenar a imagem dos discos rígidos. Segundo [Hosmer 2006] e [Garfinkel 2007] o padrão de fato para copiar informações de uma mídia é o formado denominado “*raw format*”, aquele no qual os dados são copiados setor por setor. Entretanto, nesse caso não são coletados metadados, tais como: o número serial dos dispositivos, a data e o local em que os

dados foram adquiridos e nem mesmo a assinatura digital para garantir a integridade dos dados. Além disso, este formato não diferencia setores em branco daqueles que não são acessíveis e, ainda, não oferece suporte a compactação, o que resulta no desperdício do espaço em disco (aquele para o qual os dados estão sendo copiados).

Segundo publicações do *The Common Digital Evidence Storage Format Working Group* (CDESF) alguns formatos proprietários resolvem parte das questões supracitadas, contudo criam outras limitações, entre elas:

- falta de compatibilidade entre os padrões o que, em uma tentativa de conversão entre os formatos, pode resultar em dados incorretos, perda de metadados e de tempo;
- para os investigadores que não tiverem acesso ao software de leitura do padrão gerado não há como recuperar os dados o que impossibilita o exame e análise dos dados;
- em alguns casos juízes ou advogados podem rejeitar evidências em formatos proprietários, com base na alegação de que esses padrões estão sujeitos ou violam algumas patentes.

O desenvolvimento de padrões abertos bem documentados, que contemplem acesso aos metadados, que garantam a integridade dos dados e cujas evidências possam ser examinadas e analisadas por múltiplas ferramentas, é um desafio em aberto e que representa o foco das pesquisas realizadas pelo grupo de trabalho CDESF, anteriormente mencionado. Outras características desejáveis são:

- a definição da estrutura e implementação de trilhas de auditoria que gerem a cadeia de custódia, dessa maneira seria registrado de forma segura e automática todas as ações relacionadas a aquisição e alteração dos dados adquiridos;
- a implementação das seguintes características de segurança: mecanismos de autenticação, verificação de integridade, controle de acesso as evidências e não-repúdio;

Em [Hosmer 2006] é apresentado o conceito de *Digital Evidence Bag* (DEB) uma espécie de container digital no qual as evidências coletadas são armazenadas. As diferenças existentes entre um container do mundo real e o DEB são as seguintes: o container digital permite a duplicação, cópia, e compartilhamento do seu conteúdo de maneira segura, pois prevê as características de segurança descritas acima, muito embora se reconheça que ainda é necessário continuar pesquisando e aprimorando o conceito de DEB até que seja possível implementar um protótipo com todos os atributos aqui elencados.

Ainda sobre o desenvolvimento de formatos para gerar e armazenar imagens de discos rígidos, cabe mencionar o *Advanced Forensics Format* (AFF), um formato aberto e flexível que armazena a imagem do disco em um conjunto de páginas, o que permite compactar a imagem gerada [Garfinkel 2007]. Além disso, (a) este formato oferece a possibilidade de armazenar os metadados na própria imagem ou em um arquivo separado, (b) está livre de patentes e segredos comerciais e (c) os desenvolvedores disponibilizaram um biblioteca e um conjunto de ferramentas para gerar imagem de

discos e converter formatos de imagens para o padrão AFF. Sendo assim, existe a possibilidade de integrar e desenvolver ferramentas para coleta de dados que ofereçam suporte a este padrão, inclusive este é o objetivo do projeto de acordo com [Garfinkel 2007].

Além das características mencionadas anteriormente é desejável que um formato aberto para armazenamento dos dados seja flexível e aplicável à diferentes formas de evidências digitais (tráfego de rede e *dumps* de memória), ou seja, não esteja restrito somente a dados armazenados nos discos rígidos.

1.7.2 Exame dos Dados

Nessa etapa o primeiro desafio é identificar entre todos os dados armazenados nas mídias coletadas junto as estações e servidores, quais são relevantes para auxiliar na elucidação dos fatos investigados. Essa tarefa é complexa devido, sobretudo, aos seguintes aspectos:

1. a capacidade cada vez maior de armazenamento dos dispositivos;
2. os arquivos podem ter sido criptografados ou esteganografado;
3. a presença de *rootkits* induz o perito a erros no momento de avaliar quais as informações que serão filtradas e encaminhadas para a etapa seguinte do processo de investigação;
4. um arquivo pode ter sido fragmentado e armazenado em espaços não alocados do disco ou ainda marcados, indevidamente, como bad blocks.

Essa série de barreiras faz com que seja necessário muitas horas (ou até mesmo dias) até que seja possível identificar os dados pertinentes ao caso em questão. Com o intuito de localizar tais dados, são utilizadas ferramentas que implementam técnicas de pesquisa baseada na assinaturas dos arquivos. Essas aplicações varrem o disco rígido relacionando as assinaturas dos arquivos do sistema com a sua base de dados, dessa forma é possível identificar: arquivos comprometidos por códigos maliciosos ou com ADS (vide subseção 1.4.3), arquivos criptografados, esteganografados ou protegidos por senhas.

Já para identificar a presença de *rootkits* sem comprometer os dados ou perturbar o ambiente investigado, [Carrier 2006] recomenda o desenvolvimento e utilização de mecanismos de detecção de rootkits baseado em hardware, como o *Copilot* proposto em [Nick 2004].

De acordo com [Casey 2006], atualmente não existem ferramentas para inspecionar e interpretar, de maneira ágil, as estruturas de dados da memória virtual. O desenvolvimento de tais aplicações pode auxiliar o investigador a encontrar senhas de usuários, fragmentos de arquivos visualizados pelo intruso e até mesmo senhas de arquivos criptografados em um curto espaço de tempo.

Outra questão em aberto e que esta relacionada ao exame dos dados, refere-se ao desenvolvimento de métodos que possibilitem identificar a presença de arquivos, cujos fragmentados foram gravados em espaços não alocados do disco – técnica utilizada por algumas ferramentas anti-forense.

Portanto, o desenvolvimento de técnicas e ferramentas projetadas para solucionar as questões mencionados nessa subseção, no menor tempo possível e com baixas taxas de falsos positivos, representam questões de pesquisas relevantes e cujos avanços podem representar um passo significativo para acelerar o processo de exame dos dados e garantir que as informações enviadas à análise sejam confiáveis.

1.7.3 Análise dos Dados

Após conseguir extrair das mídias somente os dados que são pertinentes a investigação o perito ainda tem pela frente outro grande desafio, identificar, entender e reconstruir os fatos ocorridos. Para tal, é necessário desenvolver e aprimorar (a) os métodos de redução de dados, (b) os mecanismos de reconhecimento de padrões de comportamento e (c) as técnicas de correlacionamento de alertas [Forte 2004].

Segundo [Casey 2006] ferramentas proprietárias voltadas à gerência de segurança, tais como: *CS-MARS* e *nFX*, embora não tenham sido projetadas com o propósito de serem utilizada para fins de análise forense, fornecem alguns dos recursos mencionados por [Forte 2004]. Portanto, o desenvolvimento de sistemas especialistas em forense que possua em suas bases de dados informações sobre como: agregar as diversas entradas dos arquivos de log distribuídos pelos ativos da organização em um único evento, correlacionar eventos de forma automatizada e que forneçam informações detalhadas sobre o ocorrido, permitem reduzir o tempo gasto na análise dos dados coletados, pois além de tornar o processo automatizado torna-o menos suscetível a erros, uma vez que não há mais a dependência unicamente do conhecimento do investigador para analisar os dados.

Uma vez que tenham sido empregadas técnicas adequadas para redução e correlacionamentos das evidências, o perito necessita realizar a análise desses dados, o que pode ser feito de forma rápida e fácil através de técnicas de visualização hierárquicas e não hierárquicas ou como uso de diagramas de link.

Técnicas de visualização não hierárquicas mostram dados estatísticos sobre os arquivos existentes em um diretório ou subdiretório sem nenhuma informação sobre o relacionamento entre arquivos e diretórios. Nesse tipo de técnica os arquivos são representados por quadrados, cujas cores indicam o tamanho dos arquivos, tonalidades escuras representam arquivos de tamanho menores, enquanto as tonalidades claras representam os arquivos maiores. Quando o parâmetro utilizado para consulta for um atributo de tempo, os quadrados de cor mais clara representam os arquivos acessados recentemente. Além do tamanho e dos atributos de tempo podem ser utilizados outros parâmetros (por exemplo, o formato do arquivo), o que permite analisar diferentes cenários de forma rápida e agradável [Teelink and Erbacher 2006].

As técnicas de visualização hierárquicas apresentam os dados em uma estrutura de árvore, como ilustrado na Figura 13, mantendo a relação entre os arquivos e os diretórios e possibilitando diferentes métodos de análise, por exemplo, cada arquivo é representado por um quadrado sombreado, cujo tamanho informa, entre outras coisas, qual o percentual do espaço total do diretório é ocupado por este arquivo. O *popup* indica uma discrepância entre o nome e o tipo de arquivo, o que indica uma tentativa de ocultar dados. Além disso, os atributos relacionados a data e hora do arquivo também são exibidos [Teelink and Erbacher 2006].

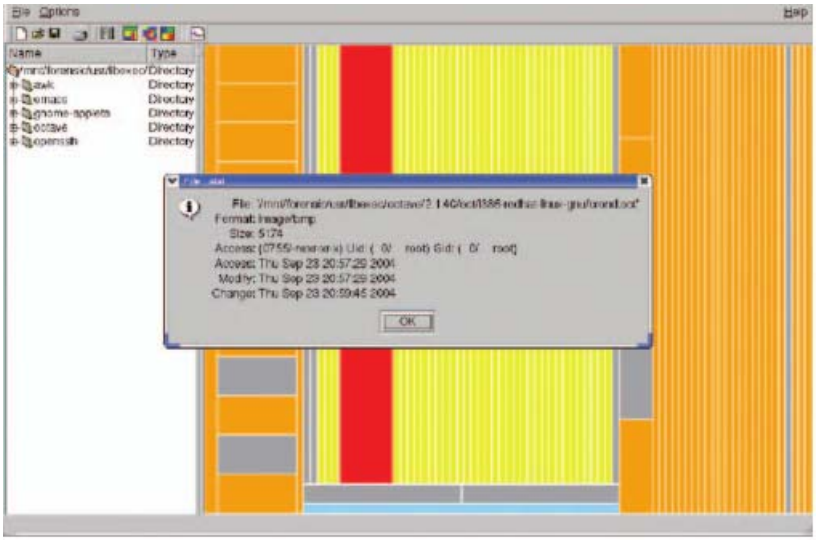


Figura 13. Diagrama hierárquico em estrutura de árvore

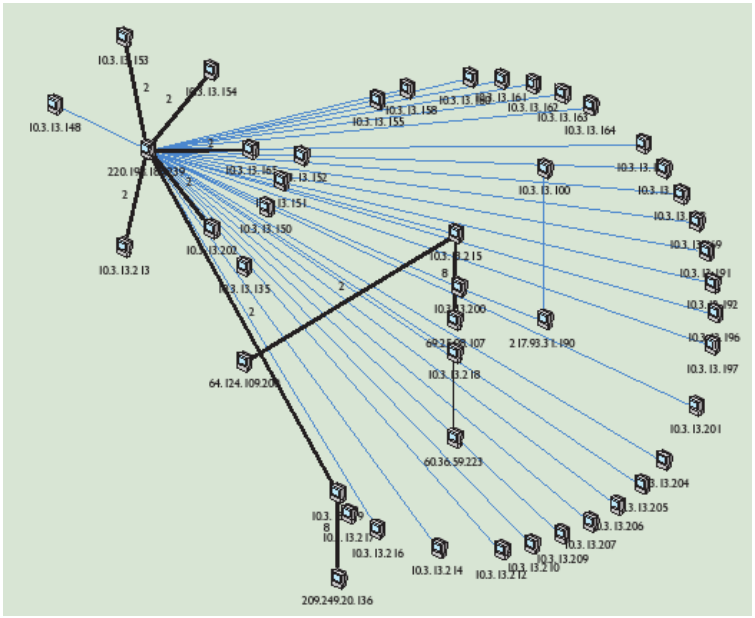


Figura 14. Análise através de diagramas de *link*

O diagrama de link é uma técnica que permite gerar filtros em função das comunicações estabelecidas e visualizar graficamente as informações existentes. No exemplo da Figura 14, os *logs* indicam que o *host*, cujo endereço IP é 220.198.186.239 (no canto superior esquerdo) acessou ou comprometeu as demais estações.

Outra maneira interessante de analisar os dados é exibindo-os ao longo da linha do tempo, o que permite reconstruir os fatos de maneira cronológica, recurso que os autores desse capítulo encontraram apenas na ferramenta proprietária *Analysts Notebook*[3], conforme ilustrado na Figura 15.

[3] http://www.i2inc.com/Products/

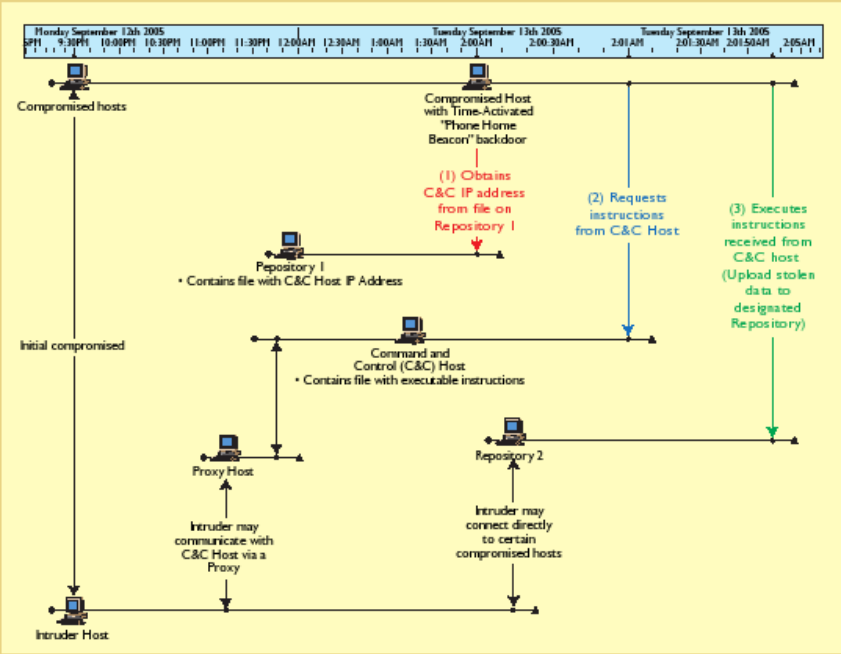


Figura 15. Análise Cronológica de Eventos

1.8 Considerações Finais

Este capítulo apresentou a forense computacional, um tema bastante atual e que tem recebido significativa atenção tanto da comunidade científica quanto da indústria. A seção sobre *malwares* apresentou os principais códigos maliciosos e exemplificou as ações realizadas pelos mesmos.

Na Seção 1.3 foi realizada uma introdução à Forense Computacional e em seguida o processo de investigação forense foi descrito tendo como base as seguintes etapas: coleta e exame dos dados, análise das informações e interpretação dos resultados. Por fim, foram apresentados características e cuidados referentes à *live forensics*. Sobre as quatro etapas mencionadas é importante ressaltar: (a) a importância de salvaguardar e garantir a integridade dos dados coletados, bem como estabelecer e manter a cadeia de custódia do material apreendido, (b) o exame das mídias deve ser realizado com o auxilio de ferramentas que permitam ao perito, no menor intervalo de tempo possível, separar os dados relevantes para (c) realizar a análise do dados, essa etapa requer o correlacionamento das informações geradas em diferentes serviços (Web, FTP e E-Mail) com os alertas gerados pelos mecanismos de proteção (*firewalls* e sistemas de detecção de intrusão) e (d) o processo é concluído com a elaboração de uma laudo técnico que descreve como os procedimentos foram realizados e quais as conclusões obtidas.

No que tange a análise de sistemas ativos, *live analysis*, cabe ressaltar que essa técnica pode fornecer evidências que não estão disponíveis na abordagem tradicional, *post mortem analysis*, contudo, em algumas situações, esse tipo de análise pode gerar resultados pouco confiáveis. Entretanto, casos em que a investigação deve ocorrer em sigilo ou em que é necessário monitorar as ações de colaboradores ou suspeitos, o tipo de abordagem mais recomendado é a *live analysis*.

A Seções subseqüentes apresentaram e discutiram problemas e questões atuais de pesquisa relacionadas a Forense Computacional, algumas totalmente em aberto outras já com alguns trabalhos sendo desenvolvidos, mas ambas extremamente relevantes e com espaço para novos contribuições.

Além dos assuntos mencionados, a Forense Digital – que conforme mencionado na Seção 1.3 possui um escopo mais abrangente do que a Forense computacional no que diz respeito ao tipo dispositivos analisados – também apresenta uma série de questões a serem tratadas, por exemplo: o desenvolvimento e aprimoramento de metodologias, ferramentas e técnicas que ofereçam suporte ao processo de forense em dispositivos móveis como aparelhos celulares e PDAs [Jansen and Ayers 2004].

Referências Bibliográficas

[Adelstein 2006] Adelstein, F. (2006). Live forensics: diagnosing your system without killing it first. Commun. ACM, 49(2):63–66.

[Adleman 1990] Adleman, L. M. (1990). An abstract theory of computer viruses. pages 1–354.

[Aleph-Null 1971] Aleph-Null (1971). Software - practice and experience. volume 1, pages 201–204.

[Bessa 2006] Bessa, L. (2006). Websense revela suas previsões sobre a segurança da internet para 2007. IMS Marketing. Websense, Inc, http://www.websense.com/global/pt/PressRoom/PressReleases/PressReleaseDetail/index.php?Release=0612191332.

[BlazingTools 2007] BlazingTools (2007). Perfect keylogger - easy to use stealth solution for pc and internet surveillance. discover the truth now! http://www.blazingtools.com/bpk.html.

[Bonfante et al. 2007] Bonfante, G., Kaczmarek, M., and Marion, J. Y. (2007) Toward an abstract computer virology.

[Cambridge 2007] Cambridge, A. L. (2007). VNC - Virtual Network Computing from AT&T Laboratories Cambridge.

[Carrier 2007a] Carrier, B. (2007a). Autopsy forensic browser. SourceForge.net, http://www.sleuthkit.org/autopsy/desc.php.

[Carrier 2007b] Carrier, B. (2007b). mactime. SouceForge.net, http://www.sleuthkit.org/sleuthkit/man/mactime.html.

[Carrier 2007c] Carrier, B. (2007c). The sleuth kit. http://www.sleuthkit.org/sleuthkit/desc.php.

[Carrier 2006] Carrier, B. D. (2006). Risks of live digital forensic analysis. Commun. ACM, 49(2):56–61.

[Carrier 2004] Carrier, I. B. (2004). A hardware-based memory acquisition procedure for digital.

[Case and Moelius 2007] Case, J. and Moelius, S. E. (2007). Cautious virus detection in the extreme. In Proceedings of the 2007 workshop on Programming languages and analysis for security (PLAS 2007), pages 47–52, New York, NY, USA. ACM Press.

[Casey 2006] Casey, E. (2006). Investigating sophisticated security breaches. Commun. ACM, 49(2):48–55.

[CERT 1999] CERT (1999). CERT Advisory CA-1999-04 Melissa Macro Virus. CERT Coordination Center (CERT/CC), http://www.cert.org/advisories/CA-1999-04.html.

[CERT 2002a] CERT (2002a). CERT Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL. CERT Coordination Center (CERT/CC), http://www.cert.org/advisories/CA-2001-13.html.

[CERT 2002b] CERT (2002b). CERT advisory CA-2002-30 trojan horse tcpdump and libpcap distributions. CERT Coordination Center (CERT/CC), http://www.cert.org/advisories/CA-2002-30.html.

[CERT.br 2007] CERT.br (2007). Centro de estudos, resposta e tratamento de incidentes de segurança no brasil. Comitê Gestor da Internet no Brasil - CGI.br, http://www.cert.br/.

[Chawki 2005] Chawki, M. (2005). A critical look at the regulation of cybercrime. http://www.crime-research.org/articles/Critical/.

[Cohen 1987] Cohen, F. (1987). Computer viruses: theory and experiments. Comput. Secur., 6(1):22–35.

[Collett and Cohen 2007] Collett, D. and Cohen, M. (2007). Forensic and log analysis gui. SourceForge.net, http://sourceforge.net/projects/pyflag/.

[Crapanzano 2003] Crapanzano, J. (2003). Deconstructing subseven, the trojan horse of choice. Technical report, SANS Institute.

[DCFL 2007] DCFL (2007). dcfldd. Department of Defense Computer Forensics Lab, http://dcfldd.sourceforge.net/.

[Dildog 2007] Dildog (2007). BO2K - Opensource Remote Administration Tool. Cult of the Dead Cow, http://www.bo2k.com/.

[eEye 2007] eEye (2007). eEye Digital Security website. http://www.eeye.com/html/index.html.

[Etrust 2007] Etrust (2007). Pestpatrol anti-spyware. http://www.pestpatrol.com/.

[Farmer and Venema 2006] Farmer, D. and Venema, W. (2006). Perícia Forense Computacional - Teoria e Prática Aplicada. 1 edition.

[Forte 2004] Forte, D. (2004). The art of log correlation. HTCIA Worldwide Conference, http://www.dflabs.com/images/Art_of_correlation_Dario_Forte.pdf.

[Foundstone 2007] Foundstone (2007). Foundstone network security: Risk management. FoundStone. McAfee, Inc, http://www.foundstone.com/us/resources-free-tools.asp.

[FTC 2005] FTC (2005). The US SAFE WEB Act: Protecting Consumers from Spam, Spyware, and Fraud. A Legislative Recommendation to Congress. Federal Trade Commission, http://ftc.gov/reports/ussafeweb/USSAFEWEB.pdf.

[Garfinkel 2007] Garfinkel, S. L. (2007). Advanced forensic format (aff). Simson L. Garfinkel and Basis Technology Corp, http://www.afflib.org/.

[Giacobbi 2007] Giacobbi, G. (2007). The gnu netcat - official homepage. http://netcat.sourceforge.net/.

[Gibson 2007] Gibson, S. (2007). Automated image and restore (air). SourceForge.net, https://sourceforge.net/projects/air-imager/.

[Guidance 2007] Guidance (2007). Encase forensic. Guidance Software, Inc, http://www.guidancesoftware.com/products/ef_index.asp.

[Haagman and Ghavalas 2005] Haagman, D. and Ghavalas, B. (2005). Trojan defence: A forensic view. Digital Investigation, 2(1):23–30.

[Harris 2006] Harris, R. (2006). Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. In The 6th Annual Digital Forensic Research Workshop (DFRWS 2006).

[Hoglund and Butler 2005] Hoglund, G. and Butler, J. (2005). Rootkits: Subverting the Windows Kernel. Addison-Wesley Professional.

[Holz 2005] Holz, T. (2005). A Short Visit to the Bot Zoo [malicious bots software]. IEEE Security & Privacy Magazine, 3(3):76–79.

[Hosmer 2006] Hosmer, C. (2006). Digital Evidence Bag. Commun. ACM, 49(2):69–70.

[IBM 2007] IBM (2007). Michelangelo madness. IBM Research, http://www.research.ibm.com/antivirus/SciPapers/White/VB95/vb95.distribnode7.html.

[Inman and Rudin 2000] Inman, K. and Rudin, N. (2000). Principles and Practice of Criminalistics: The Profession of Forensic Science (Protocols in Forensic Science). CRC.

[Jansen and Ayers 2004] Jansen, W. and Ayers, R. (2004). Guidelines on PDA Forensics: recommendations of the national institute of standards of and technology. Department of Homeland Security. National Institute of Standards and Technology, http://csrc.nist.gov/publications/nistpubs/800-72/sp800-72.pdf.

[Jones 2007a] Jones, K. J. (2007a). Galleta - an internet explorer cookie forensic analysis tool. Foundstone, Inc, http://www.foundstone.com/us/resources/proddesc/galleta.htm.

[Jones 2007b] Jones, K. J. (2007b). Pasco - an internet explorer activity forensic analysis tool. Foundstone, Inc, http://www.foundstone.com/us/resources/proddesc/pasco.htm.

[Kaspersky 2007] Kaspersky (2007). Malware descriptions: Classic viruses. Kaspersky Lab, http://www.viruslist.com/en/virusesdescribed?chapter=152540474.

[Kent et al. 2006] Kent, K., Chevalier, S., Grance, T., and Dang, H. (2006). Guide to integrating forensic techniques into incident response: Recommendations of the national institute of standards and technology. NIST Special Publication 800-86. National Institute of Standards and Technology (NIST), http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf.

[Klaus and Nelson 2001] Klaus, S. and Nelson, M. (2001). Métodos para detecção local de rootkits e módulos de kernel maliciosos em sistemas unix. In III Simpósio sobre Segurança em Informática (SSI), São José dos Campos, SP.

[Kolla 2007] Kolla, P. M. (2007). Spybot search and destroy website. http://www.safer-networking.org/pt/index.html.

[Kolter and Maloof 2006] Kolter, Z. J. and Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res., 7:2721–2744.

[Kruse and Heiser 2001] Kruse, W. G. and Heiser, J. G. (2001). Computer Forensics : Incident Response Essentials. Addison-Wesley Professional.

[Lavasoft 2007] Lavasoft (2007). Ad-Aware website. Lavasoft AB, http://www.lavasoftusa.com/software/adaware.

[Mclaughlin 2004] Mclaughlin, L. (2004). Bot software spreads, causes new worries. IEEE Distributed Systems Online, 5(6).

[Microsoft 2007a] Microsoft (2007a). Windows defender home. http://www.microsoft.com/athome/security/spyware/software/default.mspx.

[Microsoft 2007b] Microsoft (2007b). Windows sysinternals. Microsoft Technet. Microsoft Corporation, http://www.microsoft.com/technet/sysinternals/default.mspx.

[Neukamp 2007] Neukamp, P. (2007). Fdtk-ubuntubr: Linux forense digital toolkit. http://www.fdtk-ubuntubr.1br.net/.

[Newman 2006] Newman, R. C. (2006). Cybercrime, identity theft, and fraud: practicing safe internet - network security threats and vulnerabilities. In InfoSecCD ’06: Proceedings of the 3rd annual conference on Information security curriculum development, pages 68–78, New York, NY, USA. ACM Press.

[NIC.br 2007] NIC.br (2007). Registro de domínios para a internet no brasil. Núcleo de Informação e Coordenação do Ponto br - NIC.br. Comitê Gestor da Internet no Brasil - CGI.br, http://registro.br/.

[Nick 2004] Nick (2004). Copilot – a coprocessor-based kernel runtime integrity monitor. pages 179–194.

[NIST 2007] NIST (2007). National Institute Of Standards And Technology (NIST). U.S. Commerce Department’s Technology Administration, http://www.nist.gov/.

[NSRL 2007] NSRL (2007). National Software Reference Library (NSRL). National Institute of Standards and Technology (NIST). U.S. Department of Justice’s National Institute of Justice (NIJ), http://www.nsrl.nist.gov/.

[NWCCC and FBI 2006] NWCCC and FBI (2006). Internet crime report. Prepared by the National White Collar Crime Center and Federal Bureau of Investigation. The Internet Crime Complaint Center (IC3), http://www.ic3.gov/media/annualreport/2006_IC3Report.pdf.

[OpenGroup 2007] OpenGroup (2007). dd - convert and copy a file. The Open Group, http://www.opengroup.org/onlinepubs/009695399/utilities/dd.html.

[Palmer and Corporation 2001] Palmer, G. and Corporation, M. (2001). A road map for digital forensic research. Technical report.

[Pavlov 2007] Pavlov, I. (2007). LZMA SDK (Software Development Kit). http://www.7-zip.org/sdk.html.

[Payton 2006] Payton, A. M. (2006). A review of spyware campaigns and strategies to combat them. In InfoSecCD ’06: Proceedings of the 3rd annual conference on Information security curriculum development, pages 136–141, New York, NY, USA. ACM Press.

[Phrack 2007] Phrack (2007). Project loki. http://www.phrack.org/issues.html?issue=49&id=6#article.

[Ramachandran and Feamster 2006] Ramachandran, A. and Feamster, N. (2006). Understanding the network-level behavior of spammers. In Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2006), pages 291–302, New York, NY, USA. ACM Press.

[Richard and Roussev 2006] Richard, G. G. and Roussev, V. (2006). Next-generation digital forensics. Communications of the ACM, 49(2):76–80.

[SecurityFocus 2007] SecurityFocus (2007). Securityfocus website. Symantec Corporation, http://www.securityfocus.com/.

[Shukla and Nah 2005] Shukla, S. and Nah, F. F. (2005). Web browsing and spyware intrusion. Commun. ACM, 48(8):85–90.

[Sipior et al. 2005] Sipior, J. C., Ward, B. T., and Roselli, G. R. (2005). A united states perspective on the ethical and legal issues of spyware. In ICEC ’05: Proceedings of the 7th international conference on Electronic commerce, pages 738–743, New York, NY, USA. ACM Press.

[Skoudis and Zeltser 2003] Skoudis, E. and Zeltser, L. (2003). Malware: Fighting Malicious Code. Prentice Hall PTR.

[Solove and Rotenberg 2003] Solove, D. J. and Rotenberg, M. (2003). Information Privacy Law (Aspen Elective Series). Aspen Publishers.

[Sophos 2001] Sophos (2001). Glossary of terms: Companion virus. Sophos Plc., http://www.sophos.com/pressoffice/news/articles/2001/11/va_glossary.html#comp.

[Sophos 2007] Sophos (2007). Sophos - anti-virus and anti-spam software for businesses. http://www.sophos.com/.

[Subramanya and Lakshminarasimhan 2001] Subramanya, S. R. and Lakshminarasimhan, N. (2001). Computer viruses. IEEE Potentials Magazine, 20(4):16–19.

[Symantec 2007a] Symantec (2007a). Understanding Virus Behavior under Windows NT. Symantec AntiVirus Research Center, http://securityresponse.symantec.com/avcenter/reference/virus.behavior.under.win.nt.pdf.

[Symantec 2007b] Symantec (2007b). Php.pirus. Symantec Corporation, http://www.symantec.com/security_response/writeup.jsp?docid=2000-122009-2642-99.

[Teelink and Erbacher 2006] Teelink, S. and Erbacher, R. F. (2006). Improving the computer forensic analysis process through visualization. Commun. ACM, 49(2):71–75.

[Tham 2001] Tham, A. (2001). What is code red worm? As part of the Information Security Reading Room. SANS Institute, http://www.sans.org/reading_room/whitepapers/malicious/45.php.

[TST 2005] TST (2005). TST admite que empresa investigue e-mail de trabalho do empregado. Tribunal Superior do Trabalho, http://ext02.tst.gov.br/pls/no01/no_noticias.Exibe_Noticia?p_cod_noticia=5319&p_cod_area_noticia=ASCS.

[TST 2006] TST (2006). Processo E-ED-RR - 613/2000-013-10-00.7. Tribunal Superior do Trabalho, http://ext02.tst.gov.br/pls/ap01/ap_red100.resumo?num_int=29569&ano_int=2003&qtd_acesso=908889.

[Viotto 2007] Viotto, J. (2007). CSI Digital.

[Wang and Yu 2005] Wang, X. and Yu, H. (2005). How to break md5 and other hash functions. In Eurocrypt 2005, volume 3494, pages 19–35. Lecture Notes in Computer Science.

[Weiss 2005] Weiss, A. (2005). Spyware be gone! netWorker, 9(1):18–25.

[Zadjmool 2004] Zadjmool, R. (2004). Hidden threat: Alternate data streams. Articles :: Windows OS Security. Security Focus, http://www.windowsecurity.com/articles/Alternate_Data_Streams.html.

[Zhang and Paxson 2000] Zhang, Y. and Paxson, V. (2000). Detecting backdoors. In Proc. 9th USENIX Security Symposium, pages 157–170.

[Zou et al. 2005] Zou, C. C., Gong, W., Towsley, D., and Gao, L. (2005). The monitoring and early detection of internet worms. IEEE/ACM Trans. Netw., 13(5):961–974.

Capítulo 2

Esteganografia e suas Aplicações

Eduardo Pagani Julio, Wagner Gaspar Brazil, Célio Vinicius Neves Albuquerque

Universidade Federal Fluminense
Departamento de Ciência da Computação - Centro Tecnológico
{ejulio,wbrazil,celio}@ic.uff.br

Abstract

Steganography derives from the greek words stegano and graphy, where stegano means to hide, mask and graphy means to write. So, steganography is the art of cover writing. Along history, people has tried various forms to hide information within various media, searching in some form, to provide more privacy to their comunications. Some usual approaches to inserting messages into images include techniques such as: overwritting the least significant bit, as well as filtering, masquerading and and transformation algorithms. Each of these techniques can be applied to images with different levels of success. The goal of this course is to explore some steganography techniques since these techniques can be used to protect communications. Besides covering well-known techniques, we intend to show some of the applications and the applicability of steganography as an alternative to cryptographic methods.

Resumo

Esteganografia deriva do grego, onde estegano significa esconder, mascarar e grafia significa escrita. Logo, esteganografia é a arte da escrita encoberta. Durante toda a história, as pessoas buscam inúmeras formas de esconder informações dentro de outros meios, para, de alguma forma, obter mais privacidade para seus meios de comunicação. As abordagens mais comuns de inserção de mensagens em imagens incluem técnicas de: inserção no bit menos significativo, filtragem e mascaramento e algoritmos de transformações. Cada uma destas técnicas pode ser aplicada a imagens, com graus variados de sucesso. O objetivo deste curso é explorar as técnicas de esteganografia de maneira que

possam ser usadas na proteção das comunicações. Além disso, deseja-se mostrar as aplicações e a aplicabilidade da esteganografia como uma opção aos métodos de criptografia mais conhecidos.

2.1. Introdução

A segurança digital é uma área com grande potencial para pesquisa e desenvolvimento. Sistemas de detecção de intrusão, anti-vírus, *proxies* e *firewalls* ultimamente aparecem muito na mídia em geral e estão se tornando ferramentas de uso doméstico. É cada vez maior o número de pessoas que tentam a todo custo ludibriar as defesas para ter acesso a um dos bens mais preciosos da sociedade moderna: a informação. Por outro lado, existem outras pessoas que buscam o desenvolvimento e o estudo de técnicas para proteção das comunicações. As ferramentas e técnicas que provêem a segurança da informação são inúmeras e a criptografia está entre elas há milhares de anos.

Um dos ramos da criptografia é a esteganografia. De origem grega, a palavra significa a arte da escrita escondida (estegano = esconder e grafia = escrita). A esteganálise por sua vez é a arte de detectar mensagens escondidas nos mais diversos meios de comunicação. A esteganografia inclui um amplo conjunto de métodos e técnicas para prover comunicações secretas desenvolvidos ao longo da história. Dentre as técnicas se destacam: tintas invisíveis, micropontos, arranjo de caracteres (*character arrangement*), assinaturas digitais e canais escondidos (*covert channels*) (PETITCOLAS; ANDERSON; KUHN, 1999) (PETITCOLAS; KATZENBEISSER, 1999) (JOHNSON; JAJODIA, 1998).

As aplicações de esteganografia incluem identicação de componentes dentro de um subconjunto de dados, legendagem (*captioning*), rastreamento de documentos e certificação digital (*time-stamping*) e demonstração de que um conteúdo original não foi alterado (*tamper-proofing*). Entretanto, como qualquer técnica, a esteganografia pode ser usada correta ou incorretamente. Há indícios recentes de que a esteganografia tem sido utilizada para divulgar imagens de pornografia infantil na Internet (MORRIS, 2004) (HART; ASHCROFT; DANIELS, 2004), além das mensagens de redes terroristas como a Al-Qaeda.

2.1.1. Terminologia

Há um interesse cada vez maior, por diferentes comunidades de pesquisa, no campo da esteganografia, marcas dágua e seriação digitais. Com certeza, isso leva a uma certa confusão na terminologia. A seguir, encontram-se alguns dos principais termos utilizados nestas áreas e ilustrados na Figura 2.1:

- dado embutido ou *embedded data* - é o dado que será enviado de maneira secreta, normalmente em uma mensagem, texto ou figura;
- mensagem de cobertura ou *cover-message* - é a mensagem que servirá para mascarar o dado embutido. Esta mensagem de cobertura pode ser de áudio (*cover-audio*), de texto (*cover-text*) ou uma imagem (*cover-image*);
- estego-objeto ou *stego-object* - após a inserção do dado embutido na mensagem de cobertura se obtém o estego-objeto;

- estego-chave ou *stego-key* - adicionalmente pode ser usada uma chave para se inserir os dados do dado embutido na mensagem de cobertura. A esta chave dá-se o nome de estego-chave;

- número de série digital ou marca *fingerprinting* - consiste em uma série de números embutidos no material que será protegido a fim de provar a autoria do documento.

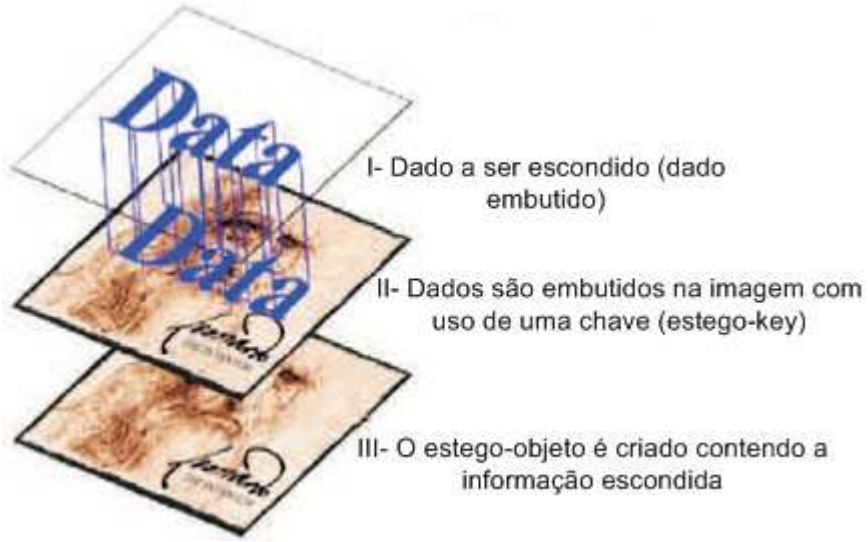


Figura 2.1: Escondendo uma imagem (PETITCOLAS; ANDERSON; KUHN, 1999).

A esteganografia pode ser dividida em dois tipos: técnica e lingüística. O primeiro tipo se refere às técnicas utilizadas quando a mensagem é fisicamente escondida, como por exemplo escrever uma mensagem em uma tábua de madeira e cobri-la com cera, como faziam alguns povos na antigüidade. A esteganografia lingüística se refere ao conjunto de técnicas que se utilizam de propriedades lingüísticas para esconder a informação, como por exemplo *spams* e imagens.

Os sistemas de marcação visam proteger a propriedade intelectual sobre algum tipo de mídia (eletrônica ou não). Estes sistemas de marcação são conhecidos também como *watermarking* (marca d'água). Apesar de aparecerem quase sempre em conjunto com esteganografia, os sistemas de marcação não pertencem ao ramo da esteganografia. Ambos pertencem a uma área de pesquisa conhecida como **ocultamento da informação** ou ***information hiding***.

O sistema de marcação tipo marca d'água se refere a métodos que escondem informações em objetos que são robustos e resistentes a modificações. Neste sentido seria impossível remover uma marca d'água de um objeto sem alterar a qualidade visual do mesmo. Por outro lado a esteganografia se propõe a esconder uma informação em uma imagem de cobertura. Se a imagem for destruída ou afetada a mensagem é perdida. Uma outra diferença clara entre esteganografia e técnicas de marca d'água é que enquanto o dado embutido da esteganografia nunca deve ficar aprente, a marca d'água pode ou não aparecer no objeto marcado, dependendo da aplicação que se queira atender.

Neste sentido pode-se classificar os sistemas de marcação segundo de acordo com a sua robustez e a sua aparência. Segundo sua robustez, podem ser classificados como:

- **robustos** - são aqueles em que mesmo após a tentativa de remoção a marca permanece intacta;
- **frágeis** - são os sistemas em que qualquer tentativa de modificação na mídia acarreta a perda da marcação. É muito útil para verificação de cópias ilegais. Quando se copia um objeto original, a cópia é feita sem a marca.

Já quanto a sua aparência, os sistemas de marcação podem ser classificados como:

- **de marcação imperceptível** - são os sistemas onde a marca encontra-se no objeto ou material, porém não é visível diretamente;
- **de marcação visível** - neste sistema a marca do autor deve ficar visível para comprovar a autoria visualmente. Um bom exemplo deste sistema são as marcas d´água em cédulas de dinheiro e em selos.

2.1.2. Aspectos Históricos

A esteganografia é uma arte antiga. Suas origens remontam à antiguidade. Os gregos já a utilizavam para enviar mensagens em tempos de guerra (KAHN, 1996). Nas "Estórias de Herodotus", existem muitas passagens mostrando o uso da esteganografia. Em uma estória, um mensageiro se disfarçou de caçador para enviar uma mensagem ao rei escondendo-a dentro de uma lebre. Como o mensageiro estava disfarçado, passou despercebido pelos portões do palácio e o rei pôde receber a mensagem.

Mensagens também foram enviadas através de escravos de confiança. Alguns reis raspavam as cabeças de escravos e tatuavam as mensagens nelas. Depois que o cabelo crescesse, o rei mandava o escravo pessoalmente com a mensagem (KAHN, 1996). Ninguém suspeitaria onde a mensagem se encontrava, a menos que soubesse exatamente onde procurar. Neste caso o segredo com a localização da mensagem deveria ser mantido. Outro exemplo de esteganografia na Grécia antiga era furar buracos em livros acima das letras que formavam a mensagem desejada. Quando o destinatário recebesse o livro poderia procurar pelos buracos sobre as letras para reconstruir as mensagens. Para quem não soubesse do código, o livro pareceria ter apenas seu conteúdo escrito pelo autor.

Os chineses e egípcios também criaram seus métodos de esteganografia na idade antiga. Os chineses escreviam mensagens em finas folhas de papel de seda que eram depois enroladas como uma bola e cobertos com cera. Esta bola era então escondida em algum lugar do corpo ou engolida para prevenir sua detecção. Os egípcios usavam ilustrações para cobrir as mensagens escondidas. O método de escrita egípcio conhecido como hieróglifo era uma técnica comum para esconder mensagens. Quando um mensageiro egípcio era pego com um hieróglifo que continha algum código, o inimigo não suspeitava e a mensagem podia ser entregue sem problemas ao destinatário.

Durante a idade média, a esteganografia foi mais estudada e desenvolvida. Em 1499, um monge chamado Tritheimius escreveu uma série de livros chamados "Steganographia" (Figura 2.2) nos quais ele descreveu várias técnicas diferentes. Uma delas, desenvolvida na idade média, foi a grade de Cardano (KAHN, 1996). Criada por Girolamo Cardano, a grade era uma lâmina que randomicamente definia retângulos. A quantidade e o posicionamento dos retângulos era o segredo da grade. O remetente escrevia as palavras da mensagem secreta nos retângulos. Depois a grade era removida e o remetente preenchia os espaços remanescentes com letras ou palavras para criar a mensagem que seria enviada (mensagem de cobertura). Uma vez entregue a mensagem, o destinatário colocaria a grade, que era a mesma do emissor, sobre o papel ou superfície que continha a mensagem e podia lê-la sem problemas, lendo os caracteres que estariam dentro dos retângulos.

Figura 2.2: Exemplar de *"Schola Steganographica"* publicado em 1680 (PETITCOLAS; KATZENBEISSER, 1999).

Os primeiros experimentos com tintas invisíveis também começaram na idade média. Giovanni Porta escreveu vários livros de história natural. Dentro destes livros estavam receitas de tintas secretas que poderiam ser usadas para escrever sobre a pele humana e outras superfícies. Este tipo de tinta foi desenvolvido e usado mais tarde no fim dos anos de 1700 e foi a chave para comunicações secretas.

Tintas invisíveis também foram muito usadas em esteganografia nos tempos mais modernos e são utilizadas até hoje. Estas tintas foram utilizadas por espiões durante a primeira e a segunda grande guerra com o desenvolvimento de reagentes químicos específicos para cada tinta. Textos eram escritos em jornais, revistas ou livros com tintas invisíveis para serem passados de forma segura até seus destinatários. Uma outra utilização era escrever a mensagem com tinta invisível sobre um papel, cortá-lo em alguns pedaços e depois rejuntá-los no destinatário (KAHN, 1996).

Outros métodos modernos de esteganografia incluem cifradores nulos e micro pontos. Cifradores nulos são mensagens nas quais certas letras devem ser usadas para formar a mensagem e todas as outras palavras ou letras são consideradas nulas. Para o uso do cifrador nulo, ambos os lados da comunicação devem usar o mesmo protocolo de uso das letras que formam a mensagem. Por exemplo, usar sempre a primeira letra de cada palavra para compor a mensagem. Este método é difícil de implementar, pois a mensagem de cobertura deve ter algum sentido, do contrário um inimigo desconfiará e

quebrará o código. Um exemplo de um código utilizando cifrador nulo é mostrado abaixo (JOHNSON, 1998).

"News Eight Weather: tonight increasing snow. Unexpected precipitation smothers eastern towns. Be extremely cautious and use snowtires especially heading east. The highways are knowingly slippery. Highway evacuation is suspected. Police report emergencies in downtown ending near Tuesday".

Usando as primeiras letras de cada palavra o texto que aparece é:

"Newt is upset because he thinks he is president".

A técnica de mMicro-pontos é também uma outra forma de esteganografia usada atualmente. Um micro-ponto é uma fotografia da mensagem secreta que deve ser entregue. Com a tecnologia avançando rapidamente, é possível tirar uma foto de uma mensagem e reduzi-la a uma fotografia circular de 0,05 polegadas ou 0,125 cm de diâmetro. Esta minúscula fotografia é então colada em um sinal de pontuação de uma frase ou no "pingo"de uma letra "i"de uma outra mensagem qualquer que será entregue. Somente aqueles que sabem onde procurar o micro-ponto poderão detectar sua presença.

Atualmente, novas técnicas de esteganografia são produzidas para serem utilizadas nos novos meios de comunicação. Por exemplo, hoje em dia muitos artistas e gravadoras estão utilizando a marca d'água para proteger suas obras. Com o crescente aumento da pirataria e de *sites* na Internet onde se pode baixar filmes, músicas e vídeos, esta técnica tem se mostrado uma aliada na proteção dos direitos autorais. O uso de esteganografia em software tem um grande potencial, pois pode esconder dados em uma infinidade de mídias. Nas técnicas que utilizam o último bit de um byte para esconder mensagens, uma mensagem de 64Kbytes pode ser escondida em uma figura de 1024 x 1024 em tons de cinza ou imagens coloridas. Esta e outras novas técnicas, representam o estado da arte da esteganografia atual e são apresentadas a seguir.

2.2. Estado da Arte

As imagens são a mídia de cobertura mais popular para esteganografia e podem ser armazenadas em um formato bitmap direto (como BMP) ou em um formato comprimido (como JPEG). Imagens de palheta de cores estão normalmente no formato GIF. O ocultamento de informações é realizado ou no domínio espacial ou no domínio de freqüência. Em termos de esquemas de inserção, vários métodos (como substituição, adição e ajuste) podem ser usados. Uma abordagem de ajuste é a QIM (*Quantization Index Modulation*), que usa diferentes quantizadores para transportar diferentes bits dos dados secretos (SULLIVAN et al., 2004).

As abordagens mais comuns de inserção de mensagens em imagens incluem técnicas de insercão no bit menos signicativo, técnicas de filtragem e mascaramento e algoritmos e transformações. Cada uma destas técnicas pode ser aplicada à imagens, com graus variados de sucesso. O metodo de inserção no bit menos signicativo é provavelmente uma das melhores técnicas de esteganografia em imagem (PETITCOLAS; ANDERSON; KUHN, 1999) (WAYNER, 2002).

2.2.1. Requisitos para Sistemas Esteganográficos

Os três requisitos mais importantes que devem ser satisfeitos para qualquer sistema esteganográfico são:

- segurança - a fim de não levantar suspeita, enquanto tenta criar uma blindagem contra um algorítmo de descoberta, o conteúdo escondido deve ser invisível tanto perceptivelmente quanto por meios estatísticos (BUCCIGROSSI; SIMONCELLI, 1999). Algumas definições baseadas em informações teóricas para um sistema seguro perfeito assumem conhecimento detalhado das estatísticas da cobertura e exigem recursos computacionais ilimitados. Estas condições não são estritamente encontradas em aplicações esteganográficas reais. Por exemplo, relativo a conhecimento estatístico, pode-se estimar estatísticamente um conjunto particular de sinais freqüentemente utilizados por um certo grupo de pessoas e estabelecer um modelo para descoberta. Mas tais modelos não tem sentido se o erro de estimação excede a extensão de modificações causadas por inclusão. Além disso, a complexidade computacional de qualquer ferramenta de esteganografia útil não pode ser infinitamente grande. Em termos de praticidade, um sistema pode ser considerado seguro, ou esteganograficamente forte (DUDA; HART; STORK, 2000), se não for possível descobrir a presença de stego-conteúdo usando qualquer meio acessível;

- carga útil - diferentemente de marca d'água, que precisa embutir somente uma quantia pequena de informações de direitos autorais, a esteganografia é direcionada à comunicação escondida e portanto normalmente exige capacidade de inclusão suficiente. Os requisitos para capacidade significativa de dados e segurança são freqüentemente contraditórios. Dependendo dos argumentos de aplicação específica, um compromisso deve ser buscado;

- robustez - embora robustez contra ataques não seja uma prioridade importante, como em marcas d'água, ter a capacidade de resistir a compressão é certamente desejável, pois a maioria das imagens JPEG coloridas são comprimidas antes de serem colocadas on-line.

2.2.2. LSB

Estas técnicas são baseadas na modificação dos bits menos significativos (*Least Significant Bit*) dos valores de pixel no domínio espacial. Em uma implementação básica, estes pixels substituem o plano LSB inteiro com o stego-dados. Com esquemas mais sofisticados em que locais de inclusão são adaptativamente selecionados, dependendo de características da visão humana, até uma pequena distorção é aceitavel. Em geral, a inclusão de LSB simples é suscetível a processamento de imagem, especialmente a compressão sem perda.

Técnicas baseadas em LSB podem ser aplicadas a cada *pixel* de uma imagem codificada em 32bits por *pixel*. Estas imagens possuem seus pixels codificados em quatro bytes. Um para o canal alfa (alpha transparency), outro para o vermelho (red), outro para o verde (green) e outro para o azul (blue). Seguramente, pode-se selecionar um bit (o menos signicativo) em cada byte do pixel para representar o bit a ser escondido sem

causar alteracões perceptíveis na imagem. Estas técnicas constituem a forma de mascaramento em imagens mais difícil de ser detectada pois podem inserir dados em pixels não sequenciais, tornando complexa a detecção (POPA, 1998) (PETITCOLAS; ANDERSON; KUHN, 1999) (WAYNER, 2002).

2.2.3. Filtragem e Mascaramento

As técnicas de esteganografia baseadas em filtragem e mascaramento são mais robustas que a insercão LSB. Estas geram estego-imagens imunes a compressão e recorte. No entanto, são técnicas mais propensas a detecção (WAYNER, 2002). Ao contrário da inserção no canal LSB, as técnicas de filtragem e mascaramento trabalham com modificacões nos bits mais significativos das imagens. As imagens de cobertura devem ser em tons de cinza porque estas técnicas não são eficazes em imagens coloridas (POPA, 1998). Isto deve-se ao fato de que modificacões em bits mais significativos de imagens em cores geram muitos artefatos tornando as informações mais propensas a detecção.

Estas técnicas são semelhantes a marca d'água visível em que valores de pixel em áreas mascaradas são aumentados ou diminuídos por um pouco de porcentagem. Reduzindo o incremento por um certo grau faz a marca invisível. No método de retalhos (*patchwork*), pares de remendos (*patches*) são selecionados pseudo-aleatoriamente. Os valores de pixel em cada par são aumentados por um valor constante pequeno em um remendo e diminuídos pela mesma quantia no outro.

2.2.4. Algoritmos e Transformações

As técnicas de esteganografia baseadas em algoritmos e transformações conseguem tirar proveito de um dos principais problemas da inserção no canal LSB que é a compressão. Para isso são utilizadas: a transformada de Fourier discreta, a transformada de cosseno discreta e a transformada Z (GONZALEZ; WOODS, 2002).

Sendo embutido no domínio de transformação, os dados escondidos residem em áreas mais robustas, espalhadas através da imagem inteira e fornecem melhor resistência contra processamento de sinal. Configuram-se como as mais sosticadas técnicas de mascaramento de informações conhecidas (POPA, 1998), embora sofisticacão nem sempre implique em maior robustez aos ataques de esteganálise. A inclusão de dados apresentados no domínio de transformação é amplamente usada para marca d'água robusta.

De forma geral, estas técnicas baseadas em algoritmos e transformações aplicam uma determinada transformação em blocos de 8x8 pixels na imagem. Em cada bloco, devem ser selecionados os coeficientes que são redundantes ou de menor importância. Posteriormente, estes coeficientes são utilizados para atribuir a mensagem a ser escondida em um processo em que cada coeficiente é substituído por um valor pré-determinado para o bit 0 ou 1 (POPA, 1998).

Para melhor entendimento do funcionamento destas técnicas, é explicada a seguir a transformada de cosseno discreta (DCT) que é muito utilizada nas compressões dos padrões JPEG e MPEG.

Transformada de Cosseno Discreta

A transformada de cosseno discreta (DCT - *Discrete Cosine Transform*) é uma transformada matemática baseada em cossenos, muito utilizada em processamento digital de imagens e compressão de dados. O valor da função da DCT de um vetor *p* de *pixels* de comprimento *n* é:

$$G_f = \frac{1}{2} C_f \sum_{t=0}^{n-1} p_t \cos\left(\frac{(2t+1)f\pi}{2n}\right), \tag{2.1}$$

$$onde: C_f = \begin{cases} \frac{1}{\sqrt{2}}, & f = 0 \\ 1 & f > 0 \end{cases} \text{para } f = 0, 1, ..., n-1.$$

A matriz dessa transformada é composta de vetores ortonormais, sendo por isso uma matriz de rotação. Na compressão de dados, esta transformada é muito utilizada pois transfere a maior parte da informação contida para os primeiros elementos do vetor, otimizando o armazenamento (para compressão sem perdas) e facilitando a quantização dos valores (para compressão com perdas), conforme mostrado na Figura 2.3.

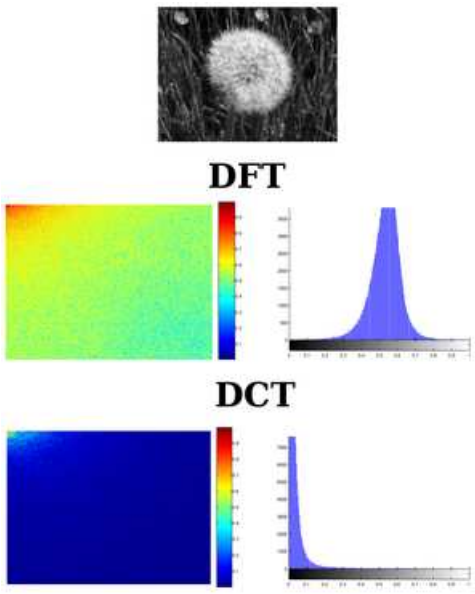


Figura 2.3: Comparação entre a Transformada de Fourier discreta e a DCT: pode se observar o acúmulo dos coeficientes mais significativos no canto superior direito da imagem da DCT, proporcionando melhor capacidade de compressão.

A recuperação dos dados transformados pode ser feita com a operação inversa, chamada de IDCT (*Inverse Discrete Cosine Transform*), que é dada pela fórmula:

$$p_t = \frac{1}{2} \sum_{j=0}^{n-1} C_f G_j \cos\left(\frac{(2t+1)j\pi}{2n}\right), \text{para } t = 0, 1, ..., n-1. \tag{2.2}$$

Em compressão de imagens e vídeos a maioria dos padrões usa a transformada discreta de cosseno do vetor *p* com o tamanho *n* = 8 (JPEG e MPEG).

Sabendo que os pixels de uma imagem tem correlação com seus vizinhos nas duas dimensões da imagem, e não apenas em uma dimensão, a DCT para ser usada na compressão de imagens também deve ser uma transformada bidimensional. A fórmula para uma matriz (ou seja uma imagem) *p* de tamanho *n* x *n* é:

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum y = 0^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right), \text{para } 0 \leq i, j \leq n-1; \quad (2.3)$$

$$onde\, C_f = \begin{cases} \frac{1}{\sqrt{2}}, & f = 0 \\ 1, & f > 0 \end{cases}.$$

Essa transformada pode ser considerada como uma rotação (ou duas rotações consecutivas, uma em cada dimensão), ou ainda como uma base ortogonal em um espaço vetorial de *n* dimensões. A recuperação dos dados transformados pode ser feita usando a transformação inversa, conhecida como IDCT bidimensional:

$$p_{xy} = \frac{1}{4} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \cos\left(\frac{(2x+1)i\pi}{2n}\right) \cos\left(\frac{(2y+1)j\pi}{2n}\right). \quad (2.4)$$

Analogamente à transformada unidimensional, a transformada bidimensional resulta em uma matriz onde os coeficientes mais significativos se acumulam no canto superior esquerdo (início da matriz) e os demais coeficientes são de pequeno valor podendo ser mais facilmente armazenados ou mesmo quantizados para proporcionar uma compressão com perdas.

Apesar de serem relativamente fáceis de implementar em qualquer linguagem de programação, a compressão de imagens demanda um grande poder de processamento e por isso precisa ser otimizada ao máximo. O uso da DCT em imagens grandes, apesar de apresentar ótimos resultados, exige um processamento muito grande. Por isso na prática a estratégia que se adota é dividir a imagem em blocos de tamanho menor (em geral de tamanho 8x8 pixels, como no JPEG), levando a uma primeira otimização:

- otimização 1 - a imagem a ser tratada deve ser dividida em blocos menores facilitando a computação das transformadas. Outra justificativa para esta abordagem é que, apesar de existir bastante correlação com os vizinhos próximos, existe pouca ou nenhuma correlação entre pontos distantes de uma mesma imagem. Os ganhos de processamento com esta abordagem suplantam em muito as perdas em termos de compressão.

O cálculo das funções de cosseno, por ser uma função transcendental, também exige bastante poder de processamento. Verificando a fórmula da DCT pode-se pré-calcular todos os valores de cosseno a serem utilizados e, depois disto, apenas realizar operações aritméticas de soma e multiplicação, o que leva à segunda otimização:

- otimização 2 - os cossenos utilizados devem ser pré-calculados e armazenados, realizando-se assim apenas operações aritméticas ao se calcular a fórmula da transformada.

Com um pouco de esforço algébrico, pode-se provar que a somatória dupla da fórmula da DCT bidimensional na Equação 3 corresponde ao produto matricial CPC^T, onde P é a matriz 8x8 representando o bloco de imagem a ser comprimido, C é a matriz definida por:

$$C_{ij} = \begin{cases} \frac{1}{\sqrt{8}}, & i = 0 \\ \frac{1}{2}\cos\left(\frac{(2j+1)i\pi}{16}\right), & i > 0 \end{cases} \tag{2.5}$$

e C^T é a sua transposta. Essa multiplicação matricial exige menor número de multiplicações e somas que a fórmula original, reduzindo ainda mais o tempo de execução da transformada. E isso leva à terceira otimização:

- otimização 3 - aplicação da transformada de cosseno discreta sob a forma matricial CPC^T para reduzir ainda mais o número de operações.

Uma última otimização é a utilização de aritmética de ponto fixo (número fixo de casas decimais). Esta técnica aproveita o fato de que muitos computadores executam as instruções de ponto fixo com mais rapidez do que as de ponto flutuante, acelerando assim o cálculo da transformada. Entretanto, esta técnica introduz uma quantização forçada, mas que no contexto da compressão de dados pode ser desprezada.

- otimização 4 - uso de aritmética de ponto fixo para aproveitar a maior velocidade desse tipo de cálculo na maioria dos computadores.

Ao aplicar a DCT, os coeficientes mais significativos se acumulam no início do vetor (ou matriz) dos dados, ficando o restante com valores muito pequenos e carregando pouca informação. Este tipo de distribuição já é suficiente para que uma técnica de redução de redundância (como os algoritmos LZ77, LZ78 ou LZW) ou uma codificação otimizada (como codificação de Huffman ou codificação aritmética) produzam melhores resultados do que na imagem ou nos dados originais. Entretanto, por se trabalhar sempre com uma precisão finita nas representações numéricas utilizadas, tem-se uma perda nos dados. Portanto, mesmo sem aplicar nenhuma forma de quantização, a compressão usando transformada de cosseno discreta é uma compressão com perdas.

Entretanto, a forma mais comum e que gera melhores resultados, é a aplicação de uma operação de quantização nos dados gerados pela transformada, e apenas o armazenamento dos dados quantizados. Essa quantização permite uma maior eficiência das técnicas de codificação e eliminação de redundância utilizada. Algumas formas de quantização normalmente utilizadas com a DCT são:

- eliminação dos componentes menos significativos - determina-se um patamar de valor ou mesmo de posição na matriz de resultados da transformada, e elimina-se ou substitui-se esses valores por 0;
- divisão inteira dos valores por um coeficiente de quantização fixo - assim pode-se usar menos dígitos, ou bits, para se representar os valores;
- divisão inteira por uma matriz de coeficientes de quantização - esta técnica é a empregada pela maioria dos padrões de compressão de dados, pois é mais flexível e permite que se ajuste a matriz a qualidade desejada da imagem.

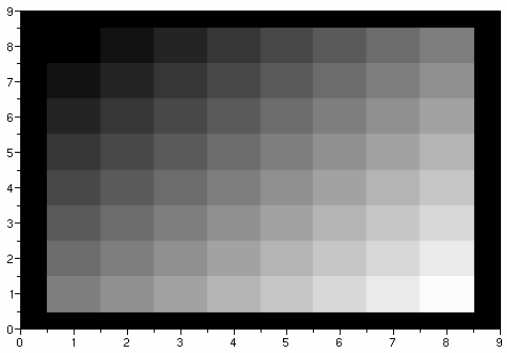
O padrão JPEG usa esta última técnica, e a tabela de coeficientes utilizada deve ser gravada junto com o arquivo comprimido da imagem. A escolha das matrizes no padrão JPEG pode ser da seguinte forma:

1. Uso das tabelas padronizadas de quantização fornecidas pelo padrão JPEG; ou
2. Uso de uma tabela de quantização Q personalizada, em geral calculada com uma fórmula simples que pode ser parametrizada para melhor ou pior qualidade de imagem. Uma fórmula bem comum é a seguinte, que usa um valor inteiro R como parâmetro:

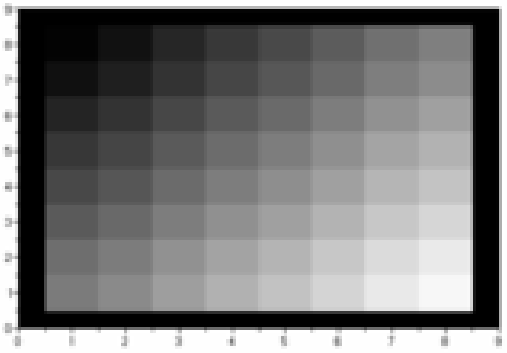
$$Q_{ij} = 1 + (i + j) \times R \tag{2.6}$$

Ainda no padrão JPEG, os coeficientes quantizados são separados (o coeficiente mais significativo de cada bloco 8x8 é separado dos demais para efeito de maior compressão) e comprimidos usando-se uma combinação de RLE (*Run Length Encoding*) e codificação de Huffman. O padrão prevê também a compressão através de uma variante das codificações aritméticas, chamada de codificação QM. Entretanto, a codificação QM, assim como a maioria das codificações aritméticas está protegida por patentes, e é preciso de uma licença do detentor das patentes para ser utilizada. Esta restrição das patentes fez com que a maioria dos compressores de JPEG utilize apenas a codificação de Huffmann, ignorando o uso do QM.

A Figura 2.4 apresenta alguns exemplos de imagens transformadas usando DCT (de tamanho 8x8 pixels, ampliadas para maior clareza), quantizadas com a tabela recomendada pelo padrão JPEG, e destransformadas para recompor a imagem descomprimida. Note que as imagens onde as transições de tons são mais suaves proporcionam uma melhor recomposição da imagem.

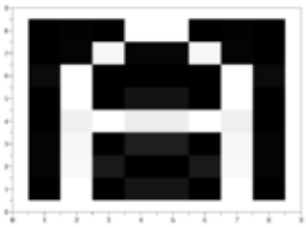


(a) Degradê cinza sem passar por DCT

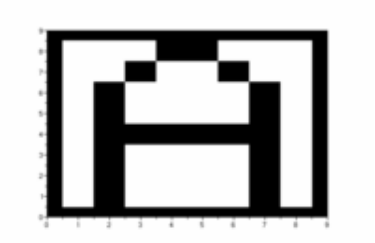


(b) Degradê cinza após DCT

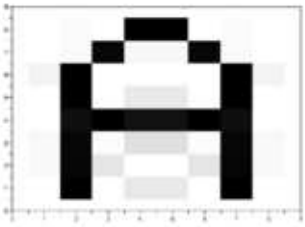
(c) Letra com fundo preto sem passar por DCT



(d) Letra com fundo preto após passar por DCT



(e) Letra com fundo branco sem passar por DCT



(f) Letra com fundo branco após passar por DCT

Figura 2.4: Efeito da DCT em imagens.

Essa característica de suavizar as bordas, que pode ser notada nas imagens da Figura 2.4, é o que faz o DCT ser amplamente utilizado em compressão de fotos, pois nesse tipo de imagem, a presença de bordas e mudanças abruptas é mais rara. Para a compressão de desenhos e textos escaneados, esta técnica não é tão boa pois "borra" ligeiramente as bordas das linhas retas, como pôde ser visto nos dois últimos conjuntos de imagens.

Para demonstrar a capacidade de compressão proporcionada, usa-se a matriz que gerou a imagem em degradê e mostra-se aqui todos os passos da compressão usando DCT. Primeiro tem-se a matriz original a seguir. Pode-se ver que essa matriz possui vários valores distintos, não alcançando bons resultados apenas com a eliminação das repetições:

$$\begin{pmatrix} 1. & 19. & 37. & 55. & 73. & 91. & 109. & 127. \\ 19. & 37. & 55. & 73. & 91. & 109. & 127. & 145. \\ 37. & 55. & 73. & 91. & 109. & 127. & 145. & 163. \\ 55. & 73. & 91. & 109. & 127. & 145. & 163. & 181. \\ 73. & 91. & 109. & 127. & 145. & 163. & 181. & 199. \\ 91. & 109. & 127. & 145. & 163. & 181. & 199. & 217 \\ 109. & 127. & 145. & 163. & 181. & 199. & 217 & 235. \\ 127. & 145. & 163. & 181. & 199. & 217 & 235. & 253. \end{pmatrix} \textbf{(1)}$$

Quando se aplica o DCT, tem-se a matriz seguinte. Esta matriz já tem vários valores zerados, que podem ser eliminados na compressão por alguma técnica de remoção de repetições, como RLE. A matriz a seguir está arredondada em duas casas decimais.

$$\begin{pmatrix} 1016. & -327.99 & 0. & -34.29 & 0. & -10.23 & 0. & -2.58 \\ -327.99 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -34.29 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -10.23 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -2.58 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{pmatrix} \textbf{(2)}$$

O passo seguinte é aplicar a quantização. Nesse momento pode-se ver que o número de posições zeradas aumenta ainda mais, e os valores restantes são todos relativamente pequenos, podendo ser representados em um arquivo com número menor de bits do que os valores maiores do arquivo original:

$$\begin{pmatrix} 63. & -30. & 0. & -2. & 0. & 0. & 0. & 0. \\ -27. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -2. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{pmatrix} \textbf{(3)}$$

Após a destransformação da matriz quantizada, usando IDCT, observa-se que os valores quase não mudaram em relação ao arquivo original, com a maior diferença entre eles na posição (2,8) que é de 6 (em um máximo de 256), ou seja, menos de 3%.

$$\begin{pmatrix} 4. & 18. & 39. & 57. & 74. & 93. & 113. & 128. \\ 17. & 32. & 52. & 71. & 88. & 106. & 127. & 141. \\ 37. & 52. & 72. & 91. & 107 & 126. & 146. & 161. \\ 56. & 70. & 91. & 109. & 126. & 144. & 165. & 179. \\ 73. & 87. & 108. & 126. & 143. & 161. & 182. & 196. \\ 91. & 106 & 126. & 145. & 161. & 180. & 200. & 215 \\ 111. & 125. & 146. & 164. & 181. & 200. & 220 & 235. \\ 124. & 139. & 159. & 178. & 195. & 213. & 234. & 248. \end{pmatrix} \textbf{(4)}$$

Para imagens onde as variações dos tons são graduais, a técnica de DCT mostra excelentes resultados, e por isso é adotada nos padrões mais usados hoje em dia.

O padrão MPEG usa para a compressão de áudio uma variante da DCT conhecida como MDCT (*Modified Discrete Cosine Transform*). Esta transformada é bastante parecida com a transformada de cosseno unidimensional, e sua fórmula é:

$$S_i = \sum_{k=0}^{n-1} x_k \cos\left(\frac{\pi}{2n}\left[2k+1+\frac{n}{2}\right](2i+1)\right), i = 0,1,..,\frac{n}{2}-1. \quad (2.7)$$

E a sua inversa, conhecida como IMDCT é dada por:

$$x_k = \sum_{i=0}^{n/2-1} S_i \cos\left(\frac{\pi}{2n}\left[2k+1+\frac{n}{2}\right](2i+1)\right), k = 0,1,..,n-1. \quad (2.8)$$

Maiores detalhes podem ser obtidos em (SALOMON, 2000).

2.2.5. Técnicas de Espalhamento de Espectro

Na técnica de espalhamento de espectro (como o espalhamento de freqüência), os dados escondidos são espalhados ao longo da imagem de cobertura. Uma stego-chave é usada para selecionar randomicamente os canais de freqüência. A *White Noise Storm* é uma ferramenta popular usando esta técnica. Em outra pesquisa (MARVEL; BONCELET; RETTER,

1999), dados embutidos como objeto a ser transmitido, a imagem de cobertura é visualizada como interferência em um *framework* de comunicação de cobertura. Os dados embutidos são primeiramente modulados com pseudo ruído e então a energia é espalhada sobre uma faixa de freqüência larga, alcançando somente um nível muito baixo de força de inclusão. Isto é valioso para alcançar a impercepitibilidade.

2.2.6. Técnicas de Esteganografia em Vídeo

Como já foi dito anteriormente, a esteganografia tira proveito de qualquer meio ou tipo de informação para esconder uma mensagem. No mundo digital atual, há grande quantidade de áudio e vídeo circulando principalmente pela Internet. E a esteganografia tira proveito deste vasto domínio de cobertura.

Quando informações são escondidas dentro de um vídeo, normalmente é usado o método da DCT. Sendo assim, esteganografia em vídeo é muito similar a esteganografia em imagens, exceto pelo fato de que as informações são escondidas em cada frame do arquivo de vídeo. Da mesma forma que nas imagens, quanto maior for a quantidade de informação a ser escondida no vídeo, maior será a possibilidade do método esteganográfico ser percebido.

Maiores detalhes sobre esteganografia em vídeos podem ser encontrados em (HARTUNG; GIROD, 1996) (LANGELAAR; LAGENDIJK; BIEMOND, 1997) (QIAO; NAHRSTEDT, 1998) (KALKER et al., 1999) (LINNARTZ; KALKER; HAITSMA, 1999).

2.2.7. Técnicas de Esteganografia em Áudio

Esconder imagens em sinais de áudio é algo desafiante, pois o sistema auditivo humano (SAH) pode trabalhar em uma faixa muito grande de freqüências. O SAH pode captar até um bilhão de potências diferentes de sinais (altura) e até mil freqüências de sinais distintas. A sensitividade a ruído também é muito apurada. Uma perturbação em um arquivo de som pode ser detectada tão baixa quanto uma em 10 mihões de partes ou 80 dB em um ambiente comum. Apesar de ser tão poderoso para captar sinais e freqüências, o SAH não consegue fazer diferenciação de tudo que recebe. Sendo assim, sons mais altos tendem a mascarar sons mais baixos. Além disso, o SAH não consegue perceber um sinal em fase absoluta, somente em fases relativas. Também existem algumas distorções do ambiente muito comuns que são simplesmente ignoradas pelo ouvido na maioria dos casos.

As técnicas de esteganografia exploram muitas destas “vulnerabilidades” do ouvido humano, porém sempre têm que levar em conta a extrema sensibilidade do SAH.

Para se desenvolver um método de esteganografia em áudio, uma das primeiras considerações a serem feitas é o ambiente onde o som trafegará entre a origem e o destino. Há pelo menos dois aspectos que devem ser considerados: a representação digital do sinal que será usado e o caminho de transmissão do sinal.

Quanto à representação digital, existem dois parâmetros críticos para a maioria das representações de áudio: método de quantização da amostra e taxa de amostragem temporal. Um dos formatos mais populares para representar amostras de áudio digital com alta qualidade é uma quantização linear de 16 bits chamada Windows Audio-Visual

(WAV) e Audio Interchange File Format (AIFF). Um outro formato popular para áudio de menor qualidade é a escala logarítmica de 8 bits $\mu - law$. Estes métodos de quantização introduzem uma distorção no sinal que é mais evidente no caso da quantização de 8 bits $\mu - law$.

Taxas de amostragem típicas para áudio incluem 8kHz, 9,6 kHz, 10 kHz, 12 kHz, 16 kHz, 22,05 kHz e 44,1 kHz. As taxas de amostragem impactam na esteganografia a medida que impõem uma barreira para a porção usável do espectro de freqüências. Não é possível, por exemplo, introduzir modificações que têm componentes de freqüência acima de 4 kHz se o sinal foi amostrado a uma freqüência de 8 kHz.

A última representação a ser considerada é a que produz perdas através do uso de algoritmos de compressão, tal como o MPEG-AUDIO. Estas representações modificam drastricamente o sinal, preservando somente as caracterísiticas que o ouvido humano pode perceber trabalhando com um modelo psico-acústico. Isso quer dizer que o som resultante será similar ao original, mesmo que o sinal resultante seja totalmente diferente.

Existem muitos meios de transmissão pelos quais um sinal pode passar no caminho do codificador até o decodificador. A primeira classe de meios de transmissão que pode ser considerada é um ambiente digital fim a fim. Neste ambiente, o arquivo de som é copiado de uma máquina para outra e não é modificado. Como resultado, a amostra é exatamente a mesma, tanto no codificador quanto no decodificador. Esta classe é a que menos impõe barreiras aos métodos esteganográficos.

A próxima classe de meios de transmissão é quando um sinal é re-amostrado para uma taxa de amostragem maior ou menor que a original, mas permanece digital. Esta transformação preserva a magnitude absoluta e a fase da maioria dos sinais, mas muda as características temporais do mesmo.

A terceira classe é a que apresenta um sinal que é "tocado" dentro de um dispositivo analógico, transmitido em uma linha analógica razoavelmente sem ruídos e depois re-amostrado (digital-analógico-digital). Neste caso não são preservados a magnitude do sinal, a quantização inicial e a taxa de amostragem. Somente a fase do sinal é preservada.

O último caso é quando um sinal é transmitido pelo ar ("tocado") e depois sofre nova amostragem com um microfone. O sinal estará possivelmente sujeito a modificações não lineares, resultando em mudanças de fase, amplitude, ecos e mudança de componentes.

Sendo assim, a representação do sinal e o caminho de transmissão devem ser considerados na escolha de um método de esteganografia. A taxa de dados é muito dependente da taxa de amostragem e do tipo de som que está sendo codificado. Um valor típico de taxa é 16 bps, mas este valor pode variar de 2 bps a 128 bps.

Codificação *Low-bit*

A codificação no bit menos significativo é a maneira mais simples de embutir dados dentro de outra estrutura de dados. Através da substituição do bit menos significativo de cada amostra por um codificador binário, é possível codificar uma grande quantidade de dados

em um sinal de áudio. De maneira ideal, a capacidade do canal deve ser de pelo menos 1kb por segundo (kbps) por 1kHz, isso é, em um canal sem ruído, a taxa de transmissão será de 8kbps em uma amostra de 8kHz e 44kbps em uma amostra de 44kHz. Como conseqüência desta grande capacidade, ruídos audíveis são sempre introduzidos. O impacto destes ruídos diferem de acordo com o conteúdo do sinal. Em um *stream* de áudio de uma partida de futebol, o barulho da torcida ao fundo mascarará o ruído introduzido pela técnica de codificação *low-bit*, enquanto que em um *stream* de música clássica, o ruído seria audível.

A maior desvantagem deste método é a sua baixa imunidade a manipulação. A informação codificada pode ser destruída pelo ruído do canal, na re-amostragem, entre outros, a menos que esta informação tenha sido codificada utilizando técnicas de redundância. Para serem robustas estas técnicas reduzem a taxa de transmissão de dados normalmente pela metade. Na prática, este método deve ser utilizado somente em ambientes de transmissão digitais (digital-digital).

Codificação em Fase

O método de codificação em fase trabalha substituindo a fase de um segmento inicial de áudio por uma fase de referência que representa os dados a serem escondidos. A fase dos segmentos subseqüentes é ajustada para preservar a fase relativa entre os segmentos 2.5.

A codificação em fase é um dos mais efetivos métodos de codificação em termos de sinal percebido quando comparado com a percepção do ruído. Quando a relação de fase entre cada componente de freqüência é mudada muito drasticamente, uma dispersão de fase será notada. Entretanto se as modificações das fases forem pequenas, uma codificação inaudível é obtida. A codificação em fase trabalha com o fato de que os componentes de fase de um som não são tão perceptíveis pelo ouvido humano quanto é o ruído.

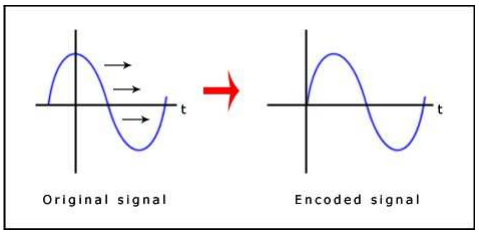


Figura 2.5: Sinal original versus sinal codificado.

Para esconder dados usando a codificação em fase, o seguinte procedimento deve ser seguido:

- o som original é quebrado em segmentos menores, os quais devem possuir tamanho igual à mensagem a ser escondida;
- uma Transformada de Fourier Discreta deve ser aplicada em cada segmento para criar uma matriz de fases e uma outra de magnitude do sinal;

- calcular as diferenças de fases entre os segmentos adjacentes;
- as mudanças de fase entre segmentos consecutivos são facilmente detectáveis. A fase absoluta dos segmentos pode mudar, mas as diferenças relativas de fase entre segmentos adjacentes deve ser preservada. Por isso a mensagem a ser escondida deve ser inserida somente no vetor de fase do primeiro segmento da seguinte forma:
  - $\pi/2$ se o bit a ser representado for 0;
  - $-\pi/2$ se o bit a ser representado for 1.
- uma nova matriz de fases é criada utilizando a fase do primeiro segmento e a matriz de fase original;
- utilizando a nova matriz de fase e a matriz original de magnitude, o som é reconstruído através da DFT inversa para depois concatenar os segmentos que formarão o som a ser transmitido.

Para a decodificação e a extração da mensagem secreta, o receptor deve conhecer o tamanho dos segmentos. O receptor pode então usar a DFT para obter as fases e extrair a informação. Uma desvantagem associada com a codificação em fase é uma baixa taxa de transmissão de dados pelo fato de que a mensagem secreta é codificada somente no sinal do primeiro segmento. Este problema pode ser resolvido aumentando o tamanho do segmento. Entretanto isso pode mudar muito a relação entre cada componente de freqüência, tornando a técnica fácil de ser detectada. Como resultado, o método de codificação em fase deve ser usado somente quando uma pequena quantidade de dados precisa ser escondida.

Spread Spectrum

No contexto de esteganografia em áudio, o *spread spectrum* (SS) tenta espalhar informações secretas sobre o espectro de freqüências de áudio tanto quanto possível. A codificação usando SS é análoga à LSB que randomicamente espalha os bits da mensagem sobre todo o arquivo de som. Entretanto, diferentemente da codificação LSB, o SS espalha a mensagem secreta sobre o espectro de freqüências do arquivo de som utilizando um código que é independente do sinal. Assim, o sinal resultante ocupa uma banda superior à utilizada para a transmissão do sinal original.

Duas versões de SS podem ser utilizadas na esteganografia: *Direct-Sequence Spread Spectrum* (DSSS) e *Frequency Hopping Spread Spectrum* (FHSS). No DSSS, a mensagem secreta é espalhada utilizando uma chave chamada *chip rate* e depois modulada com um sinal pseudo-randômico. Então a mensagem modulada é misturada com o sinal de cobertura. Já no FHSS, o espectro de freqüência do arquivo de áudio é alterado de tal forma que a mensagem seja codificada segundo um padrão de saltos entre as freqüências do espectro (PETITCOLAS; KATZENBEISSER, 1999).

O método utilizando SS tem bom potencial e é melhor em algumas circunstâncias que o LSB e a técnica de codificação em fase, pois oferece taxa de transmissão moderada

enquanto mantém alto nível de robustez contra técnicas de remoção. Entretanto, o método SS tem a desvantagem de introduzir ruído no som de cobertura, assim como a abordagem LSB.

Escondendo Informações com Eco

Nas técnicas de esteganografia utilizando eco, a informação é escondida em um arquivo de som através da introdução de um eco. Assim como o método de *spread spectrum*, este método também apresenta a vantagem de permitir uma maior taxa de transmissão e robustez superior quando comparado com outros métodos indutores de ruído.

Para esconder os dados de maneira eficaz, variam-se três parâmetros do sinal de eco: amplitude, taxa de deterioração e variação do sinal original (*offset*). Os três parâmetros são configurados abaixo dos limites que o ouvido humano pode perceber facilmente. O *offset* é utilizado para representar a mensagem binária codificada. O codificador utiliza dois valores de tempo de atraso: um para representar o bit 1 (*offset*) e outro para representar o bit 0 (*offset* mais delta), conforme visto na Figura 2.6

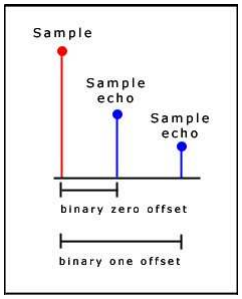


Figura 2.6: Codificando o eco.

Se um eco do sinal original for produzido, somente um bit de informação será codificado. Por isso, o sinal original é quebrado em blocos antes do processo de codificação começar. Uma vez que o processo de codificação é completado, os blocos são concatenados novamente. A cada bloco é assinalado o valor “1” ou “0” baseado na mensagem que será transmitida.

Ao utilizar esta implementação de esteganografia em eco, o processo pode resultar em um sinal que possui uma mistura de ecos, acarretando no aumento do risco de detecção. Uma segunda implementação pode resolver este problema. Primeiro um sinal de eco é criado a partir do som original inteiro usando o valor binário 0 do *offset*. Então um segundo sinal de eco é criado utilizando o sinal original inteiro agora utilizando o valor de *offset* binário 1. Desta forma, o primeiro eco somente contém zeros e o segundo somente contém valores um. Para efetuar a junção, dois *mixers* de sinais são utilizados. O *mixer* tem valor 0 ou 1, dependendo do bit que deve ser codificado. Como exemplo, será codificada a palavra “HEY” que apresenta os dois sinais, conforme Figura 2.7.

O sinal de eco “1” é então multiplicado pelo “1” do mixer e o sinal de eco “0” é multiplicado pelo “0” do mixer. Então os dois resultados são adicionados para obter o sinal final, que é menos abrupto do que o obtido usando o primeiro método.

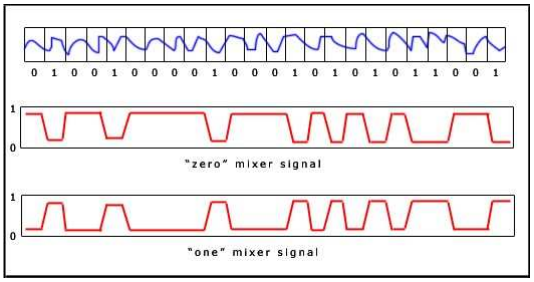


Figura 2.7: Codificando com mixer.

Para extrair a mensagem secreta do stego-sinal, o receptor deve quebrar o sinal na mesma seqüência do bloco usada durante o processo de codificação. Então, a função de autocorrelação do sinal (esta função é uma transformada de Fourier no espectro de freqüência do sinal) pode ser usada para decodificar a mensagem, pois revela um ponto em cada *offset* do tempo do eco, permitindo que a mensagem seja reconstruída.

Através da utilização dos métodos descritos é possível codificar e decodificar informações na forma de bits dentro de um fluxo de áudio com alterações mínimas do som original em uma taxa aproximada de 16 bps. Estas alterações mínimas são as que, na média, o ouvido humano não pode diferenciar entre o sinal original e o sinal alterado.

Outros trabalhos relacionados à esteganografia em áudio podem ser encontrados em (BONEY; TEWFIK; HAMDY, 1996) (BASSIA; PITAS, 1998) (PRANDONI; VETTERLI, 1998) (SWANSON; ZHU; TEWFIK, 1999) (SU et al., 1999) (SWANSON et al., 1998) (LU; LIAO; CHEN, 2000) (LI; YU, 2000) (KIM, 2000).

2.2.8. Técnicas de Esteganálise

Grande parte das técnicas de esteganografia possuem falhas ou inserem padrões que podem ser detectados. Algumas vezes, basta um agressor fazer um exame mais detalhado destes padrões gerados para descobrir que há mensagens escondidas. Outras vezes, o processo de mascaramento de informações é mais robusto e as tentativas de detectar ou mesmo recuperar ilicitamente as mensagens podem ser frustradas. A pesquisa de métodos para descobrir se há alguma mensagem escondida por esteganografia é chamada de **esteganálise**.

Recuperar os dados escondidos está além da capacidade da maioria dos testes atuais, uma vez que muitos algoritmos de mascaramento utilizam geradores aleatórios muito seguros para esconder a informação durante o processo de mascaramento. Muitas vezes, os bits são espalhados pelo objeto de cobertura. Desta forma, os melhores algoritmos de esteganálise podem não ser capazes de dizer onde está a informação, mas devem dizer se há dados escondidos.

Tipos de Ataques

Existem diversas abordagens para se detectar a presença de conteúdo escondido em imagens digitais. Estas abordagens podem ser divididas em três tipos (ROCHA, 2006): ataques

aurais, estruturais e estatísticos.

- ataques aurais - estes ataques consistem em retirar as partes significativas da imagem como um meio de facilitar aos olhos humanos a busca por anomalias na imagem. Um teste comum é mostrar os bits menos significativos da imagem. Câmeras, scanners e outros dispositivos sempre deixam alguns padrões nos bits menos significativos.

- ataques Estruturais - a estrutura do arquivo de dados algumas vezes muda assim que outra mensagem é inserida. Nesses casos, um sistema capaz de analisar padrões estruturais seria capaz de descobrir a mensagem escondida. Por exemplo, se mensagens são escondidas em imagens indexadas (baseadas em paletas de cores), pode ser necessário usar diferentes versões de paletas. Este tipo de atitude muda as características estruturais da imagem de cobertura, logo as chances de detecção da presença de uma mensagem escondida aumentam (WAYNER, 2002).

- ataques estatísticos - os padrões dos pixels e seus bits menos significativos freqüentemente revelam a existência de uma mensagem secreta nos perfis estatísticos (WAYNER, 2002; WESTFELD; PFITZMANN, 2000; PROVOS; HONEYMAN, 2003). Os novos dados não têm os mesmos perfis esperados. Muitos dos estudos de Matemática e Estatística têm por objetivo classificar se um dado fenômeno ocorre ao acaso. Cientistas usam estas ferramentas para determinar se suas teorias explicam bem tal fenômeno. Estas técnicas estatísticas também podem ser usadas para determinar se uma dada imagem e/ou som possui alguma mensagem escondida. Na maioria das vezes, os dados escondidos são mais aleatórios que os dados que foram substituídos no processo de mascaramento ou inserem padrões que alteram as propriedades estatísticas inerentes do objeto de cobertura (WESTFELD; PFITZMANN, 2000; PROVOS; HONEYMAN, 2003; WAYNER, 2002).

Principais Técnicas de Esteganálise

A seguir, são apresentadas algumas das principais técnicas de esteganálise baseadas em ataques estatísticos existentes.

1. Esteganálise por teste do χ^2 (*Chi-Square Test Approach*).

   O teste Chi-quadrado permite verificar igualdade (semelhança) entre categorias discretas e mutamente exclusivas (por exemplo, diferenças de comportamento entre homens e mulheres). Cada indivíduo ou item deve pertencer a uma e somente uma categoria.

   As seguintes suposições precisam ser satisfeitas:

   (a) os dois grupos são independentes;

   (b) os itens de cada grupo são selecionados aleatoriamente;

   (c) as observações devem ser freqüências ou contagens;

(d) cada observação pertence a uma e somente uma categoria;

(e) a amostra deve ser relativamente grande (pelo menos 5 observações em cada célula e no caso de poucos grupos (2 x 2) pelo menos 10);

A hipótese *H0* é que não existe diferença entre as freqüências (contagens) dos grupos. A hipótese alternativa é que existe diferença. Para se testar as hipótese é preciso testar se existe diferença significativa entre as freqüências observadas e as esperadas em cada extrato.

Exemplo: Deseja-se saber se existe diferença na percepção de homens e mulheres em relação a uma afirmativa feita. As categorias são homens e mulheres, e número total de mulheres é diferente do número total de homens. Cada item pertence a uma e somente uma destas categorias. Da mesma maneira, cada indivíduo poderá responder somente de uma forma. O resultado deve ser comparado com o que seria obtido se não houvesse diferença entre os grupos. Para ilustrar, supõe-se 99 homens e 99 mulheres na amostra. Neste caso, se os grupos se comportassem da mesma forma e respondessem igualmente para cada situação o resultado seria 33 pessoas em cada célula.

Em geral os grupos não são igualmente distribuídos. O valor esperado de cada célula é uma proporção do valor total. Um caso real é apresentado na Tabela 2.1:

Tabela 2.1: Tabela exemplo para o teste χ^2.

	Homens	Mulheres	Total
Concorda	58	35	93
Neutro	11	25	36
Não concorda	10	23	33
Total	79	83	162

Os valores esperados para cada célula são obtidos multiplicando o percentual da coluna pelo total da linha, isto é, total da linha x (total coluna / total). Por exemplo: 45,35 = 93 x 79/162, conforme Tabela 2.2.

O valor de chi-quadrado para cada célula é a diferença ao quadrado entre o valor esperado e o valor medido dividido pelo valor esperado, conforme formula a seguir.

$$\chi^2 = \frac{(ValorEsperado - ValorMedido)^2}{ValorEsperado} \tag{2.9}$$

O chi total é a soma dos valores de cada célula. O valor de χ^2 calculado deve ser comparado com o valor de chi tabelado, quanto maior o valor de chi calculado, maior a diferença. Para obter o valor de chi tabelado (tabela de distribuição χ^2) deve-se escolher o valor do nível de significância(alfa) adequado para a situação.

Em esteganografia, as funções de cobertura de alguns softwares, por exemplo o Ezstego (EZSTEGO, 2007) reescrevem os bits menos significativos dos bytes sorteados para tal fim guardando seus índices. Isso gera valores modificados de bytes

Tabela 2.2: Cálculo do χ^2.

Esperado				
		Homens	Mulheres	Total
	Concorda	45,35185	47,64815	93
	Neutro	17,55556	18,44444	36
	Não concorda	16,09259	16,90741	33
	Total	79	83	162
Chi				
		3,527434	3,357437	
		2,447961	2,329987	
		2,306632	2,195469	
Chi Tabelado =	2			

que só diferem, quando diferem, no último bit. Este par de valores (iniciais e transformados) será chamado de PoVs (*Pair of Values*). Se os bits usados para escrever no bit menos significativo são igualmente distribuídos, a freqüência dos valores de cada PoV se torna igual. A idéia dos ataques estatísticos é comparar uma distribuição de freqüência teórica esperada m um histograma com algumas distribuições observadas em possíveis imagens que podem ter sido modificadas. A distribuição de freqüência teórica é obtida com o chi tabelado usando o nível de significância adequado (alfa).

Um ponto crítico é como obter a distribuição de freqüência teórica. Esta freqüência não deve ser derivada da amostra que está sendo analisada pois a amostra pode ter sido modificada por esteganografia. O problema é que na maioria dos casos não se tem a amostra original para comparar. Na amostra original, a freqüência teórica esperada é a média aritmética das duas freqüências de um PoV. Isso porque a função mascaramento do método esteganográfico sobrescreve os bits menos significativos e isso não muda a soma destas duas freqüências (freqüência de um PoV). A contagem dos valores de freqüência pares é transferida para o valor ímpar correspondente de freqüência em cada PoV e vice-versa. Este fato permite obter a distribuição de freqüência esperada da amostra analisada, não necessitando da original para o teste.

2. Análise RS.

   Apresentada por Jessica Fridrich (FRIDRICH; GOLJAN, 2002), esta técnica consiste na análise das inter-relações entre os planos de cores presente nas imagens analisadas. A classificação é feita pontualmente, sem utilização de treinamento e é dependente do contexto da imagem analisada.

   Este é um dos métodos de detecção mais robustos disponíveis. Para análise podem ser utilizadas imagens coloridas ou em tons de cinza. Não existe distinção na profundidade de cores na imagem analisada, isto pode ser válido tanto para imagens de 8 bpp (bits por pixel) quanto para imagens de 32 bpp.

   As definições feitas por Rocha (ROCHA, 2006) ressaltam os seguintes aspectos:

   “Seja IMG a imagem testada. IMG possui MxN *pixels* e cada *pixel* tem os seus

valores dados por um conjunto P. Como exemplo, para uma imagem de 8 bpp, tem-se P = 0,....255. Então, divide-se IMG em grupos de *pixels* em disjuntos G de n *pixels* adjacentes, onde G = $(x_1,...,x_n)$.

Como exemplo, pode-se escolher grupos de n = 4 *pixels* adjacentes. Feito isso, define-se uma função de discriminação f responsável por atribuir um número real $f(x_1,...,x_n)$ para cada grupo de *píxels* G = $(x_1,...,x_n)$. Quanto mais aleatório for o grupo, maior o valor da função de discriminação, dada por $f(x_1,x_2,...,x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|$.

Finalmente, define-se uma operação inversível F sobre P chamada *flipping*. Por *flipping* entende-se a permutação dos níveis de cores e consiste em 2 ciclos. Assim, F tem a propriedade que F^2 = Identidade ou F(F(x)) = x para todo $x \in P$. A permutação $F_1 : 0 \Leftrightarrow 1, 2 \Leftrightarrow 3, ..., 254 \Leftrightarrow 255$ corresponde a negar o LSB de cada nível de cor. Adicionalmente pode-se definir uma função de *shifting* (deslocamento) $F_{-1} : -1 \leftrightarrow 0, 1 \leftrightarrow 2, ..., 255 \leftrightarrow 256$, ou $F_{-1}(x)$ = $F_1(x+1)$-1 $\forall x$.

Para completar, define-se F_0 como sendo a permutação de identidade F(x)= x $\forall x \in P$. Estas operações são utilizadas para classificar os grupos de *pixels* em três categorias diferentes R, S e U:

- grupos regulares: $G \in R \Leftrightarrow f(F(G)) > f(G)$;
- grupos singulares: $G \in S \Leftrightarrow f(F(G)) < f(G)$;
- grupos não-usáveis: $G \in U \Leftrightarrow f(F(G)) = f(G)$.

Nestas expressões, F(G) significa que a função de *flipping* F foi aplicada para os componentes do vetor G = $(x_1,...,x_n)$. Caso seja desejado aplicar diferentes *flippings* em diferentes *pixels*, deve-se usar uma máscara M que irá denotar quais os *pixels* deverão sofrer alterações. A máscara M é uma n-tupla com valores -1, 0, 1. Define-se o grupo alterado GA como: GA = $(F_{M(1)}(x1), F_{M(2)}(x2), ..., F_{M(n)}(xn))$.

O objetivo da função F é perturbar os *pixels* de uma forma pouco significativa tal como aconteceria no processo de mascaramento de uma mensagem."

O método também proposto por Rocha (ROCHA, 2006) baseado em (FRIDRICH; GOLJAN, 2002) é descrito a seguir:

"Seja R_M a percentagem do número de grupos regulares em relação ao total de grupos para a máscara M. De forma similar, S_M irá notar o número relativo de grupos singulares. Tem-se que $R_M + S_M \leq 1$ *e* $R_{-M} + S_{-M} \leq 1$, para a máscara negativa. A hipótese estatística para o método é que, em imagens típicas, o valor esperado de R_M é aproximadamente igual ao de R_{-M} e o mesmo é verdade para S_M *e* S_{-M}. A equação definida em $R_M \cong R_{-M}$ *e* $S_M \cong S_{-M}$, foi empiricamente comprovada (FRIDRICH; GOLJAN, 2002). A randomização do plano LSB força a diferença entre R_M *e* S_M para zero à medida que o tamanho m da mensagem escondida cresce. Depois de alterar os LSBs de 50 por cento dos *pixels* (é o que acontece quando se esconde uma mensagem aleatória em todos os *pixels*), obtém-se $R_M \cong S_M$, isto é o mesmo que dizer que a capacidade de mascaramento no plano LSB agora é zero. O fato surpreendente é que um efeito contrário acontece com R_{-M} *e* S_{-M}, sua diferença aumenta proporcionalmente ao tamanho da mensagem escondida."

Desta forma, ao analisar a imagem testada, esta provavelmente estará escondendo uma mensagem se:

- Condição 1: $R_M - R_{-M}$ = i e i é muito grande;
- Condição 2: $R_M - S_M$ = k e k é muito grande.

Valores muito grandes para i acontecem quando $i \geq 2,5\%$ do total de grupos. Valores muito grandes para k acontecem quando $k \geq 25\%$ do total de grupos. Um mascaramento detectável ocorre toda vez que a primeira condição é verdadeira. Caso apenas a segunda condição seja verdadeira, há apenas uma suspeita de que houve um mascaramento (FRIDRICH; GOLJAN, 2002).

3. Métricas de qualidade de imagens (*Image Quality Metrics*).

   Métricas de qualidade de imagem são utilizadas, de forma geral, na avaliação de codificação de artefatos, predição de desempenho de algoritmos de visão computacional, perda de qualidade devido a inadequabilidade de algum sensor, entre outras aplicações.

   Nesta abordagem proposta por Ismail Avcibas (AVCIBAS; MEMON; SANKUR, 2001), essas mesmas métricas são utilizadas para construir um discriminador de imagens de cobertura (sem conteúdo escondido) de estego-imagens (com conteúdo escondido) através da utilização de regressão multi-variada. A classificação é feita por um discriminante linear após um certo treinamento (estabilização dos coeficientes da regressão multi-variada).

4. Métricas de tons contínuos e análise de pares de amostragem (*Continuous Tone Metrics and Sample Pair Analysis*).

   Proposta por Sorina Dumitrescu (DUMITRESCU; WU, 2002), esta abordagem consiste em analisar as relações de identidade estatística existentes sobre alguns conjuntos de *pixels* considerados. As identidades observadas são muito sensíveis ao mascaramento LSB e as mudanças nestas identidades podem indicar a presença de conteúdo escondido.

2.3. Aplicações

Em atividades militares, a descoberta de comunicações secretas pode levar a um ataque imediato do inimigo. Mesmo com a criptografia, a simples detecção do sinal é fatal pois descobre-se não somente a existência de inimigos como também a sua posição. Unindo o conceito de ocultamento de informação com técnicas como modulação em espalhamento de espectro torna-se mais difícil de os sinais serem detectados ou embaralhados pelo inimigo.

Várias técnicas relacionadas a ocultamento de informação levam em consideração sistemas com níveis de segurança. Em uma rede de computadores militares existem vários níveis de segurança. Um vírus ou um programa malicioso se propaga dentro do sistema passando de níveis de segurança inferiores para os superiores. Uma vez que alcança seu objetivo, tenta passar informações sigilosas para setores de nível de segurança

menores. Para isso, ele se utiliza de técnicas de ocultamento para esconder informações confidenciais em arquivos comuns de maneira que o sistema lhe permita ultrapassar níveis de segurança diferentes.

Existem situações onde se deseja enviar uma mensagem sem que seja possível descobrir quem a enviou. Geralmente, esse tipo de situação é mais uma característica de atividades ilegais onde os criminosos envolvidos não desejam ser descobertos se sua mensagem for rastreada. Entretanto, essa situação também tem aplicações em atividades legais onde se deseja que a privacidade do remetente seja mantida. Alguns exemplos dessas situações são: registros médicos ou votações online.

Um tema importante a ser considerado pelo criador das técnicas de ocultamento de informação é a ética. Assim como os conhecimentos apresentados podem ser usados para garantir privacidade de dados médicos, votações ou prover serviços online com segurança, algumas pessoas podem encontrar meios de se aproveitar das vantagens dessa 'comunicação invisível e não rastreável' para executar ações ilícitas como difamações, chantagens ou seqüestros. É um dever dos desenvolvedores de sistemas de ocultamento de informação prestar atenção aos abusos que podem ocorrer.

Existe também grandes aplicações na área da indústria médica no que diz respeito a imagens médicas. Normalmente, é usada uma forma de comunicação padrão chamada DICOM (*digital imaging and communications in medicine*) que separa a imagem das informações relativas ao paciente e ao exame como o nome, data e o médico. Em alguns casos, a ligação entre os dados e a imagem é perdida. Então, se as informações fossem ocultadas dentro da própria imagem, não haveria risco de a imagem se separar dos dados. Ainda não existem pesquisas aprofundadas sobre o efeito que tais inserções de dados na imagem poderiam causar alteração na precisão do diagnóstico. Estudos recentes na área de compressão de imagens médicas revelam que tais procedimentos não atrapalham, o que pode indicar uma certa robustez do diagnóstico a pequenas alterações como as causadas pelas técnicas de ocultamento de informação (FILHO et al., 2005).

Em alguns casos, se deseja monitorar um dado arquivo com direitos autorais que está sendo distribuído na Internet, por exemplo. Para isso, utiliza-se um programa robô que procura em *sites* a divulgação desses arquivos. Ele baixa os arquivos, tenta retirar qualquer informação que possa estar escondida e compara com a informação do arquivo original. Caso as informações sejam compatíveis, sabe-se que o arquivo está sendo distribuído de maneira ilegal. O mesmo pode ser feito com músicas tocadas em transmissões via rádio. O programa procura no sinal do rádio marcas inseridas propositalmente nas músicas a serem protegidas. Se em um dado momento a marca é inteiramente decodificada do sinal, sabe-se que a estação de rádio investigada tocou a música sem autorização.

Pode-se também inserir pedaços de informações dentro dos dados que estão sendo transmitidos para que o público que a receba possa usar. Como exemplo, pode-se ter informações de um dado produto anunciado por uma rádio onde o cliente, com um simples apertar de botão, pode descobrir o preço, local de venda mais próximo ou fabricante. Essas informações são enviadas sem a necessidade de se usar outra banda para transmissão pois ela é inserida no próprio sinal de rádio sem prejudicar a qualidade do mesmo.

Outra aplicação seria inserir uma forma de indexação de músicas a serem arma-

zenadas no banco de dados de uma estação de rádio para que elas sejam acessadas de maneira mais fácil. Pode-se inserir também dados da transmissão como país de origem, autor e produtora.

Atualmente a esteganografia tem sido também explorada em ramos de sistemas de detecção de intrusão (SIEFFERT et al., 2004) e em sistemas de arquivos (HIROHISA, 2007).

Outras aplicações de esteganografia incluem as técnicas de autenticação, criptografia e rastreamento de documentos, que por serem utilizadas normalmente em conjunto com a técnica de marca d'água, são mencionadas a seguir.

2.3.1. Marcas D'Água

O grande crescimento dos sistemas de multimídia interligados pela rede de computadores nos últimos anos apresenta um enorme desafio nos aspectos tais como propriedade, integridade e autenticação dos dados digitais (áudio, vídeo e imagens estáticas). Para enfrentar tal desafio, o conceito de marca d'água digital foi definido.

Uma marca d'água é um sinal portador de informação, visualmente imperceptível, embutido em uma imagem digital. A imagem que contém uma marca é dita imagem marcada ou hospedeira. Apesar de muitas técnicas de marca d'água poderem ser aplicadas diretamente para diferentes tipos de dados digitais, as mídias mais utilizadas são as imagens estáticas.

Existe uma certa confusão entre as marcas d'água imperceptíveis e as visíveis utilizadas em cédulas de dinheiro, por exemplo. As visíveis são usadas em imagens e aparecem sobrepostas sem prejudicar muito a percepção da mesma. São usadas geralmente para que se possa expor imagens em locais públicos como páginas na Internet sem o risco de alguém copiá-la e usá-la comercialmente, pois é difícil remover a modificação sem destruir a obra original. É possível também inserir digitalmente marcas visíveis em vídeo e até audíveis em música.

Marcas Robustas e Frágeis

As marcas d'água digitais são classificadas, de acordo com a dificuldade em removê-las, em robustas, frágeis e semifrágeis. Esta classificação também normalmente determina a finalidade para a qual a marca será utilizada.

As marcas robustas são projetadas para resistirem a maioria dos procedimentos de manipulação de imagens. A informação embutida em uma imagem através de uma marca robusta poderia ser extraída mesmo que a imagem hospedeira sofresse rotação, mudança de escala, mudança de brilho/contraste, compactação com perdas com diferentes níveis de compressão, corte das bordas (*cropping*), etc. Uma boa marca d'água robusta deveria ser impossível de ser removida, a não ser que a qualidade da imagem resultante deteriore a ponto de destruir o seu conteúdo visual. Isto é, a correlação entre uma imagem marcada e a marca robusta nela inserida deveria permanecer detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuar visualmente reconhecível e identificável como a imagem original. Por esse motivo, as marcas d'água robustas são normalmente utilizadas para a verificação da propriedade (*copyright*) das

imagens. Apesar de muitas pesquisas, parece que ainda não foi possível obter uma marca d'água robusta realmente segura.

As marcas frágeis são facilmente removíveis e corrompidas por qualquer processamento na imagem. Este tipo de marca d'água é útil para checar a integridade e a autenticidade da imagem, pois possibilita detectar alterações na imagem. Em outras palavras, uma marca d'água frágil fornece uma garantia de que a imagem marcada não seja despercebidamente editada ou adulterada. As marcas frágeis de autenticação detectam qualquer alteração na imagem. Às vezes, esta propriedade é indesejável. Por exemplo, ajustar brilho/contraste para melhorar a qualidade da imagem pode ser um processamento válido, que não deveria ser detectado como uma tentativa de adulteração maliciosa. Ou então, compactar uma imagem com perdas (como JPEG ou JPEG2000) em diferentes níveis de compressão deveria ser uma operação permitida. Ainda, imprimir e escanear uma imagem não deveria levar à perda da autenticação. Assim, foram criadas as marcas d'água semifrágeis.

Uma marca semifrágil também serve para autenticar imagens. Diferentemente, estas procuram distinguir as alterações que modificam uma imagem substancialmente daquelas que não modificam o conteúdo visual da imagem. Uma marca semifrágil normalmente extrai algumas características da imagem que permanecem invariantes através das operações permitidas e as insere de volta na imagem de forma que a alteração de uma dessas características possa ser detectada.

Tipos de Marcas de Autenticação

Pode-se subdividir as marcas de autenticação (tanto frágeis como semifrágeis) em três subcategorias: sem chave, com chave secreta (cifra simétrica) e com chave pública/privada (cifra assimétrica).

Uma marca de autenticação sem chave é útil para detectar as alterações não intencionais na imagem tais como um erro de transmissão ou de armazenamento. Funciona como uma espécie de *checksum*. Se o algoritmo de autenticação sem chave estiver disponível publicamente, qualquer pessoa pode inserir este tipo de marca em qualquer imagem e qualquer pessoa pode verificar se uma imagem contém uma marca válida.

A marca de autenticação com chave secreta (cifra simétrica) é usada para detectar uma alteração que pode ser inclusive intencional ou maliciosa. Este tipo de marca é similar aos códigos de autenticação de mensagem, sendo que a única diferença é que o código de autenticação é inserido na imagem ao invés de ser armazenado separadamente. Os algoritmos para inserção e detecção deste tipo de marca podem ser disponibilizados publicamente, e uma chave secreta é usada em ambas as fases.

As marcas de autenticação com chave pública (cifra assimétrica) utilizam a criptografia de chave pública para inserir uma assinatura digital na imagem. Usando uma cifra de chave pública, a autenticidade de uma imagem pode ser julgada sem a necessidade de se tornar pública qualquer informação privada.

Marca de Autenticação em Imagens de Tonalidade Contínua e Imagens Binárias

Existe uma forma natural de embutir as marcas de autenticação em imagens de tonalidade contínua (*contone*) não compactadas, que é inserir os dados nos bits menos significativos (LSBs). Alterar os LSBs afeta muito pouco a qualidade da imagem, ao mesmo tempo em que se conhece exatamente os bits que serão afetados pela inserção da marca.

Não ocorre o mesmo com as imagens binárias, onde cada *pixel* consiste de um único bit, de forma que não existe LSB. Isto traz dificuldades especiais para projetar marcas de autenticação para este tipo de imagem. Inserir uma marca de autenticação em imagens *contone* compactadas com perdas também apresenta dificuldades especiais.

Entre os três tipos de marca de autenticação, a de chave pública é a que oferece mais recursos. Alguns possíveis usos de uma marca de autenticação de chave pública são mostrados a seguir:

- câmera digital segura - costuma-se citar o artigo de (FRIEDMANN, 1993) como o trabalho que inspirou os primeiros trabalhos de marca d'água de autenticação. A câmera digital proposta produz dois arquivos de saída para cada imagem capturada: a primeira é a própria imagem digital capturada pela câmera em algum formato; e a segunda é uma assinatura digital produzida aplicando a chave privada da câmera (que deve estar armazenada de forma segura em um circuito integrado dentro da câmera). O usuário deve tomar cuidado para guardar os dois arquivos, para que se possa autenticar a imagem mais tarde. Uma vez que a imagem digital e a assinatura digital são geradas pela câmera e armazenadas no computador, a integridade e a autenticidade da imagem pode ser verificada usando um programa para decodificar a assinatura digital, que pode ser distribuído livremente aos usuários. O programa de verificação recebe como entrada a imagem digital, a assinatura digital e a chave pública da câmera. Ele calcula a função *hash* da imagem digital, decriptografa a assinatura digital e verifica se as duas impressões digitais obtidas são iguais. O esquema proposto por Friedman poderia ser melhorado de duas formas. A primeira seria embutir a assinatura digital no arquivo da imagem, o que eliminaria a necessidade de armazenar dois arquivos para cada imagem. Alguns formatos de imagem permitem armazenar alguns dados adicionais no cabeçalho ou rodapé do arquivo. Mas o mais interessante seria embutir a assinatura digital na própria imagem. A segunda seria permitir a localização da região alterada. Isto poderia ser interessante, por exemplo, para descobrir a intenção do falsificador ao adulterar a imagem. A marca d'água de autenticação de chave pública pode ser usada para incorporar essas melhorias à câmera de Friedman;

- autenticação de imagens distribuídas pela rede - uma agência de notícias necessita distribuir pela Internet uma fotografia jornalística, com alguma prova de autenticidade de que a foto foi distribuída pela agência e que ninguém introduziu alterações maliciosas na foto. A agência pode inserir uma marca d'água de autenticação na imagem e distribui a foto marcada;

- fax confiável - uma "máquina de FAX confiável" poderia conter internamente uma chave privada e inserir uma marca d'água em todos os documentos transmitidos por

ela. O receptor de FAX, usando a chave pública da máquina transmissora, poderia verificar que o documento foi originado de uma máquina específica de FAX e que o documento não foi manipulado.

Extração de Marca D'água

Com relação a extração da marca dágua, tem-se três tipos de sistemas diferentes. Cada um deles se diferencia pela sua natureza ou combinação de entradas e saídas:

- marcas d'água privadas (também chamadas de não-cegas) - esse sistema requer a marca d'água original. Dentro desse esquema, existem 2 tipos. No primeiro, é necessário o arquivo original para achar pistas de onde se localiza a marca dentro do arquivo marcado. O sistema do segundo tipo necessita das mesmas informações do anterior, mas ele somente tenta responder a seguinte pergunta: o arquivo contém a marca d'água? Sim ou não?. Espera-se que este sistema seja mais robusto já que transporta pouca informação e requer acesso a dados secretos;

- marcas d'água semiprivadas ou semi-cegas - diferente do anterior, não utiliza o arquivo original na extração. Entretanto, tenta responder a mesma questão. Algumas aplicações onde se poderia utilizar esse esquema seria para provar a propriedade em côrte ou em mecanismos de controle de cópia como em aparelhos de DVDs. No último caso, a chave poderia ser guardada dentro da memória do DVD e qualquer disco que fosse colocado no aparelho só poderia ser decodificado se a marca d'água pudesse ser extraída dos dados de vídeo do anterior. Como não se pode colocar os dados originais de todos os possíveis vídeos a serem usados no aparelho, não se pode usar o esquema de marcas d'água privadas descrito no item anterior;

- marcas d'água públicas ou cegas - não requer nem o arquivo original nem a marca. A intenção do esquema é tentar retirar a marca do dado sem pistas de onde ele se localiza ou como ele seria.

Um estudo sobre diversos algoritmos de marca d'água está descrito em (MEERWALD, 2001).

Para complementar, a tabela 2.3 apresenta um comparativo indicando o objetivo, as especificações e os detalhes de detecção e extração dos algoritmos de marca d'água e esteganografia .

2.3.2. Aplicativos Existentes

Atualmente as redes de computadores provêem um canal de fácil utilização para a esteganografia. Vários tipos de arquivo podem ser utilizados como imagem de cobertura incluindo imagens, sons, texto e até executáveis. Por isso é grande o número de aplicativos já criados para tentar usar esta facilidade. Por outro lado, existem também alguns softwares de esteganálise que tentam localizar os dados embutidos nas diversas mensagens de cobertura. Tais aplicações podem ser encontradas facilmente na Internet e funcionam em

Tabela 2.3: Tabela comparativa entre Esteganografia e Marca D'água (WANG; WANG, 2004).

	Exigências	Marca d'agua		Esteganografia
		Privado	Público	
Objetivo	Proteção de direitos de propriedade intelectual	++++		–
	Transmissão da mensagem secreta sem levantar suspeita	–		++++
Especificações	Invisibilidade Perceptível	++++		+++++
	Estatístico ou Algoritmo de Invisibilidade	+		+++++
	Robustes em relação à remoção, destruição, ou falsificação maldosa	+++++		–
	Resistência em relação ao processo de um sinal normal	++++		+
	Capacidade de resistência a compressão comum	++++		++
	Alto Custo	++		++++
Detecção/ Extração	Extração/Detecção sem objeto inserido	–	++++	++++
	Extração somente com presença do objeto inserido	++++	–	–
	Exigência de baixa complexidade na extração/deteção	++		+++
	capacidade opcional de download automático do objeto	+		++
Nota: Crucial: +++++ necessário: ++++ importante: +++ desejável: ++ útil: + desnecessário ou irrelevante: – Os esquemas públicos de marca d'água não necessitam do objeto original na deteção/extração; os esquemas confidenciais requerem a presença do original.				

várias plataformas, desde o DOS, Windows passando por MAC/OS até o Unix/Linux. Esta subseção apresenta alguns exemplos destes softwares e seu funcionamento.

As ferramentas *Ezstego* e *Stego Online* (EZSTEGO, 2007) foram desenvolvidas em Java por Romana Machado e limitadas a imagens indexadas de 8bits no formato GIF. Outra aplicação em Java fácil de usar é o *Revelation* (SOFTWARE, 2007), que esconde arquivos em imagens de cobertura no formato *bitmap* de 24 bits. Por serem escritas em Java as ferramentas são altamente portáveis, podendo ser executadas em Linux, Unix, Windows e MAC/OS.

A tela inicial do Revelation é bem auto explicativa (Figura 2.8). A opção *Conceal* esconde o dado embutido na imagem de cobertura a ser escolhida. A opção *Reveal* decodifica o dado embutido a partir da imagem de cobertura. A seguir encontram-se as telas do processo de leitura do dado embutido. Um arquivo chamado *texto.txt* foi escondido em uma imagem chamada *bitmap.bmp* utilizando o software Revelation (Figura 2.9). Após a decodificação, o arquivo de saída gerado se chama *texto2.txt*, conforme pode ser visto na Figura 2.10.

Figura 2.8: Tela inicial do software Revelation utilizado para esteganografia em figuras com formato bitmap de 24 bits.

Figura 2.9: Passo 1 da decodificação do arquivo escondido na imagem bitmap.bmp. A imagem modificada é escolhida.

Figura 2.10: Segundo passo na decodificação usando o Revelation. Ao pressionar o botão de próximo passo (seta para a direita) a imagem de cobertura é decodificada.

A última tela, com o prompt do sistema operacional, compara os dois arquivos e mostra que são idênticos. Sendo assim, foi possível esconder o texto na imagem e depois recuperá-lo sem problemas (Figura 2.11).

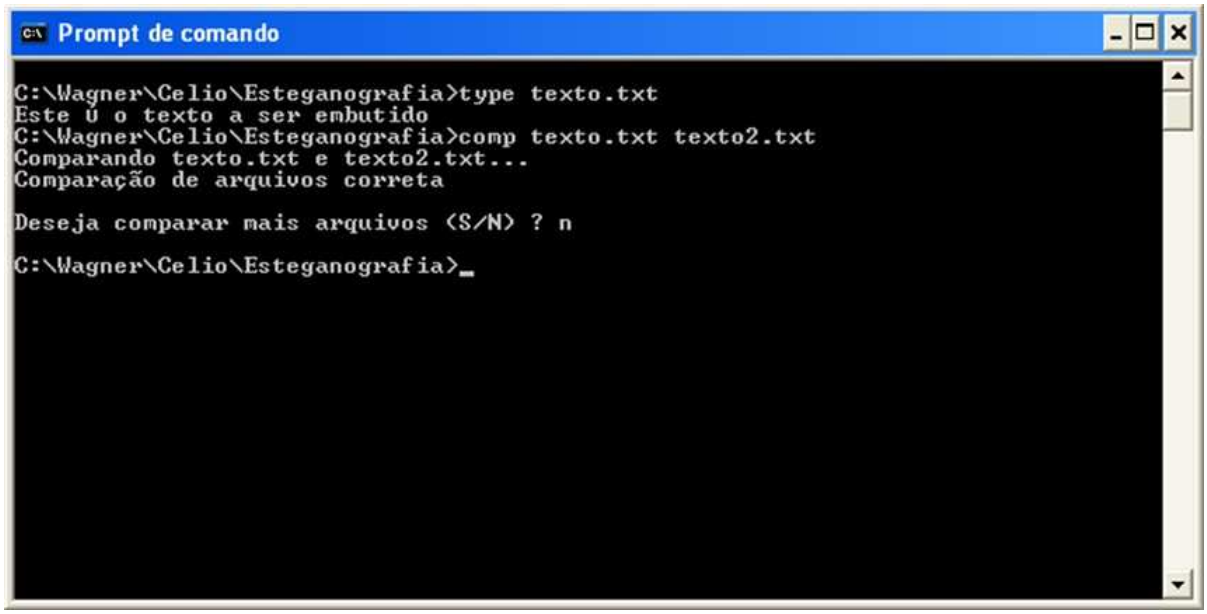


Figura 2.11: Comparação dos arquivos antes e depois do processo. O arquivo de saída do Revelation é idêntico ao arquivo que foi escondido.

O *Hide and Seek* (HIDE; SEEK, 2007) foi desenvolvido por Colin Maroney e é capaz de inserir uma lista de arquivos em uma imagem no formato JPEG. A ferramenta porém não faz uso de criptografia. Uma outra ferramenta parecida chamada *Jphide and Seek* (JPHIDE; SEEK, 2007) desenvolvida por Allan Latham, contém na verdade dois arquivos: *jphide.exe* e *jpseek.exe*. O primeiro esconde a mensagem em um arquivo JPEG e o segundo extrai a mensagem. O programa utiliza criptografia de chave simétrica e o usuário é obrigado a fornecer uma *pass phrase*.

O Jphide and Seek também é de simples operação (Figura 2.12). A opção *Hide* esconde o dado embutido (arquivo de entrada) na imagem de cobertura com formato JPEG (Figura 2.13). É interessante notar que o aplicativo analisa a imagem de cobertura e diz qual o tamanho máximo que o arquivo de entrada deve ter para que o processo seja seguro (Figura 2.14). A opção *Seek* decodifica o dado embutido usando a imagem de cobertura e o salva em um arquivo de saída (Figura 2.15).

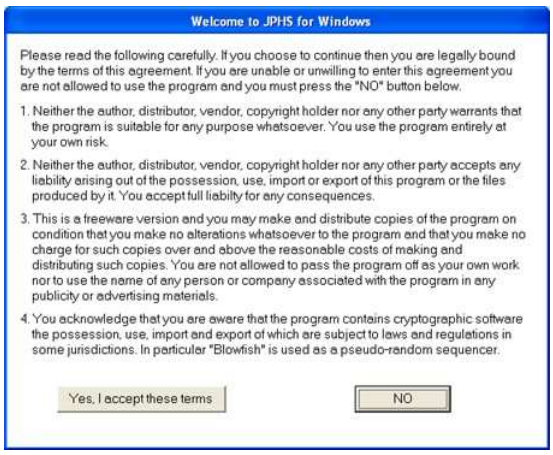


Figura 2.12: Tela inicial do JpHide and Seek.

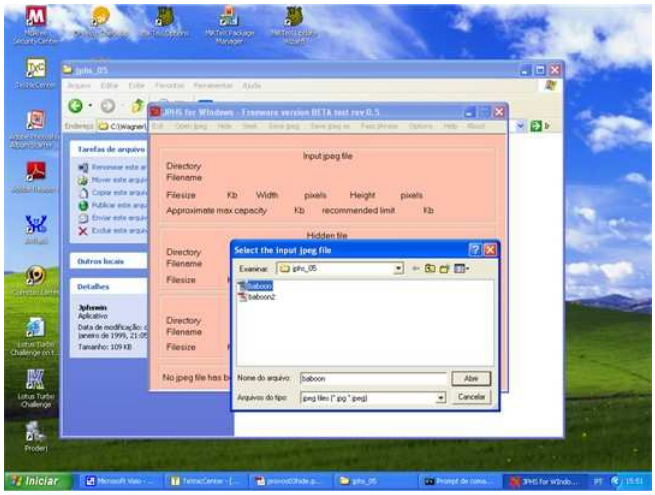


Figura 2.13: Escondendo um arquivo no Jphide and Seek.

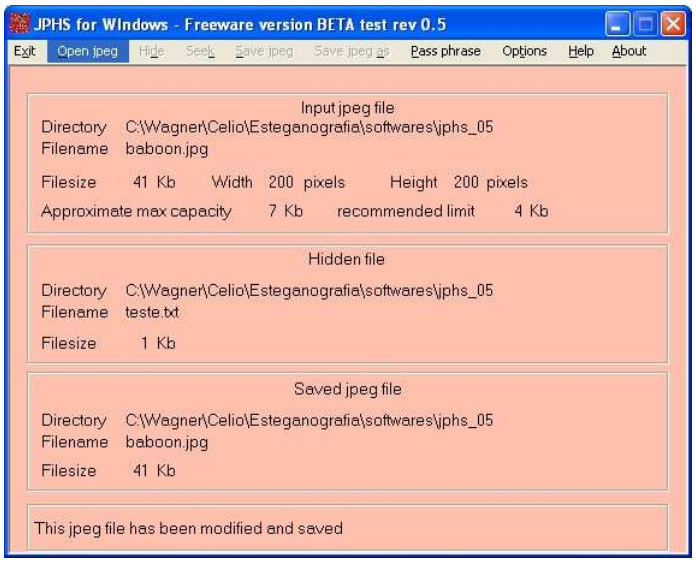


Figura 2.14: Tela após o processo de esteganografia. O arquivo *teste.txt* está escondido dentro da imagem *baboon.jpg*.

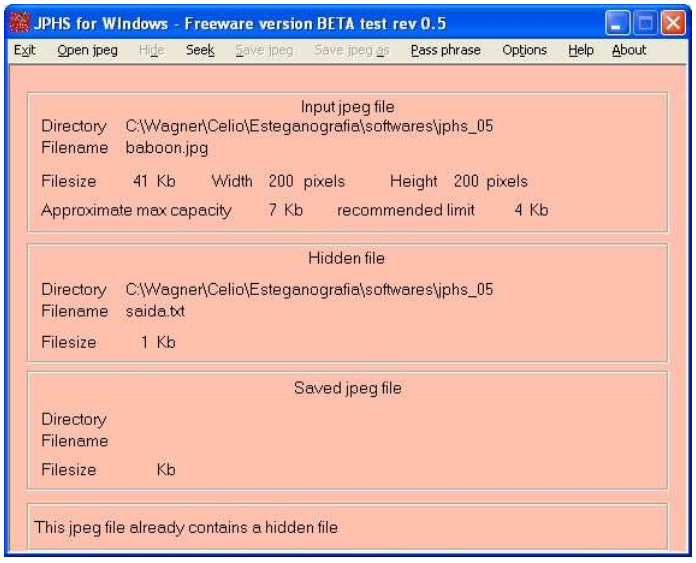


Figura 2.15: Recuperando o dado embutido com Jphide and Seek.

Niels Provos desenvolveu o *Outgess* (OUTGESS, 2007). Este software se propõe a melhorar o passo da codificação da imagem JPEG através de um gerador pseudo-randômico de números. Os coeficientes da transformada de cosseno são escolhidos também de maneira randômica para serem substituídos pelos número gerados aleatoriamente. O bit menos significativo dos coeficientes selecionados é substituído pela mensagem cifrada. Testes estatísiticos de primeira ordem não são capazes de detectar mensagens mascaradas com o *Outgess*. O pseudo-código gerado a partir do Outguess é apresentado na Figura 2.16.

```
Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
while data left to embed do
   get pseudo-random DCT coefficient from cover image
   if DCT != 0 and DCT != 1 then
      get next LSB from message
      replace DCT LSB with message LSB
   end if
   insert DCT into stego image
end while
```

Figura 2.16: Pseudo-código do OUTGESS (PROVOS; HONEYMAN, 2003).

Os softwares de esteganálise se dispõem a descobrir se os arquivos usados como mensagem de cobertura contém algum dado embutido e se possível identificar o software utilizado no processo de esteganografia. Um destes softwares é o *StegSpy* (STEGSPY, 2007), que permite a identificação de um arquivo que serve como mensagem de cobertura (Figura 2.17). O programa detectará a esteganografia e o software utilizado para esconder o dado embutido (Figura 2.18). A versão atual do software também identifica a localização da mensagem embutida dentro do arquivo de cobertura (Figura 2.19). O *StegSpy* atualmente identifica os programas Hiderman, JPHide and Seek, Masker, JPegX e Invisible Secrets.

Figura 2.17: Tela inicial do StegSpy.

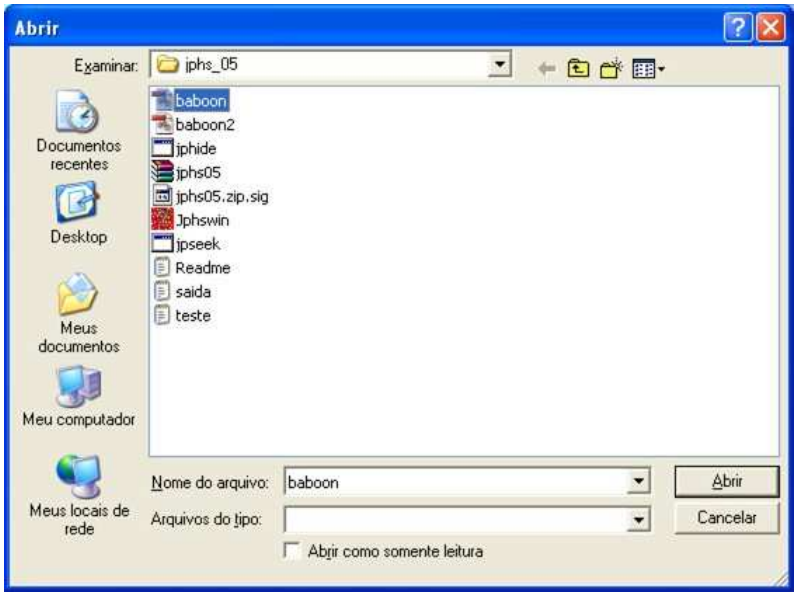


Figura 2.18: Escolhendo uma imagem que sofreu processo de esteganografia. Neste caso o arquivo “baboon.jpg” que é imagem de cobertura utilizada pelo Jphide and Seek.

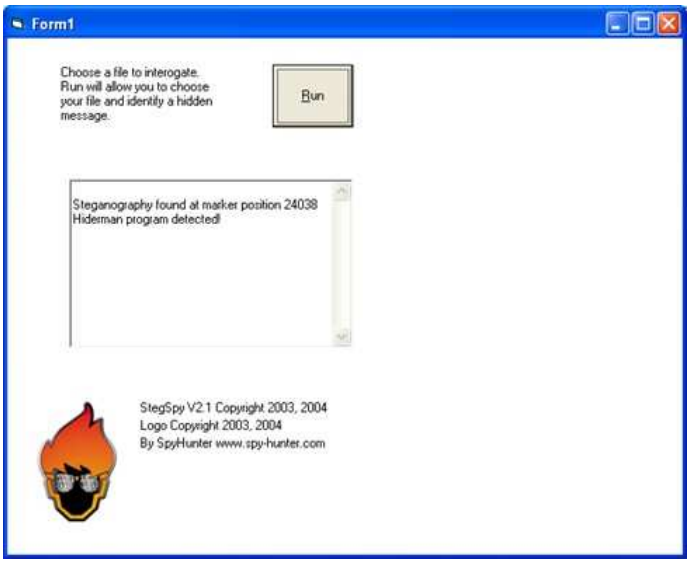


Figura 2.19: O StegSpy acusou uma assinatura de esteganografia no arquivo analisado.

Outra ferramenta de esteganálise é o *StegDetect* (STEGDETECT, 2007) que foi desenvolvida pelo mesmo autor do Outgess (Niels Provos). Este software se propõe a detectar conteúdo esteganográfico gerado pelo softwares Jsteg, JP Hide and Seek, Invisible Secrets, versões mais antigas do Outguess, F5, AppendX, e Camouflage. A versão mais atual do StegDetect suporta análise discriminante linear (LDA) para detectar qualquer estego sistema.

No campo das marcas d'água, existem vários softwares para gerar marcas em diversos tipos de mídias tais como TeleTrax, Alpha Tec, Syscop e DataMark [1]. O ponto fundamental de todos os programas é a robustez da marca produzida. Neste sentido, é preciso testar esta robustez de alguma forma. Em novembro de 1997 a primeira versão do *StirMark* (STIRMARK, 2007) foi publicada. Trata-se de uma ferramenta para testes de robustez de algoritmos de marca d'água. Com o *StirMark* foi possível realizar o primeiro *benchmarking* de algoritmos de marca d'água em 1999 utilizando a versão 3.1 deste software.

O programa SignIt (SIGNIT, 2007) da AlpVision é de fácil utilização para esconder números de série em imagens de vários formatos (Figura 2.20).

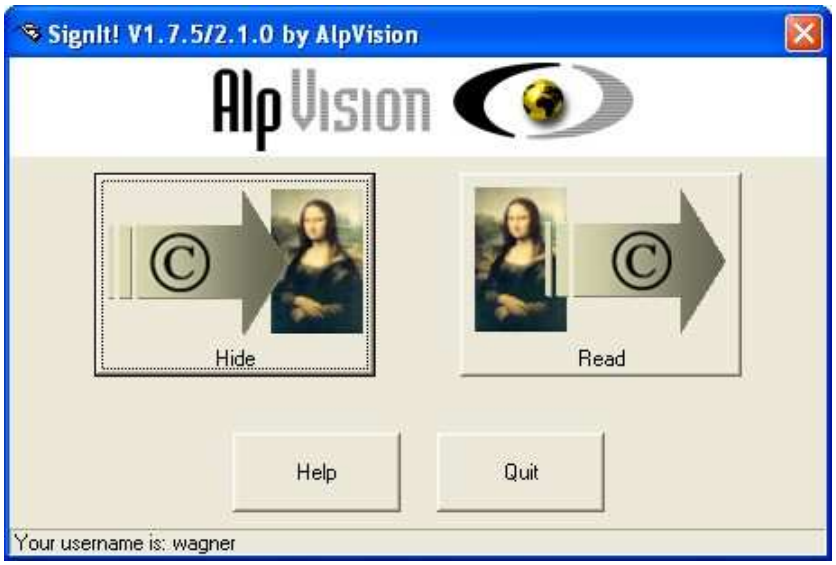


Figura 2.20: Tela inicial do SignIt utilizado para geração e leitura de marcas d´água.

Após instalado deve-se iniciar o programa e escolher entre *esconder* ou *ler* uma marca d´água em uma imagem. Na Figura 2.21, um número IDDN (*Inter Deposit Digital Number*) está sendo escondido em uma imagem. Este número é escondido em todos os lugares na imagem e não pode ser visto pelo olho nu. Além disso, é impossível remover o número de inscrição embutido sem alterar a imagem em um modo visível.

Para controlar a sua utilização, o software se conecta com a empresa desenvolvedora através da Internet (Figura 2.22), que armazena o IDDN de todos os usuários registrados, tornando esse identificador único, podendo ser utilizado para proteger os direitos autorais de imagens e localizar cópias ilegais.

[1] Todos estes programas estão listados em http://home.earthlink.net/ emilbrandt/stego/watermrk.html

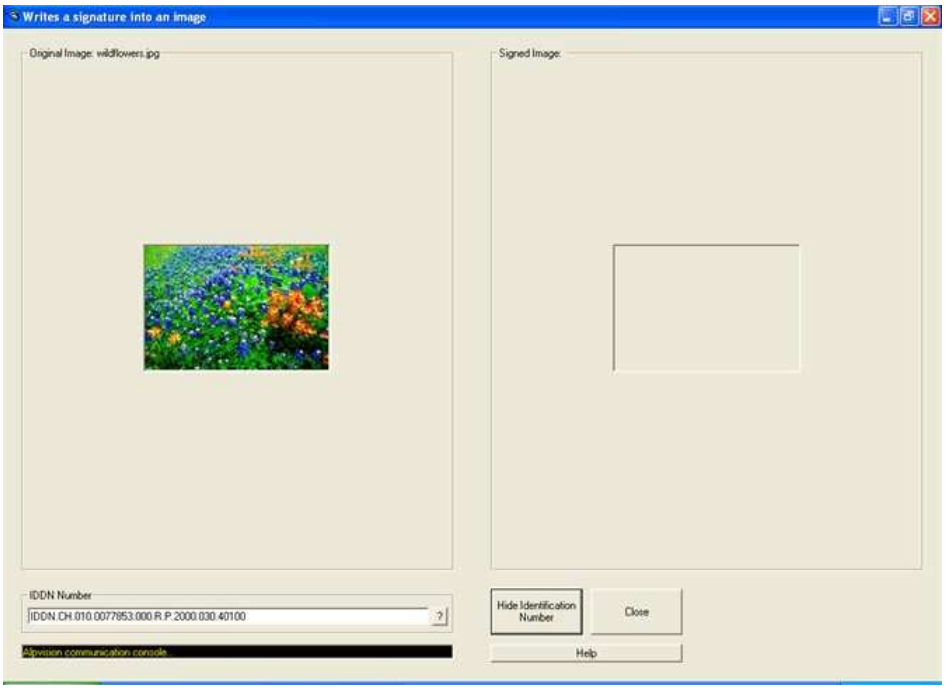


Figura 2.21: Escondendo uma número de série em uma imagem.

Figura 2.22: Software se conecta via Internet com a empresa fabricante para controlar a utilização.

Logo após a inserção da marca d'água (IDDN), o software apresenta uma comparação entre a imagem original e a imagem marcada (2.23).

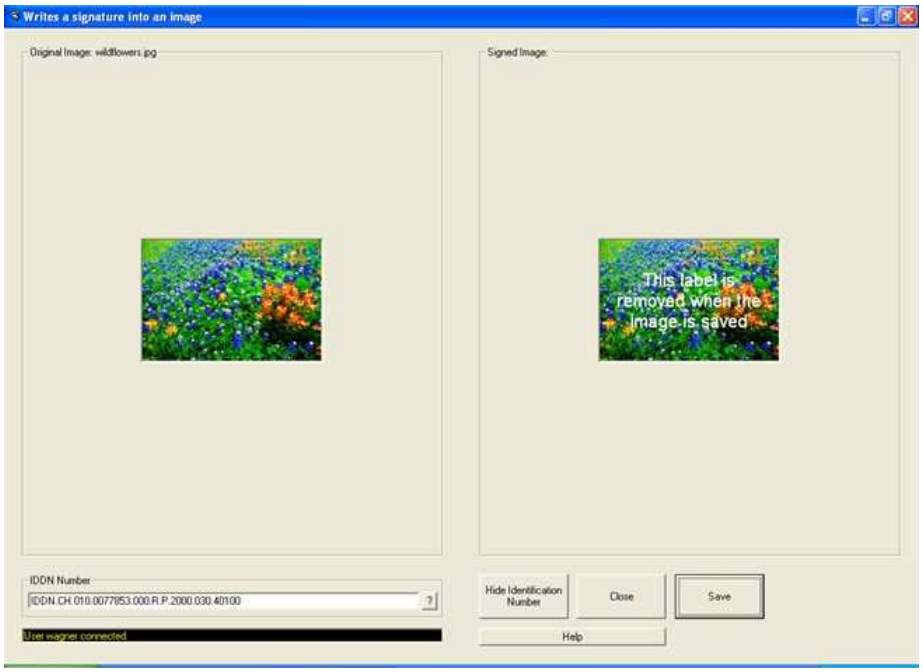


Figura 2.23: Comparação entre a imagem normal e a imagem assinada.

Ainda é possível verificar se uma determinada imagem está protegida por alguma marca, conforme visto na Figura 2.24.

Figura 2.24: Lendo uma imagem com marca d´água. A marca foi reconhecida.

O software GWatermarker foi desenvolvido por Rajan Sheth, Pinto de Adrain e Marina Chandy, todos do Departamento de Tecnologia de Informação pertencente ao Instituto de Tecnologia de San Francis, Mumbai, India (GWATERMARKER, 2007).

GWatermarker insere tanto marca d'água de forma visível a olho nu quanto de maneira invisível de forma robusta. Escrito em Virtual C++.NET, o software utiliza algoritmos próprios para a inserção e remoção das marcas visíveis e invisíveis (algoritmo RC4 para a inserção da chave secreta e o algoritmo hash MD5).

A Figura 2.25 mostra a tela inicial do GWatermarker, com a imagem Lena (LENA, 1972), muito utilizada em trabalhos relacionados a compressão de imagens e marca d'água.

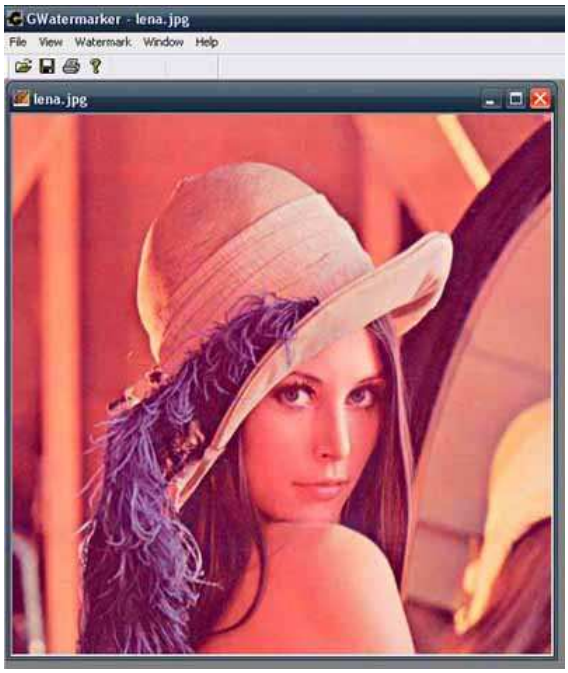
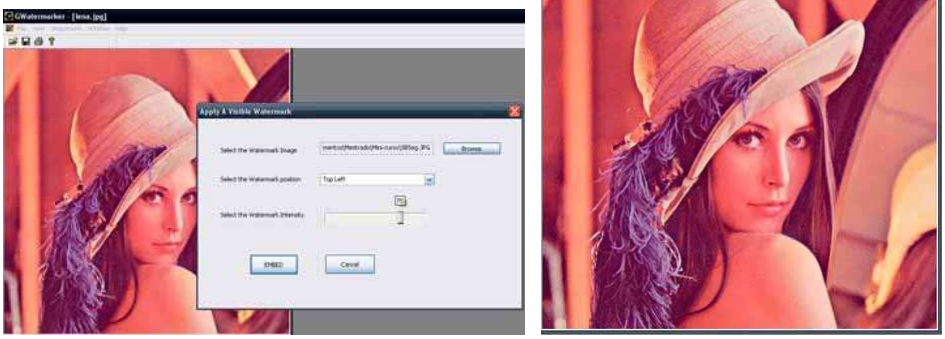


Figura 2.25: Tela inicial.

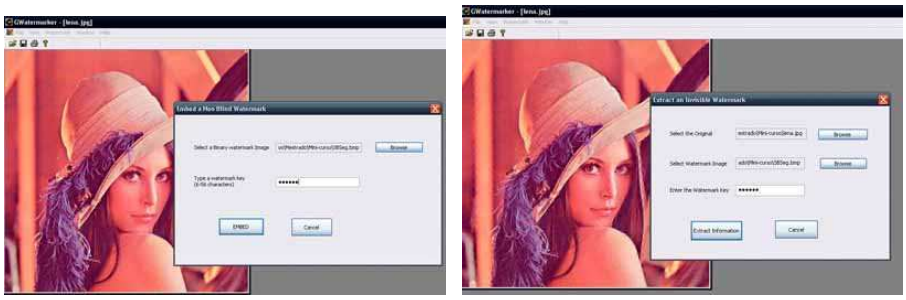
Para inserir uma marca d'água visível, deve-se selecionar o menu *Watermark/Visible Watermarking* e em seguida escolher a imagem, a posição onde ficará a marca e sua intensidade (Figura 2.26(a)). A Figura 2.26(b) apresenta a marca d'água no canto superior esquerdo da imagem, conforme selecionado.

Para a inserção da marca d'água invisível, o procedimento é similar. A partir do menu *Watermark/Invisible (Not Blind)/Embed* (Figura 2.27(a)), define-se a imagem e a chave simétrica (de 6 a 56 caracteres). Para a extração da marca d'água invisível, seleciona-se o menu *Watermark/Invisible (Not Blind)/Extract*, e em seguida a imagem original (sem a marca) a imagem marcada e a chave para a extração (Figura 2.27(b)). A Figura 2.28 confirma a presença da marca d'água na imagem Lena.



(a) Inserindo a marca d'água visível. (b) Marca d'água visível inserida.

Figura 2.26: Marca d'água visível.



(a) Inserindo marca d'água invisível. (b) Extraindo a marca d'agua invisível.

Figura 2.27: Marca d'água invisível.

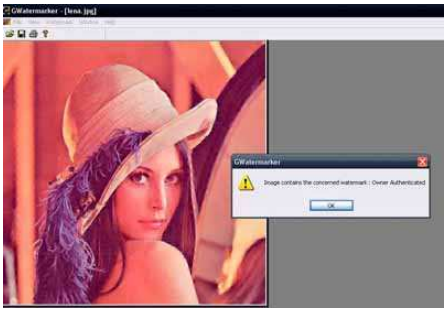


Figura 2.28: Confirmação da presença de marca d'água.

2.4. Considerações Finais e Tendências Futuras

Tanto a esteganografia quanto a marca d'água descrevem técnicas que são usadas na intenção de ocultar uma comunicação dentro de uma informação disfarce. Entretanto, esteganografia se refere tipicamente a uma comunicação ponto-a-ponto. Por isso, o método geralmente não é robusto contra modificações ou tem somente uma robustez limitada que a protege de pequenas alterações que possam ocorrer em termos de transmissão, armazenamento, mudanças de formato, compressão ou conversões digital-analógicas.

Em marcas d'água, por outro lado, o foco está na robustez. Não existe comunicação ponto-a-ponto, mas deseja-se que a marca inserida em um dado seja recuperada de algum modo depois da imagem circular por quaisquer canais típicos da aplicação. Por exemplo, pode-se marcar uma imagem que deseja-se proteger contra cópias sem autorização. Caso alguém a copie e utilize técnicas de processamento de imagem para tentar apagar a marca, ainda assim deve ser possível decodificar a marca da imagem alterada. Isso provaria quem é o verdadeiro autor ou proprietário da imagem. A questão da detecção não é tão importante, apesar de que, se o observador não perceber a marca, ele talvez nem tente removê-la.

Um exemplo de aplicação oposta seria marcar uma imagem para verificar se ela sofrerá alterações. Caso a imagem seja modificada de alguma forma, a marca será destruída, mostrando que o ato realmente aconteceu. A robustez ou a ausência dela define a aplicação da marca utilizada. As marcas d'água robustas devem resistir a ataques e alterações na imagem. As marcas frágeis devem ser destruídas caso a imagem sofra alterações.

Atualmente existem estudos para proteger a esteganografia das técnicas de esteganálise. Em (PROVOS, 2001)são apresentados novos métodos que permitem esconder mensagens de forma segura e resistentes a análise estatística.

Técnicas esteganográficas têm uso legal e ilegal. Como uso legal no presente e no futuro, esteganografia tem sido usada e será cada vez mais utilizada na proteção de direitos intelecutais, principalmente quando se considera as novas formas de comercialização utilizando mídia digital. Neste sentido as técnicas de marca dágua parecem ser um campo profícuo de pesquisa e aplicações no futuro.

Por outro lado, há o uso ilegal de técnicas esteganográficas, que cresce cada vez mais, em virtude da facilidade de acesso a Internet. Usar esteganografia para transitar mensagens ou até pequenas imagens de pornografia ou pedofilia é possível e provável. Um relatório de crimes de tecnologia (HIGH..., 2007) lista alguns tipos de crime comuns utilizando alta tecnologia:

- comunicações criminosas;
- fraudes;
- *hacking*;
- pagamentos eletrônicos;
- pornografia e pedofilia;

- ofensas a propriedade intelectual;
- propagação de vírus e cavalos de tróia.

Um exame preliminar desta lista mostra vários casos de mau uso da esteganografia, principalmente no que se refere à comunicação criminosa. Em termos de segurança da informação há também outras áreas de interesse. Uma área com uso potencial em várias aplicações é o desenvolvimento de protocolos que usam esteganografia para burlar censura. Em (HASELTON, 2000), o coordenador da organização peacefire.org, uma organização que se opõe à censura na Internet a menores de 18 anos, descreve um protocolo que seria "indetectável" por sensores.

Há também a possibilidade de ataques de vírus utilizarem técnicas de esteganografia. As técnicas e ferramentas esteganográficas podem ser utilizadas em conjunto com outras aplicações para automaticamente extrair informações escondidas sem a intervenção do usuário. Um cenário possível para um ataque de vírus poderia ser o envio de uma mensagem escondida em uma imagem enviada por e-mail. Um cavalo de tróia instalado na máquina poderia então extrair o vírus da imagem e infectar várias máquinas.

Finalizando, este trabalho apresentou a evolução da esteganografia ao longo da história e suas aplicações modernas com a chamada esteganografia digital. Foram mostradas as principais técnicas de mascaramento e, em especial, mascaramento em imagens. A esteganografia, quando bem utilizada, fornece meios eficientes e eficazes na busca por proteção digital. Associando criptografia e esteganografia, as pessoas têm em mãos o poder de comunicar-se em segredo pela rede mundial de computadores mantendo suas identidades íntegras e secretas.

Referências Bibliográficas

AVCIBAS, I.; MEMON, N.; SANKUR, B. Steganalysis based on image quality metrics. In: *Proceedings of the Fourth Workshop on Multimedia Signal Processing*. USA: IEEE, 2001. p. 517–522.

BASSIA, P.; PITAS, I. Robust audio watermarking in the time domain. In: *9th European Signal Processing Conference (EUSIPCO'98)*. Island of Rhodes, Greece: [s.n.], 1998. p. 25–28. ISBN 960-7620-05-4. Disponível em: <citeseer.ist.psu.edu/bassia99robust.html>.

BONEY, L.; TEWFIK, A. H.; HAMDY, K. N. Digital watermarks for audio signals. In: *International Conference on Multimedia Computing and Systems*. [s.n.], 1996. p. 473–480. Disponível em: <citeseer.ist.psu.edu/article/boney96digital.html>.

BUCCIGROSSI, R. W.; SIMONCELLI, E. P. Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans Image Proc*, v. 8, n. 12, p. 1688–1701, December 1999.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification (2nd Edition)*. [S.l.]: Wiley-Interscience, 2000. ISBN 0471056693.

DUMITRESCU, S.; WU, X. Steganalysis of lsb embedding in multimedia signals. In: *Proceedings of the Intl. Conference on Multimedia and Exp*. USA: IEEE, 2002. v. 3, p. 581–584.

EZSTEGO, S. e. *Stego e Ezstego*. 2007. Disponível em: <http://www.stego.com>.

FILHO de L. et al. Electrocardiographic signal compression using multiscale recurrent patterns. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, v. 52, n. 12, p. 2739–2753, 2005.

FRIDRICH, J.; GOLJAN, M. *Practical Steganalysis of Digital Images — State of the Art*. 2002. 1-13 p.

FRIEDMANN, G. L. The trustworthy digital camera: Restoring credibility to the photographic image. v. 39, n. 4, p. 905–910, nov. 1993. ISSN 0098-3063. Disponível em: <http://www.cl.cam.ac.uk/ fapp2/steganography/bibliography/043125.html>.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 2nd. ed. Boston, MA, USA: Prentice-Hall, 2002.

GWATERMARKER. *GWatermarker*. 2007. Disponível em: <http://www.cse.unt.edu/ smohanty/ISWARwatermarker/>.

HART, S. V.; ASHCROFT, J.; DANIELS, D. J. *Forensic examination of digital evidence: a guide for law enforcement*. Department of Justice - Office of Justice Programs, USA, April 2004. Technical Report NCJ 199408.

HARTUNG, F.; GIROD, B. Digital watermarking of raw and compressed video. In: *Proc. European EOS/SPIE Symposium on Advanced Imaging and Network Technologies*. Berlin, Germany: [s.n.], 1996. Disponível em: <citeseer.ist.psu.edu/hartung96digital.html>.

HASELTON, B. *A Protocol that uses steganography to circumvent network level censorship*. 2000.

HIDE; SEEK. *Hide and Seek*. 2007. Disponível em: <ftp://csua.berkeley.edu/pub/cypherpunks/steganography/hdsk41b.zip>.

HIGH Technology Crime in California. 2007. Disponível em: <www.ocjp.ca.gov/publications/pub$_h$*tk*1.*pdf*>.

HIROHISA, H. Crocus: a steganographic filesystem manager. In: *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*. New York, NY, USA: ACM Press, 2007. p. 344–346. ISBN 1-59593-574-6.

JOHNSON, N. *Steganography*. George Mason University, 1998.

JOHNSON, N. F.; JAJODIA, S. Exploring steganography: Seeing the unseen. *IEEE Computer*, v. 31, n. 2, p. 26–34, 1998. Disponível em: <citeseer.ist.psu.edu/johnson98exploring.html>.

JPHIDE; SEEK. *Jphide and Seek*. 2007. Disponível em: <http://linux01.gwdg.de/ alatham/stego.html>.

KAHN, D. The history of steganography. In: *Proceedings of the First International Workshop*. Cambridge, UK: [s.n.], 1996.

KALKER, T. et al. Video watermarking system for broadcast monitoring. In: WONG, P. W.; III, E. J. D. (Ed.). SPIE, 1999. v. 3657, n. 1, p. 103–112. Disponível em: <http://link.aip.org/link/?PSI/3657/103/1>.

KIM, H. Stochastic model based audio watermark and whitening filter for improved detection. In: *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*. Washington, DC, USA: IEEE Computer Society, 2000. p. 1971–1974. ISBN 0-7803-6293-4.

LANGELAAR, G. C.; LAGENDIJK, R. L.; BIEMOND, J. *Real-time Labeling of MPEG-2 Compressed Video*. 1997. Disponível em: <citeseer.ist.psu.edu/519721.html>.

LENA. *Lena*. 1972. Disponível em: <http://www.cs.cmu.edu/ chuck/lennapg/lenna.shtml>.

LI, X.; YU, H. H. Transparent and robust audio data hiding in subband domain. In: *ITCC '00: Proceedings of the The International Conference on Information Technology: Coding and Computing (ITCC'00)*. Washington, DC, USA: IEEE Computer Society, 2000. p. 74. ISBN 0-7695-0540-6.

LINNARTZ, J.-P.; KALKER, T.; HAITSMA, J. Detecting electronic watermarks in digital video. In: *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*. Washington, DC, USA: IEEE Computer Society, 1999. p. 2071–2074. ISBN 0-7803-5041-3.

LU, C.; LIAO, H.; CHEN, L. *Multipurpose Audio Watermarking*. 2000. Disponível em: <citeseer.comp.nus.edu.sg/lu00multipurpose.html>.

MARVEL, L.; BONCELET, C.; RETTER, J. *Spread spectrum image steganography*. 1999. Disponível em: <citeseer.ist.psu.edu/article/marvel99spread.html>.

MEERWALD, P. *Digital Image Watermarking in the Wavelet Transform Domain*. Dissertação (Mestrado) — Department of Scientific Computing, University of Salzburg, Austria, January 2001. Disponível em: <http://www.cosy.sbg.ac.at/ pmeerw/Watermarking/MasterThesis>.

MORRIS, S. *The future of netcrime now (1) - threats and challenges*. Home Office Crime and Policing Group, USA, 2004. Technical Report 62.

OUTGESS. *Outgess*. 2007. Disponível em: <http://www.outgess.org>.

PETITCOLAS, F. A. P.; ANDERSON, R. J.; KUHN, M. G. Information hiding — A survey. *Proceedings of the IEEE*, v. 87, n. 7, p. 1062–1078, 1999. Disponível em: <citeseer.ist.psu.edu/petitcolas99information.html>.

PETITCOLAS, F. A. P.; KATZENBEISSER, S. *Information hiding techniques for steganography and digital watermarking*. 1st. ed. [S.l.]: Artech House Books, 1999.

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

PRANDONI, P.; VETTERLI, M. *Perceptually hidden data transmission over audio signals*. 1998. Disponível em: <citeseer.ist.psu.edu/prandoni98perceptually.html>.

PROVOS, N. Defending against statistical steganalysis. In: *10th USENIX Security Symposium*. [s.n.], 2001. Disponível em: <niels.xtdnet.nl/papers/defending.pdf>.

PROVOS, N.; HONEYMAN, P. Hide and seek: An introduction to steganography. *IEEE Security and Privacy*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 1, n. 3, p. 32–44, 2003. ISSN 1540-7993.

QIAO, L.; NAHRSTEDT, K. Watermarking methods for MPEG encoded video: Towards resolving rightful ownership. In: *International Conference on Multimedia Computing and Systems*. [s.n.], 1998. p. 276–285. Disponível em: <citeseer.ist.psu.edu/article/qiao98watermarking.html>.

ROCHA, A. de R. *Randomização Progressiva para Esteganálise*. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, Brasil, 2006.

SALOMON, D. *Data Compression: The Complete Reference*. Segunda edição. Nova Iorque: Springer, 2000.

SIEFFERT, M. et al. Stego intrusion detection system. AFRL/ASU Assured Information Security, Rome, NY, USA, 2004.

SIGNIT. *SignIt*. 2007. Disponível em: <http://www.alpvision.com>.

SOFTWARE, R. *Revelation Software*. 2007. Disponível em: <http://revelation.atspace.biz>.

STEGDETECT. *Stegdetect*. 2007. Disponível em: <http://www.outguess.org/detection.php>.

STEGSPY. *StegSpy*. 2007. Disponível em: <http://www.spy-hunter.com/stegspydownload.htm>.

STIRMARK. *Stirmark*. 2007. Disponível em: <htttp://www.petitcolas.net/fabien/watermarking/stirmark/index.html>.

SU, P.-C. et al. Digital image watermarking in regions of interest. In: *PICS*. [S.l.: s.n.], 1999. p. 295–300.

SULLIVAN, K. et al. Steganalysis of quantization index modulation data hiding. In: *IEEE International Conference on Image Processing*. [s.n.], 2004. p. 1165–1168. Disponível em: <http://vision.ece.ucsb.edu/publications/04ICIPKen.pdf>.

SWANSON, M. D.; ZHU, B.; TEWFIK, A. H. Current state of the art - challenges and future directions for audio watermarking. In: *ICMCS, Vol. 1*. [S.l.: s.n.], 1999. p. 19–24.

SWANSON, M. D. et al. Robust audio watermarking using perceptual masking. *Signal Processing*, v. 66, n. 3, p. 337–355, 1998. Disponível em: <citeseer.ist.psu.edu/swanson98robust.html>.

WANG, H.; WANG, S. Cyber warfare: steganography vs. steganalysis. *Commun. ACM*, ACM Press, New York, NY, USA, v. 47, n. 10, p. 76–82, 2004. ISSN 0001-0782.

WAYNER, P. *Disappearing Cryptography: Information Hiding: Steganography and Watermarking (2nd Edition)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. ISBN 1558607692.

WESTFELD, A.; PFITZMANN, A. Attacks on steganographic systems. In: *IH '99: Proceedings of the Third International Workshop on Information Hiding*. London, UK: Springer-Verlag, 2000. p. 61–76. ISBN 3-540-67182-X.

Capítulo 3

Introdução à Segurança Demonstrável

Rafael Dantas de Castro, Ricardo Dahab e Augusto Jun Devegili
Instituto de Computação
Universidade Estadual de Campinas
rafael.castro@gmail.com, rdahab@ic.unicamp.br, augusto@devegili.org

Abstract

Provable security is the subject of theoretical cryptography that studies the formal definition of strong security requirements and methods that may be used to analyse the security of cryptographic schemes with regard to these requirements. There are currently two main models in use: the standard model and the random oracle model. We use a chronological approach to present the dawn of modern cryptography and the need of strong security notions, the first provable secure cryptographic schemes and the development of the random oracle model. This text intends to be an introductory reference for those interested in understanding provable security techniques.

Resumo

Segurança demonstrável é a área de criptografia teórica que estuda a definição formal de requisitos de segurança forte e provê métodos para analisar esquemas criptográficos em relação a esses requisitos. Atualmente há dois modelos principais usados para demonstrar a segurança de tais esquemas: o modelo padrão e o modelo do oráculo aleatório. Usando uma abordagem histórica, descrevemos os primórdios da criptografia moderna e a necessidade de noções de segurança forte, os primeiros esquemas demonstravelmente seguros e o surgimento do modelo do oráculo aleatório. Este texto procura ser uma referência introdutória para os interessados em compreender as técnicas de segurança demonstrável.

3.1. Introdução

A criptografia nasceu da necessidade de pessoas se comunicarem de forma sigilosa através de um canal potencialmente inseguro. Mas ela sempre foi encarada mais como uma arte: a arte de se comunicar secretamente através de canais públicos. Seu desenvolvimento era essencialmente empírico: um "criptossistema" era bom porque ninguém sabia como quebrá-lo, e no dia em que alguém conseguisse quebrá-lo, ele deixaria de ser adequado. Tratamos aqui do processo através do qual, nas últimas décadas, a criptografia vem mudando cada vez mais o seu status de uma arte para o de uma ciência, de maneira que um pesquisador possa avaliar precisamente a segurança provida por um criptossistema e que um usuário possa tranqüilamente confiar seus segredos a um esquema baseado não na intuição dos que o criaram mas sim em evidências matemáticas incontestáveis.

O primeiro grande passo nesta direção foi dado por Shannon nos primeiros anos do pós-guerra. Ele estudou formalmente o que significa exatamente "ser seguro" e, baseado na teoria da informação, estabeleceu os principais alicerces sobre os quais a criptografia, enquanto ciência, se desenvolveria nas próximas décadas. Ele conseguiu definir, entre outras coisas, o que é a "segurança perfeita", quais são os requisitos indispensáveis para se alcançá-la e como analisar quão perto um dado criptossistema está deste ideal de segurança, criando um modelo matemático razoavelmente completo para descrever e analisar sistemas criptográficos. Isto é de extrema importância para o desenvolvimento de qualquer ciência, pois consegue-se assim uma abstração (adequada) da realidade definida por um conjunto de axiomas a partir dos quais se descobrem fatos/verdades (teoremas) sobre a (abstração dessa) realidade.

Talvez o resultado mais célebre obtido por Shannon seja a demonstração matemática de um fato que já era uma crença largamente difundida dentro da comunidade científica: a segurança do *One-time Pad*. A relevância deste resultado não pode ser superestimada. A sua maior importância não está no resultado em si, mas sim em representar o nascimento da segurança demonstrável: pela primeira vez alguém conseguia um argumento matemático que justificasse a segurança de um criptossistema dentro de um certo modelo. A observação de que esta demonstração se baseia em um modelo às vezes escapa a muita gente: Shannon, de alguma maneira, modelou matematicamente a realidade, fez algumas suposições sobre a maneira como o criptossistema seria usado e sobre como potenciais adversários poderiam interagir com ele para tentar quebrá-lo. Somente sob essas circunstâncias ele foi capaz de provar que o *One-time Pad* é perfeitamente seguro.

Falaremos mais sobre as idiossincrasias deste resultado em §3.3, mas aqui gostaríamos de destacar as duas grandes contribuições de Shannon, implícitas neste resultado, que levaram ao início da transformação da criptografia em uma ciência:

- **Noções de segurança.** Shannon conseguiu definir formalmente os requisitos que um criptossistema deveria apresentar para ser seguro, definindo claramente o que significa "quebrar" o sistema.

- **Modelos de ataque.** Shannon definiu exatamente como ataques ao sistema poderiam ser executados, qual seria o comportamento esperado de um adversário e como analisar isto matematicamente.

Shannon chegou a estes resultados fortemente baseado na intuição geral do que se esperava de um criptossistema, mas o grande passo dele foi formalizar estas intuições e conseguir construir uma teoria utilizando estes formalismos como alicerces.

Na década de 70, com o gradativo aumento da importância dos computadores em diversas áreas e o seu proporcional aumento de capacidade, criou-se simultaneamente a demanda e a oportunidade para o surgimento da criptografia assimétrica. Neste novo paradigma de criptografia as teorias de Shannon, como previamente definidas, eram de pouca valia: a noção importante agora não era mais a de um sigilo perfeito, impossível de ser atingida na criptografia assimétrica, mas sim de um sigilo computacional, que indicasse que o custo envolvido em quebrar um criptossistema seria proibitivo.

Como é comum quando ocorre qualquer mudança grande de paradigma, os primeiros trabalhos encontraram dificuldades para formalizar as noções de segurança devido à falta de um modelo apropriado. Só após alguns anos percebeu-se qual seria o caminho adequado para esta formalização e um trabalho, de certa forma análogo ao feito por Shannon décadas antes para a criptografia simétrica, passou a ser realizado para definir o que significa "ser seguro" neste novo paradigma de criptografia (e, logicamente, definir esquemas que concretizem estas definições).

A questão aqui é naturalmente mais complicada devido à assimetria: enquanto no tratamento dado por Shannon o objetivo é provar que um possível adversário não tem informação suficiente para quebrar o sistema, no caso da criptografia assimétrica isso não é verdade. Por exemplo, uma chave pública sempre tem informação suficiente para que se calcule a sua chave privada correspondente; o que precisa ser provado aqui é o quão difícil é extrair esta informação.

Novamente, a base deste estudo passa pela definição de noções de segurança e modelos de ataque, e é este o principal assunto deste texto. Iniciaremos nosso tratamento do tema na seção §3.2 com uma rápida passagem pelos resultados seminais de Shannon relacionados à teoria da informação e pelos primórdios da criptografia assimétrica, contando com noções ainda ingênuas de segurança. Esta seção é baseada em [Goldreich, 2003] e [Stinson, 2006]. A seguir, na seção §3.3 revisaremos o trabalho de formalização da criptografia feito durante os anos 80, fortalecendo as noções de segurança utilizadas, revisando criptossistemas e esquemas de assinaturas que as concretizavam e apresentando diferentes provas de segurança para sustentar estes resultados. Boa parte da apresentação desta seção é baseada em [Goldreich, 2003] e [Goldreich, 2004]. Em seguida, na seção §3.4, discutiremos o *paradigma do oráculo aleatório* para projeto de protocolos criptográficos, apresentando alguns esquemas cuja segurança será analisada neste modelo, e tecendo alguns comentários sobre a controvérsia que cerca o uso e aceitação deste modelo. Encerramos com algumas considerações finais na seção §3.5, destacando alguns dos tópicos importantes que infelizmente não puderam ser cobertos pelo texto e comentando alguns dos principais desafios enfrentados pelos pesquisadores da área atualmente.

3.2. Primeiras Noções de Segurança

Nesta seção abordamos brevemente dois trabalhos de suma importância para o surgimento da criptografia moderna: o tratamento dado por Shannon para a segurança de

criptossistemas, baseado na teoria da informação; e o surgimento da criptografia assimétrica em [Diffie e Hellman, 1976], acompanhado das primeiras, ainda ingênuas, definições de segurança.

3.2.1. Shannon e a Criptografia Simétrica

Formalmente, um *criptossistema* $\mathfrak{C}$ é uma tupla $(\mathscr{P},\mathscr{C},\mathscr{K},\mathscr{E},\mathscr{D})$ onde:

1. $\mathscr{P}$ é um conjunto enumerável de possíveis *textos em claro* com uma distribuição de probabilidade discreta associada e um algoritmo de amostragem $A_{\mathscr{P}}$;
2. $\mathscr{C}$ é um conjunto enumerável de possíveis *textos cifrados* com uma distribuição de probabilidade discreta induzida por $\mathscr{P}$ e $\mathscr{K}$ e um algoritmo de amostragem $A_{\mathscr{C}}$;
3. $\mathscr{K}$ é um conjunto enumerável de possíveis *chaves* com uma distribuição de probabilidade discreta associada e um algoritmo de amostragem $A_{\mathscr{K}}$;
4. Para cada $K \in \mathscr{K}$, existe uma regra de ciframento $e_K \in \mathscr{E}$ e uma regra de deciframento $d_K \in \mathscr{D}$ correspondente. Cada $e_K : \mathscr{P} \to \mathscr{C}$ e $d_K : \mathscr{C} \to \mathscr{P}$ são funções tais que $d_K(e_K(x)) = x$ para todo texto em claro $x \in \mathscr{P}$.

Perceba que esta definição não trata da segurança do criptossistema e até esquemas trivialmente inseguros se encaixam. Abordaremos o assunto da segurança dos esquemas após uma rápida revisão de conceitos de probabilidade necessários à discussão.

Uma *variável aleatória discreta* $\mathbf{X}$ consiste em um conjunto enumerável X e uma distribuição de probabilidade discreta definida em X. A probabilidade de que a variável aleatória $\mathbf{X}$ tenha o valor x é denotada por $\Pr[\mathbf{X} = x]$ ou então $\Pr[x]$ caso a variável $\mathbf{X}$ esteja fixada. Seja $E \subseteq X$. A probabilidade de que $\mathbf{X}$ tenha um valor em E é dada por $\Pr[x \in E] = \sum_{x \in E} \Pr[x]$; E é chamado de *evento*. A *probabilidade condicional* $\Pr[x|y]$ é a probabilidade de que $\mathbf{X}$ tenha o valor x dado que $\mathbf{Y}$ tem o valor y. Pelo teorema de Bayes, $\Pr[x|y] = (\Pr[x]\Pr[y|x])/\Pr[y]$; duas variáveis $\mathbf{X}$ e $\mathbf{Y}$ são *variáveis aleatórias independentes* se e somente se $\Pr[x|y] = \Pr[x]$ para todo $x \in X$ e $y \in Y$. Se $F(\cdot)$ é uma função com domínio X, então $F(\mathbf{X})$ denota a aplicação da função F a um elemento de X escolhido de forma aleatória conforme a distribuição de probabilidade definida em X.

Com estas definições de criptossistema e probabilidade elementar, podemos começar a descrever propriedades da segurança de criptossistemas. Suponha que Alice e Beto queiram usar um criptossistema $\mathfrak{C}$ para se comunicar de forma confidencial. Alice e Beto escolhem uma chave $K \in \mathscr{K}$ conforme o algoritmo de amostragem definido em $\mathscr{K}$. Podemos assumir que as variáveis aleatórias referentes a $\mathscr{P}$ e $\mathscr{K}$ são independentes. Alice cifra seu texto $x \in X$ com a chave K e envia $y = e_K(x) \in \mathscr{C}$ para Beto. Suponha que Eva, a adversária, intercepte y. O quanto de informação ela consegue obter sobre x a partir de y?

Em um mundo ideal, Eva não deveria conseguir informação alguma sobre x a partir de y. Neste caso, dizemos que o criptossistema $\mathfrak{C}$ possui *sigilo perfeito*. Definindo formalmente sigilo perfeito, dizemos que a probabilidade *a posteriori* de se saber x a partir de y deve ser igual à probabilidade *a priori* de x. Pelo teorema de Bayes,

$$\Pr[x|y] = \frac{\Pr[x]\Pr[y|x]}{\Pr[y]}$$

onde $\Pr[x|y]$ é a probabilidade *a posteriori* da mensagem x caso o texto cifrado y seja interceptado, $\Pr[x]$ é a probabilidade *a priori* de x, $\Pr[y|x]$ é a probabilidade do texto cifrado y caso o texto em claro x tenha sido escolhido, ou seja, a soma das probabilidades de todas as chaves que produzem y a partir de x, e $\Pr[y]$ é a probabilidade de se obter o texto cifrado y.

Para sigilo perfeito precisamos de que $\Pr[x|y] = \Pr[x]$, i.e., ou $\Pr[x] = 0$, o que não deve acontecer[1], ou então $\Pr[y|x] = \Pr[y]$. Temos então uma caracterização de sigilo perfeito conforme o teorema a seguir.

Teorema 1 (Sigilo perfeito)**.** *Um criptossistema possui sigilo perfeito se e somente se* $\Pr[y|x] = \Pr[y]$ *para todo* $x \in \mathscr{P}$ *e* $y \in \mathscr{C}$*, i.e.,* $\Pr[y|x]$ *deve ser independente de x.*

O teorema de sigilo perfeito pode ser interpretado da seguinte forma: para todo $x, w \in \mathscr{P}$ e $y \in \mathscr{C}$, a probabilidade total de todas as chaves que transformam x em y deve ser igual à de todas as chaves que transformam w no mesmo y. Note que ao se fixar uma chave $K \in \mathscr{K}$ existe um e apenas um $y = e_K(x)$ para um determinado texto em claro x. Por conseguinte, o espaço de textos cifrados precisa ter pelo menos a mesma cardinalidade que o espaço de textos em claro. Note também que, para obter sigilo perfeito, $\Pr[y|x] = \Pr[y] \neq 0$ para quaisquer valores de x, y. Como a probabilidade *a posteriori* de y independe da probabilidade de x, existe pelo menos uma chave que transforma qualquer texto em claro x em qualquer um dos textos cifrados y. Entretanto, cada chave que transforma um dado x em um dado y precisa ser diferente, e portanto chegamos a uma importante conclusão de Shannon:

> *Para se obter sigilo perfeito, o número de chaves diferentes precisa ser pelo menos tão grande quanto o número de textos em claro:* $|\mathscr{K}| \geqslant |\mathscr{P}| \leqslant |\mathscr{C}|$.

Podemos medir a quantidade de informação produzida quando se escolhe um texto em claro através da entropia: $H(\mathscr{P}) = -\sum_{x \in \mathscr{P}} \Pr[x] \log \Pr[x]$, e de forma similar para a incerteza associada à escolha de uma chave, $H(\mathscr{K})$. Se todos os textos em claro são eqüiprováveis, então $H(\mathscr{P}) = \log |\mathscr{P}|$ e este é o limite superior para a entropia de um texto em claro. Caso a entropia da chave seja menor do que $\log |\mathscr{P}|$, a entropia do texto em claro diminui. A quantidade de informação de um texto em claro só pode ser escondida completamente caso a incerteza do espaço de chaves seja pelo menos igual à dos textos em claro: $H(\mathscr{P}) \leqslant H(\mathscr{K})$, o que nos leva a outra observação importante de Shannon:

> *Existe um limite para o que se consegue obter dada a incerteza da chave: a quantidade de incerteza que se consegue introduzir na solução de um criptossistema não pode ser maior do que a incerteza da chave.*

Suponha que haja um gerador seguro de chaves e que, para um texto em claro $x \in \mathscr{P}$ com tamanho L_x, seja necessária uma chave $K \in \mathscr{K}$ de tamanho L_K para cifrá-lo. Sejam R_x, R_K os logaritmos dos comprimentos dos alfabetos de $\mathscr{P}, \mathscr{K}$. Para se obter sigilo perfeito, é necessário que $R_x L_x \leqslant R_K L_K$. Esta desigualdade tem duas conseqüências diretas:

[1] Pois a igualdade deve ser verdadeira independentemente do valor de x escolhido.

- se os alfabetos de $\mathscr{P}, \mathscr{K}$ forem iguais, a condição simplifica para $L_x \leqslant L_K$;
- se o espaço de textos em claro possui cardinalidade infinita, então não é possível obter sigilo perfeito com um espaço de chaves finito.

O exemplo canônico de criptossistema com sigilo perfeito é o *One-time Pad*:

Criptossistema 1. *One-time Pad* (`OTP`)

Geração de parâmetros. Seja k um inteiro e $\mathscr{P} = \mathscr{C} = \mathscr{K} = \{0,1\}^k$.

Geração de chave. Seja $K \in \mathscr{K}$ uma chave escolhida conforme o algoritmo de amostragem definido em $\mathscr{K}$.

Ciframento. Calcule $e_K(x)$ como o resultado da operação de ou-exclusivo bit-a-bit de K e x.

Deciframento. Calcule $d_K(y)$ como o resultado da operação de ou-exclusivo bit-a-bit de K e y.

Apesar de ser uma cifra perfeita e extremamente eficiente, os requisitos de que a chave precise ser pelo menos do mesmo tamanho do texto em claro e que cada chave deva ser usada uma única vez tornam o *One-time Pad* impraticável.

Shannon também formalizou o fato de que a probabilidade de um criptoanalista decifrar mensagens aumenta quanto maior a quantidade de textos cifrados que ele obtém. Seja a *entropia condicional* dos textos em claro em relação aos textos cifrados definida como $H(\mathscr{P}|\mathscr{C}) = \sum_{x\in\mathscr{P}, y\in\mathscr{C}} \Pr[x \cap y] \log \Pr[x|y]$; é possível definir, de forma similar, a entropia respectiva do espaço de chaves. Intuitivamente, a entropia condicional é uma medida da incerteza de um texto em claro (ou chave) dado que se conhece um texto cifrado: uma entropia condicional próxima de zero indica incerteza próxima de zero, ou seja, a probabilidade de um texto em claro (ou uma chave) ser correto é próxima de um.

Como visto anteriormente, não é possível obter sigilo perfeito em um esquema onde o espaço de textos em claro seja infinito e o espaço de chaves seja finito. Um criptossistema com *sigilo ideal* é aquele em que as entropias condicionais $H(\mathscr{P}|\mathscr{C})$ e $H(\mathscr{K}|\mathscr{C})$ não se aproximam de zero quando o número de mensagens interceptadas tende ao infinito. Nestes criptossistemas, não importa quantos textos em claro um criptoanalista obtenha — não haverá uma solução única para os textos cifrados mas sim várias soluções com probabilidade similar, mesmo que o espaço de chaves seja finito. Em um criptossistema *fortemente ideal*, a entropia condicional $H(\mathscr{K}|\mathscr{C})$ mantém o valor constante $H(\mathscr{K})$.

À medida que um criptoanalista intercepta mais exemplares de texto cifrado, a entropia condicional diminui. Parametrizando a função de entropia condicional pela quantidade de texto cifrado interceptado, o teorema seguir apresenta algumas propriedades da entropia condicional.

Teorema 2. *Seja n a quantidade de texto em claro interceptado. A entropia condicional de uma chave $H(\mathscr{K}|\mathscr{C},n)$ é uma função decrescente em n. A entropia condicional das primeiras m letras do texto em claro é uma função decrescente em n e é menor ou igual à entropia condicional da chave.*

As definições de entropia, sigilo perfeito e entropia condicional são um subconjunto do trabalho de Shannon para analisar matematicamente o que significam e como analisar informação, comunicação e confidencialidade. Note que, embora Gilbert Vernam haja descrito o *One-time Pad* em 1917 e este ter sido informalmente considerado "inquebrável" por muitos anos, a primeira demonstração matemática de sua segurança incondicional surgiu com Shannon mais de 30 anos depois. Além do apresentado neste texto, conceitos como distância de unicidade, criptossistemas produto, álgebra de criptossistemas, criptossistemas puros e mistos, e métodos estatísticos aplicados a criptossistemas (incluindo confusão e difusão) foram desenvolvidos por Shannon e explorados para analisar criptossistemas simétricos.

3.2.2. Criptografia Assimétrica

Em 1976, Diffie e Hellman publicaram o artigo seminal que introduziu o conceito de criptografia de chave pública, ou criptografia assimétrica [Diffie e Hellman, 1976]. Diferentemente dos criptossistemas projetados até então, nos quais uma mesma chave era usada tanto para cifrar quanto para decifrar mensagens, a criptografia de chave pública usa um *par de chaves*: uma chave pública para ciframento e uma chave privada para deciframento. Como a chave pública é de conhecimento de todos, deixa de existir a necessidade dos criptossistemas simétricos de distribuir uma chave de forma segura apenas entre os participantes envolvidos em uma comunicação sigilosa. Mais do que isso, tornam-se possíveis esquemas simples de estabelecimento e distribuição de chaves, bem como esquemas de assinaturas digitais. Conforme a notação definida em §3.1, em um criptossistema de chave pública $\mathfrak{C} = (\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$ temos que:

- cada entidade possui um par de chaves $(SK, PK) \in \mathscr{K}$; SK é dita chave privada e PK é dita chave pública;
- para um par de chaves (SK, PK), a regra de ciframento é $e_{PK} \in \mathscr{E}$ e a regra de deciframento é $d_{SK} \in \mathscr{D}$.

Uma outra definição comumente utilizada é a seguinte: um criptossistema de chave pública é uma tupla de algoritmos $(Gen, \mathscr{E}, \mathscr{D})$ tais que

- Gen é o algoritmo de geração de chaves: recebe como parâmetro 1^k, onde k é um parâmetro de segurança, e devolve um par de chaves (SK, PK);
- $\mathscr{E}$ é o algoritmo de ciframento: dado um texto em claro x pertencente ao espaço de textos em claro $\mathscr{P}$, o algoritmo $\mathscr{E}_{PK}(x)$ devolve um texto cifrado y com tamanho polinomial $p(k)$;
- $\mathscr{D}$ é o algoritmo de deciframento: $\mathscr{D}_{SK}(y)$ devolve o texto em claro x ou o símbolo $\perp$ que representa um deciframento incorreto;
- o criptossistema precisa satisfazer o requisito de *correção*: para todo $x \in \mathscr{P}$ e todos os possíveis pares (PK, SK) devolvidos por Gen, tem-se que $\mathscr{D}_{SK}(\mathscr{E}_{PK}(x)) = x$.

Criptossistemas de chave pública requerem a existência de dois tipos de função: funções unidirecionais e funções (unidirecionais) com segredo. Informalmente, uma função é unidirecional se é fácil calculá-la mas difícil invertê-la. Uma função é com segredo se ela é uma função unidirecional em que a inversão é fácil quando se conhece um segredo. Quando a função induz uma permutação no domínio, usamos as expressões permutação unidirecional e permutação com segredo.

Definição 1. *Uma função* $f : \{0,1\}^* \to \{0,1\}^*$ *é dita* unidirecional *se as seguintes condições valem:*

1. ***Eficiência.*** *Existe um algoritmo A de tempo polinomial tal que* $A(x) = f(x)$*;*
2. ***Dificuldade de inversão.*** *Para todo algoritmo probabilístico* A' *de tempo polinomial, todo polinômio positivo* $p(\cdot)$ *e n suficientemente grande,*

$$\Pr[A'(f(X_n), 1^n) \in f^{-1}(f(X_n))] < \frac{1}{p(n)}.$$

A condição de eficiência indica a facilidade de calcular a função f. A condição de dificuldade de inversão indica que todo algoritmo probabilístico de tempo polinomial tem chance desprezível[2] de inverter f.

Definiremos agora funções (unidirecionais) com segredo.

Definição 2. *Uma função (unidirecional)* $f_r : \{0,1\}^* \to \{0,1\}^*$ *é dita* com segredo *se existe um par* $(r,s) \in \{0,1\}^* \times \{0,1\}^*$ *tal que as seguintes condições valem:*

1. ***Eficiência.*** *Existe um algoritmo F de tempo polinomial tal que* $F(x,r) = f_r(x)$*;*
2. ***Dificuldade de inversão.*** *Para todo algoritmo probabilístico* A' *de tempo polinomial, todo polinômio positivo* $p(\cdot)$ *e n suficientemente grande,*

$$\Pr[A'(r, f_r(X_n), 1^n) \in f_r^{-1}(f_r(X_n))] < \frac{1}{p(n)}.$$

3. ***Facilidade de inversão com segredo.*** *Existe um algoritmo polinomial determinístico, denotado por* F^{-1}*, tal que* $F^{-1}(F(x,r),s) = x$*.*

A seguir damos alguns exemplos dos primeiros criptossistemas de chave pública eas permutações com segredo por eles utilizadas.

Criptossistema 2. *RSA*

Geração de chaves. Selecione aleatoriamente dois números primos p,q tais que $p \neq q$. Seja $N = pq$ e $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente d,e tais que $de \equiv 1 \pmod{\phi(N)}$. A chave privada é o par $SK = (N,d)$ e a chave pública é o par $PK = (N,e)$.

Ciframento. Calcule $e_{PK}(x) = x^e \pmod{N}$.

Deciframento. Calcule $d_{SK}(y) = y^d \pmod{N}$.

O RSA é uma família de permutações com segredo indexada pelos pares (N,e) conforme a definição de criptossistema acima. O algoritmo F_{RSA}, ao receber como entrada $((N,e),x)$, devolve

$$\mathrm{RSA}_{N,e}(x) \stackrel{\mathrm{def}}{=} x^e \pmod{N}.$$

O algoritmo F_{RSA}^{-1} é idêntico ao algoritmo F_{RSA}.

[2]Uma função $\mu : \mathbb{N} \to \mathbb{R}$ é *desprezível* em n se para todo polinômio positivo $p(\cdot)$ e n suficientemente grande, $\mu(n) < 1/p(n)$.

Caso um adversário consiga fatorar N, é trivial calcular $d = e^{-1} \pmod{\phi(N)}$. Não se sabe se a fatoração de números inteiros pode ser reduzida à inversão de $\mathrm{RSA}_{N,e}$, mas os melhores algoritmos para inverter esta permutação com segredo são baseados em fatoração. De qualquer forma, acredita-se que a família RSA é de difícil inversão. Quanto à inversão com segredo, é fácil verificar que para todo $x \in \mathbb{Z}_N^*$,

$$F_{\mathrm{RSA}}^{-1}((N,d), F_{\mathrm{RSA}}((N,e),x)) = x^{ed} = x \pmod{N}.$$

Apresentamos agora o criptossistema de Rabin [Rabin, 1979].

Criptossistema 3. *Rabin*

Geração de chaves. Gere dois números primos grandes p,q tais que $p \neq q$ e $p,q \equiv 3 \pmod 4$, e calcule $N = pq$. A chave privada é o par $SK = (p,q)$ e a chave pública é $PK = N$.

Ciframento. Calcule $e_{PK}(x) = x^2 \pmod{N}$.

Decifrамento. Calcule $d_{SK}(y) = \sqrt{y} \pmod{pq}$.

Percebe-se pela descrição acima que o criptossistema de chave pública de Rabin é um caso particular do RSA com $e = 2$ e $d = 1/2$. Os índices passam a ser os valores de N e os segredos passam a ser os pares (p,q).

De forma geral, calcular raízes quadradas módulo um número composto N (em particular, um produto de dois primos grandes) é um problema para o qual não há um algoritmo que o resolva em tempo polinomial. Por outro lado, quando se conhece a fatoração de N, basta calcular as raízes quadradas módulo os fatores primos de N e aplicar o teorema chinês sobre o resto para combinar os resultados. No caso particular de $N = pq$, calcular $\sqrt{y} \pmod{pq}$ significa:

1. Calcular $w_1 = \sqrt{y} \pmod{p}$ (eficiente se $p \equiv 3 \pmod 4$);
2. Calcular $w_2 = \sqrt{y} \pmod{q}$ (eficiente se $q \equiv 3 \pmod 4$);
3. Usar o teorema chinês sobre o resto para resolver os quatro sistemas abaixo:

$$\begin{array}{lllll} (1) & x \equiv w_1 \pmod p & \text{e} & x \equiv w_2 \pmod q \\ (2) & x \equiv w_1 \pmod p & \text{e} & x \equiv -w_2 \pmod q \\ (3) & x \equiv -w_1 \pmod p & \text{e} & x \equiv w_2 \pmod q \\ (4) & x \equiv -w_1 \pmod p & \text{e} & x \equiv -w_2 \pmod q. \end{array}$$

Cada um desses sistemas possui uma solução única (pois p,q são primos distintos) que também é solução de $\sqrt{y} \pmod{pq}$.

Isto mostra que as operações definidas pelo Rabin podem ser executadas eficientemente. Mas como podemos analisar a sua segurança? Certamente as idéias de Shannon não são aplicáveis aqui: a chave pública contém *toda* a informação necessária para a recuperação da chave privada. O que esperamos, no entanto, é que efetivamente extrair esta informação seja difícil.

Provamos então a segurança do Rabin reduzindo um problema difícil à quebra do criptossistema. Usamos aqui uma noção bastante ingênua de segurança, onde "quebrar" um criptossistema significa decifrar um texto cifrado sem conhecer a chave privada

correta. Na próxima sessão discutimos mais profundamente por que esta é uma noção bastante ingênua de segurança e apresentamos definições mais robustas. Começamos nossa demonstração então mostrando que fatoração e extração de raízes quadradas modulares são problemas computacionalmente equivalentes no sentido de que um pode ser reduzido a outro e vice-versa. Depois mostramos, trivialmente, que a extração de raízes pode ser reduzida à quebra do Rabin (como definida acima).

O método descrito acima para calcular $\sqrt{y} \pmod{pq}$ é a base da redução do problema de extração de raízes quadradas modulares para o problema da fatoração. Para a redução da fatoração à extração de raízes modulares, considere o seguinte algoritmo probabilístico que recebe como entrada N, um número composto:

1. Selecione de forma aleatória e uniforme um número $r \in \{1, \ldots, N-1\}$;

2. Calcule $g = \gcd(N, r)$. Se $g > 1$, então gere g como saída e termine.

3. Seja $s = r^2 \pmod{N}$.

4. Execute o algoritmo de extração de raízes modulares com entrada s, obtendo como resultado r' tal que $(r')^2 \equiv s \pmod{N}$.

5. Calcule $g = \gcd(N, r - r')$. Se $g > 1$, então gere g como saída e termine.

Como o algoritmo imprime g, um divisor não-trivial de N, basta calcular a divisão N/g para obter outro divisor. A fatoração completa de N é obtida através da invocação recursiva do algoritmo passando como entrada tanto g quanto N/g.

Demonstramos agora a correção do algoritmo. Podemos assumir que r é coprimo com N. Assuma que $N = pq$ e portanto existem quatro raízes quadradas de s módulo N; sejam $\{\pm r, \pm t\}$ estas raízes. Observe que $\Pr[r' = \pm t] = 1/2$. Como $r^2 \equiv t^2 \pmod{N}$, temos que $r^2 - t^2 \equiv 0 \pmod{N}$ e portanto $(r+t)(r-t) \equiv 0 \pmod{N}$, ou seja, $r-t$ (ou $r+t$) é um fator de N com probabilidade $1/2$. Por conseguinte, o algoritmo encontra um fator de N com probabilidade *pelo menos* $1/2$. Portanto, após t execuções do algoritmo tem-se uma probabilidade $(1 - 2^{-k})$ de fatorar N.

Outra maneira de encarar este resultado é a seguinte: se existir um algoritmo A capaz de calcular raízes módulo N *para uma parcela não-desprezível* dos $x \in \mathbb{Z}_N$, então podemos construir o algoritmo S_A, como descrito acima, que consegue fatorar N em tempo polinomial. A observação de que A tem que ser capaz de calcular raízes (ou decifrar mensagens) numa parcela não desprezível dos $x \in \mathbb{Z}_N$ é necessária para garantir que S_A rode em tempo polinomial e é fonte de uma importante objeção a esta redução: há aqui uma sutil suposição sobre o espaço de mensagens a serem cifradas, especificamente em relação ao seu tamanho. Se o espaço de textos claros tiver cardinalidade maior ou igual a $\log|\mathbb{Z}_N|$, um algoritmo A' capaz de quebrar o esquema pode ser usado para criar um $S_{A'}$ que executa em tempo polinomial. No entanto, se o espaço de textos em claro tiver cardinalidade menor que $\log|\mathbb{Z}_N|$, um adversário A'' capaz de quebrar o esquema não gera um $S_{A''}$ que executa em tempo polinomial, ou seja, a redução acima não traz nenhuma garantia de segurança. Esta observação será discutida mais profundamente em §3.3.

3.2.3. Assinaturas Digitais

Além dos criptossistemas de chave pública, a criptografia assimétrica permite a construção de esquemas de assinaturas, os quais permitem verificar que uma determinada informação foi atestada por uma entidade. A seguir, definimos formalmente esquemas de assinatura.

Definição 3 (Esquemas de assinatura)**.** *Um esquema de assinatura é uma tupla* (G,S,V) *de algoritmos probabilísticos de tempo polinomial tais que:*

1. *Ao receber* 1^n *o algoritmo G (geração de chaves) retorna um par de strings;*
2. *Para todo todo par* (s,v) *imagem de* $G(1^n)$*, para todo* $\alpha \in \{0,1\}^*$*, os algoritmos S (assinatura) e V (verificação) satisfazem*

$$\Pr[V(v,\alpha,S(s,\alpha)) = 1] = 1\,.$$

Muitas vezes no texto usaremos a notação $V_v(\alpha,\sigma)$ e $S_s(\alpha)$ para $V(v,\alpha,\sigma)$ e $S(s,\alpha)$. Note que esta definição de esquemas de assinatura não leva em consideração requisitos particulares de segurança para assinaturas digitais. De fato, assim como em nossa primeira definição de criptossistema, esquemas trivialmente inseguros são concretizações válidas desta definição[3]. De forma ingênua, pode-se implementar um esquema de assinatura através de permutações com segredo de forma análoga aos criptossistemas de chave pública: o algoritmo de verificação como um "ciframento" com a chave pública e o algoritmo de assinatura como um "deciframento" com a chave privada correspondente. Por exemplo, pode-se usar a família de permutações RSA como um possível esquema de assinatura onde o algoritmo F_{RSA} é o algoritmo de verificação e o algoritmo F_{RSA}^{-1} é o algoritmo de assinatura.

Na próxima seção, discutiremos mais profundamente requisitos de segurança mais fortes para criptossistemas e esquemas de assinatura no contexto da criptografia assimétrica, especificamente discutindo por que definições ingênuas como as usadas na demonstração de segurança do Rabin são inadequadas e fornecendo opções mais robustas que vieram a se tornar padrão.

3.3. Noções Fortes de Segurança

Os anos 80 trouxeram uma necessidade de reavaliação da recente revolução impulsionada pela criptografia assimétrica. As ferramentas utilizadas até então para a avaliação de segurança de criptossistemas não se adequavam bem ao novo cenário: por exemplo, um criptossistema assimétrico nunca poderia atingir sigilo perfeito pois sempre existe uma relação matemática entre chaves públicas e privadas. A segurança neste cenário não está mais baseada na *existência* de informação relativa a textos em claro nos respectivos textos cifrados, mas em quão difícil é calcular (ou utilizar) esta informação.

Um profundo estudo de formalização da criptografia, fortemente baseado na teoria da complexidade, marcou esta década, impulsionado por trabalhos como os de [Goldwasser e Micali, 1984] [Goldwasser e Micali, 1982], [Yao, 1982], [Blum e Goldwasser, 1985], e [Micali et al., 1988]. Neste primeiro momento algumas

[3]De fato, $S_s(\alpha) = \alpha$ e $V_v(\alpha,\sigma) = 1, \forall \alpha = \sigma$ obedecem à definição 3.

noções fortes de segurança foram definidas, criptossistemas que as seguiam foram propostos e, finalmente, a relação entre as diferentes noções de segurança foi esclarecida.

Posteriormente, a preocupação se voltou para o tipo de ataque contra o qual "segurança" era definida. Nos primeiros trabalhos considerava-se sempre segurança contra adversários passivos. Modelos que davam mais poder aos adversários foram projetados de maneira a provar a segurança dos criptossistemas em situações mais diversas, culminando com a busca por criptossistemas resistentes a ataques completamente adaptativos.

Mostraremos os principais resultados teóricos desta época, bases da segurança demonstrável que se tem atualmente, fazendo um panorama abrangente dos resultados que se conheciam até o início da década de 90.

3.3.1. Ciframento, ou Como Jogar Pôquer Mentalmente

No final dos anos 70 a comunidade de criptografia se deparava com dois problemas aparentemente simples mas que não pareciam ser facilmente resolvíveis no ambiente de criptografia assimétrica como definida em [Diffie e Hellman, 1976]:

1. Como jogar pôquer pelo telefone de maneira segura. [Shamir et al., 1979] [Lipton, 1981];
2. Como enviar de forma segura um bit de informação.

Estes são problemas que, intuitivamente, deveriam ser resolvidos por criptossistemas seguros. No entanto, não se conseguia nenhuma evidência matemática de que, a partir das idéias de Diffie & Hellman, estes problemas poderiam ser resolvidos de forma satisfatória.

Estes dois problemas destacam que a abordagem sugerida pelo artigo seminal de [Diffie e Hellman, 1976] para ciframento assimétrico, e implementada nos esquemas RSA [Rivest et al., 1978] e Rabin [Rabin, 1979], sofre de duas limitações importantes [Goldwasser e Micali, 1984]:

1. f **ser uma função unidirecional não implica que ela seja de difícil inversão para um subconjunto do seu domínio.** Uma função unidirecional é uma função de difícil inversão para entradas escolhidas aleatoriamente do seu domínio. Isto não implica que esta mesma função será de difícil inversão quando se sabe que a mensagem cifrada é, por exemplo, um número de um domínio pequeno, ou a codificação de uma carta de baralho.
2. f **ser uma função unidirecional não implica a dificuldade de calcular informações parciais sobre sua entrada.** Dado um $f(x)$ aleatório, se f é unidirecional, sabemos que é difícil calcular o valor de x. Porém, esta definição por si só não diz nada sobre a (im)possibilidade de calcular informações parciais sobre x como sua paridade ou um bit específico.

Um criptossistema seguro não deve ser vulnerável em qualquer das situações acima. Entretanto, qualquer implementação diretamente baseada nas idéias de [Diffie e Hellman, 1976] sofre destes problemas: por exemplo, como todo texto em claro M tem um equivalente cifrado $E(M)$ único, predicados como *"$E(M)$ é par?"* são sempre facilmente calculáveis.

3.3.1.1. Noções Fortes de Segurança

Com estes problemas em mente, três noções fortes de segurança foram propostas, cada uma à sua maneira, adaptando as idéias de Shannon à criptografia de chave pública. Nas definições abaixo k é um parâmetro de segurança, os adversários A são algoritmos de tempo polinomial e $\mathfrak{C}$ é o criptossistema sendo analisado.

- **Segurança Polinomial.**[4] Esta definição de segurança deriva diretamente da idéia de *indistinguibilidade*: dado um par de mensagens e uma delas cifrada, um adversário não deve ser capaz de distinguir, em tempo polinomial, a qual das mensagens em claro o texto cifrado corresponde. Dado um par de mensagens $\{m_0, m_1\}$ arbitrariamente escolhidas, sejam $i \stackrel{r}{\leftarrow} \{0,1\}$, $E \leftarrow \mathscr{C}(1^k)$ e $\alpha \leftarrow E(m_i)$. O criptossistema $\mathfrak{C}$ é *polinomialmente seguro* se, para todos os adversários A e todo $c > 0$,
$$\Pr[A_k(E, m_0, m_1, \alpha) = m_i] < \frac{1}{2} + k^{-c}.$$
- **Segurança Semântica.**[5] Seja f uma função definida no espaço de mensagens. Informalmente, $f(m)$ representa informação sobre m. A noção de segurança semântica traduz o fato de que deveria ser difícil calcular o valor de qualquer $f(m)$ dado o texto cifrado $E(m)$: é uma tradução do conceito de "sigilo perfeito" de Shannon para o ambiente onde todos os participantes estão limitados por um número polinomial de passos. Dados os três jogos a seguir:
  - **Jogo 1.** Escolha $m \stackrel{r}{\leftarrow} M$. Neste jogo, A_1 tem que calcular o valor de $f(m)$ sem conhecer m.
  - **Jogo 2.** Escolha $m \stackrel{r}{\leftarrow} M$. Calcule um $\alpha \leftarrow E(m)$ e forneça ao adversário. Neste jogo, A_2 tem que calcular o valor de $f(m)$ conhecendo $E(m)$.
  - **Jogo 3.** Deixe A_3 escolher uma função f^* definida em M. Escolha $m \stackrel{r}{\leftarrow} M$, calcule um $\alpha \leftarrow E(m)$ e forneça ao adversário. A_3 tem que calcular o valor de $f^*(m)$.

  Seja A_i^* o evento em que o adversário vence o jogo i. Um criptossistema $\mathfrak{C}$ é semanticamente seguro se
  $$\Pr[A_3 *] < \Pr[A_1^*] + k^{-c},$$
  ou seja, o conhecimento da mensagem cifrada e a possibilidade de escolher a função que se quer calcular não devem trazer vantagem para o adversário.
- **Segurança "Yao".**[6] A definição de segurança de Yao é baseada em teoria da informação, adicionando a limitação polinomial nas operações das partes envolvidas. Esta definição é um pouco menos intuitiva que as duas anteriores, e exige um conjunto maior de conceitos auxiliares. Por este motivo apresentaremos uma explicação um tanto superficial da definição, referindo o leitor a [Yao, 1982] e [Micali et al., 1988] para uma definição mais precisa.

[4] Também conhecida como *indistinguibilidade de textos cifrados*. Mantivemos o nome original, como definido em [Goldwasser e Micali, 1984].

[5] Como definido em [Goldwasser e Micali, 1984].

[6] Como definido em [Yao, 1982].

A(lice) tem n^k mensagens que ela gostaria de transmitir a B(ob). Essas mensagens foram selecionadas de um conjunto $\mathscr{P}$ de possíveis mensagens com uma distribuição de probabilidade conhecida por A e B. Qual a quantidade mínima de bits que A precisa transmitir para B de maneira que este possa recuperar todas as n^k mensagens? Seja q_1 esta quantidade de bits. Agora imagine que B conhece um texto cifrado $E(m_i)$ para cada uma das n^k. Seja q_2 o número de bits necessários a B nesta situação. Certamente $q_2 \leqslant q_1$. Um criptossistema é *Yao-Seguro* se $q_2 = q_1$, ou seja, se o número de bits que A precisa enviar a B é o mesmo independentemente de B conhecer ou não os textos cifrados.

Estas definições de segurança têm em comum a preocupação com a proteção de toda informação referente ao texto em claro. Não estava muito claro na época qual seria a definição correta de segurança, até o trabalho de [Micali et al., 1988], onde os autores demonstram o teorema a seguir.

Teorema 3. *As noções de segurança polinomial, segurança semântica e segurança "Yao" são equivalentes [Micali et al., 1988].*

Este resultado trouxe consigo um aumento na confiança de que estas noções de segurança eram, de fato, noções "corretas".

Ciframento Determinístico. Note que um esquema de ciframento determinístico, como o RSA e o Rabin, nunca pode atingir estas noções fortes de segurança. Como os três tipos de segurança são equivalentes, basta observar que um criptossistema determinístico nunca pode ser polinomialmente seguro: dados uma função de ciframento E, dois textos em claro (m_0 e m_1), e $\alpha = E(m_i)$ para $i \xleftarrow{r} \{0,1\}$, um adversário pode sempre aplicar E a m_0 e m_1 e comparar o resultado a α. Se E é uma função determinística, o adversário sempre vai ser capaz de decidir corretamente a quem α corresponde. Veja que se E fosse aleatorizada este procedimento não funcionaria pois existiria um número exponencialmente grande de possíveis textos cifrados para cada m_i: a probabilidade de o adversário aplicar E a m_i e obter α seria desprezível.

3.3.1.2. Ciframento Probabilístico

Goldwasser & Micali, responsáveis pelas duas primeiras definições de segurança apresentadas acima, propuseram um novo paradigma de ciframento: o *ciframento probabilístico*. Em [Goldwasser e Micali, 1984] eles argumentam a necessidade de quebrar a bijeção entre textos em claro e cifrados para que se consigam esquemas realmente seguros (i.e., segundo as definições em §3.3.1.1). A ferramenta para isto é, ainda segundo os autores, o uso de funções de ciframento que funcionem de maneira probabilística e que possam, para uma mensagem m_i qualquer, gerar uma quantidade exponencialmente grande de diferentes textos cifrados $E(m_i)$, todos válidos. O deciframento continua sendo determinístico e $\Pr[D(E(m_i)) = m_i] = 1$.

Como um exemplo deste paradigma de ciframento probabilístico, os autores propuseram um criptossistema de chave pública e provaram que ele é polinomialmente seguro. Esta proposta utiliza, em lugar de funções unidirecionais com segredo, a idéia

de *predicados não-aproximáveis com segredo*. Basicamente um predicado $B : \{0,1\}^* \to \{0,1\}$ é *não-aproximável* e *com segredo* se qualquer pessoa pode escolher x e y tais que $B(x) = 1$ e $B(y) = 0$, mas só quem conhece o segredo é capaz de, dado um z, calcular $B(z)$.

Digamos que Bob escolha um predicado não-aproximável com segredo $B^* : \{0,1\}^k \to \{0,1\}$ tal que ele conheça o segredo s^* correspondente. Bob pode então publicar B^* e ele passará a servir como sua chave pública, enquanto s^* será sua chave privada. Suponha agora que Alice quer enviar a Bob um bit $b \in \{0,1\}$ de maneira segura. Alice pode escolher um x aleatório tal que $B^*(x) = b$ e enviar para Bob. Bob, conhecendo s^*, será capaz de calcular $B^*(x)$ e recuperar o valor de b; qualquer outra pessoa que intercepte a mensagem, no entanto, será incapaz de calcular b, pois isto implicaria o cálculo de $B^*(x)$.

Este esquema simples resolve o problema de como enviar um bit de forma segura. Agora podemos generalizar esta abordagem para uma mensagem de tamanho arbitrário simplesmente representando-a como uma seqüência de bits $M = m_0 m_1 m_2 \ldots m_{n-1}$, onde m_i é o i-ésimo bit de M. Alice pode cifrar M bit a bit, utilizando o procedimento descrito acima. O criptossistema `GM` é então definido como:

Criptossistema 4. *Goldwasser-Micali* (`GM`)

Geração de chaves. Selecione aleatoriamente um predicado não-aproximável e com segredo B^* e um segredo s^* associado a ele. A chave pública é (a descrição de) B^*, e a chave privada é s^*.

Ciframento. Seja $M = m_0 m_1 m_2 \ldots m_{n-1}$ a mensagem a ser cifrada. Para cada m_i escolha aleatoriamente x_i tal que $B^*(x_i) = m_i$. O texto cifrado é $C = x_0 || x_1 || \ldots || x_{n-1}$.

Deciframento. Seja $C = x_0 || x_1 || \ldots || x_{n-1}$ o texto a ser decifrado. Utilizando s^* calcule $m'_i = B^*(x_i)$ para cada i. A mensagem original é $M = m_0 || m_1 || \ldots || m_{n-1}$.

Este criptossistema foi proposto em [Goldwasser e Micali, 1982] utilizando o problema do resíduo quadrático como predicado. Sua segurança polinomial foi provada neste mesmo artigo, tornando-o assim o primeiro criptossistema provado seguro sob uma noção forte de segurança. Apresentamos aqui este esquema principalmente como uma referência histórica, mas analisaremos mais profundamente uma variação deste cuja demonstração de segurança é mais simples e didática.

3.3.1.3. Ciframento Probabilístico Baseado em Permutações com Segredo

A idéia anterior de Goldwasser & Micali pode ser adaptada para utilizar *permutações com segredo*. Lembramos que, intuitivamente, uma permutação com segredo é uma função unidirecional com segredo que induz uma permutação no seu domínio (para mais detalhes, consulte §3.2.2).

Já discutimos que o fato de uma função (resp. permutação) ser de difícil inversão não implica que informações parciais sobre a entrada não possam ser calculadas a partir

do resultado. Para tornar explícito o que é difícil de ser calculado sobre um x arbitrário dado apenas o conhecimento de $f(x)$ (resp. $p(x)$), definimos a noção de *hard-core* de uma função (resp. permutação).

Definição 4 (Hard-Core de uma Função)**.** *Seja* $b : \{0,1\}^* \to \{0,1\}^*$ *uma função calculável em tempo polinomial tal que para todo* $|x| = |y|$ *tem-se* $|b(x)| = |b(y)|$*. Seja* $l(n) \stackrel{def}{=} |b(1^n)|$*. A função b é chamada de hard-core da função f se, para todo algoritmo polinomial A, todo polinômio positivo* $p(.)$ *e todo n suficientemente grande,*

$$\text{abs}(\Pr[A(f(X_n), b(X_n)) = 1] - \Pr[A(f(X_n), R_{l(n)}) = 1]) < \frac{1}{p(n)}, \tag{1}$$

onde X_n *e* $R_{l(n)}$ *são duas variáveis aleatórias independentes uniformemente distribuídas em* $\{0,1\}^n$ *e* $\{0,1\}^{l(n)}$ *respectivamente.*

Definimos então criptossistema abaixo.

Criptossistema 5. *Um esquema simples de ciframento com chave pública* (`CS-2`)

Geração de chaves. Selecione aleatoriamente uma permutação p^* e um segredo s^* associado a ela; a chave pública é (a descrição de) p^* e a chave privada é s^*.

Ciframento. Seja σ o bit a ser cifrado e seja $b^*(.)$ um *hard-core* de p^*. Selecione aleatoriamente um elemento r do domínio de p^* e calcule o texto cifrado $C = (p^*(r), \sigma \oplus r)$.

Decifrамento. Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Utilizando s^*, calcule $r' = p^{-1}(\gamma_1)$; o bit original é $\sigma = \gamma_2 \oplus r'$.

Vamos apresentar uma demonstração simples da segurança deste criptossistema baseada nas propriedades da permutação com segredo e de seus *hard-cores*.

Teorema 4. *O criptossistema* `CS-2` *é polinomialmente seguro.*

Demonstração. Lembrando a definição de segurança polinomial, precisamos demonstrar que, dadas duas mensagens (m_0, m_1) e uma delas cifrada (γ), é difícil distinguir qual das duas foi cifrada. Lembre também que a noção de "difícil" aqui significa "algo que, em tempo polinomial, só pode ser feito com probabilidade desprezível no tamanho da entrada". Como estamos cifrando apenas um bit, basta provarmos que, dada a chave α, é difícil distingüir o ciframento de 0 do ciframento de 1, i.e., é difícil distinguir as distribuições $E_\alpha(1) = (p_\alpha(r), 1 \oplus b(r))$ de $E_\alpha(0) = (p_\alpha(r), 0 \oplus b(r))$, para r aleatório. Isto é, para todo algoritmo polinomial A tem-se

$$\text{abs}(\Pr[A(p_\alpha(r), 1 \oplus b(r))] - \Pr[A(p_\alpha(r), 0 \oplus b(r))]) < \frac{1}{\text{poli}(n)}, \tag{2}$$

para $r \stackrel{r}{\leftarrow} \{0,1\}^n$, todo polinômio poli(.) e n suficientemente grande. Intuitivamente, isto segue diretamente do fato de $b(.)$ ser um *hard-core* de p_α. Relembrando a equação 1 (adaptada à permutação p), para qualquer algoritmo A,

$$\text{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), R_{l(n)}) = 1]) < \frac{1}{\text{poli}(n)}, \tag{3}$$

para todo polinômio poli(.) e todo n suficientemente grande. Como no nosso cenário $l(n) = 1$ (o ciframento ocorre bit-a-bit), temos

$$\begin{aligned}\Pr[A(p(X_n), R_{l(n)}) = 1] &= \frac{1}{2}\Big(\Pr[A(p(X_n),1) = 1] + \Pr[A(p(X_n),0) = 1]\Big) \\ &= \frac{1}{2}\Big(\Pr[A(p(X_n), b(X_n)) = 1] + \Pr[A(p(X_n), b(X_n)\oplus 1) = 1]\Big).\end{aligned}$$

Substituindo então na eq. (3) tem-se

$$\begin{aligned}\mathrm{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), R_{l(n)}) = 1]) &= \\ \mathrm{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), b(n)\oplus 1) = 1])/2 &< \frac{1}{\mathrm{poli}(n)},\end{aligned}$$

que implica, trivialmente, a eq. (2), demonstrando a segurança polinomial de `CS-2`. □

Algumas observações cabem em relação a esta demonstração. A primeira é que ela é propositadamente simples por deslocar muito da sua complexidade para definições auxiliares: não especificamos como construir *hard-cores* de permutações (de fato, nem especificamos uma permutação a ser utilizada), e a segurança do criptossistema não depende diretamente da estrutura dos *hard-cores* ou da permutação. Em segundo lugar, note que esta demonstração de segurança, na verdade, não depende diretamente de o ciframento operar bit-a-bit, mas sim de que ele opere em blocos do contradomínio do *hard-core* $b(.)$ em questão. Usamos o exemplo em que este contradomínio é $\{0,1\}$ por facilidade de apresentação, mas nada impede que outros *hard-cores* sejam utilizados.

Tendo estas duas observações em mente, apresentamos a seguir uma versão do `CS-2` baseada no RSA, que chamaremos de `RSA-Aleatorizado` na discussão que segue, e que é demonstravelmente segura.

Lembramos que a permutação induzida pela Função-RSA é

$$F_{\mathrm{RSA}}((N,e),x) = x^e \pmod N.$$

Suponha agora que F_{RSA} é de fato uma permutação com segredo (isto é, que ela é de difícil inversão) e, adicionalmente, que o cálculo dos m bits menos significativos de x seja um *hard-core* de F_{RSA} para $|N| = k$. Podemos então construir um esquema análogo ao anterior que funciona em blocos de m bits da seguinte maneira:

Criptossistema 6. *Instanciação do `CS-2` baseada no RSA* (`RSA-Aleatorizado`)

Geração de chaves. Selecionam-se aleatoriamente dois primos de n bits, p e q. Seja $N = pq$; note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par (e,d) tal que $ed \equiv 1 \mod \phi(N)$. A chave pública será o par (N,e) e a chave privada será o par (N,d).

Ciframento. Seja $M \in \{0,1\}^m$ a mensagem a ser cifrada. Escolha $r \xleftarrow{r} \{0,N-1\}$ e calcule o texto cifrado $C = (r^e \pmod M, \mathrm{BMS}_m(r) \oplus M)$, onde $\mathrm{BMS}_m(r)$ representa os m bits menos significativos de r.

Deciframento. Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Através de (N,d) calcula-se cada $m'_i = \gamma_2 \oplus (\gamma_1^d \pmod N)$.

Teorema 5. *Supondo que a extração dos m bits menos significativos da entrada seja um hard-core de F_{RSA}, o criptossistema* `RSA-Aleatorizado` *é polinomialmente seguro.*

A prova de segurança desta versão do RSA pode ser construída de forma análoga à segurança do `CS-2` e é deixada como exercício para o leitor.

Sabe-se que para $m \in O(\log n)$ a extração dos m bits menos significativos é um *hard-core* de F_{RSA} [Chor e Goldreich, 1985]. Contudo, para valores assintoticamente maiores de m a dificuldade da extração dos m bits menos significativos ainda é um problema em aberto. Em particular, seria interessante usar algo próximo de $m = n/2$.

É importante destacar aqui que *o uso do RSA simples não é seguro* (uma vez que ele é determinístico). O uso do `RSA-Aleatorizado` para $m \in O(\log(n))$ é demonstravelmente seguro supondo que a Função-RSA induza uma permutação com segredo no seu domínio, todavia é ineficiente. O uso do `RSA-Aleatorizado` para $m = n/2$ é *possivelmente* seguro, mas este ainda é um problema em aberto. Logo, é preferível utilizar este último que possivelmente é seguro, dada uma conjetura razoável (que a extração de $n/2$ bits da entrada é um *hard-core* da Função-RSA), a utilizar o RSA simples que certamente é *inseguro*.

3.3.1.4. Ciframento Probabilístico Eficiente

Em [Blum e Goldwasser, 1985] os autores apresentaram o primeiro criptossistema seguro realmente eficiente. As propostas existentes até então podem ser consideradas resultados de "plausibilidade", extremamente importantes em termos teóricos, mas que não traziam uma opção que fosse utilizável na prática.

Com este esquema os autores atingem eficiência comparável, e às vezes até superior, à do RSA. Ele, de alguma maneira, herda o conceito de utilizar um *hard-core* de uma permutação com segredo, como nos esquemas `CS-2` e `RSA-Aleatorizado`, mas usa um truque engenhoso para evitar a expansão de dados observada nestes esquemas: através da noção de *geradores pseudo-aleatórios*, algoritmos capazes de, a partir de uma *semente* aleatória, gerar uma seqüência de bits "pseudo-aleatória", indistinguível em tempo polinomial de uma seqüência verdadeiramente aleatória desde que a semente seja desconhecida. Sendo assim, para cifrar uma mensagem M de tamanho $|M| = n$, usa-se uma semente de tamanho k e a partir dela gera-se uma seqüência de n bits pseudo-aleatórios usados para cifrar M.

Para facilitar a apresentação, manteremos o padrão de primeiro apresentar uma versão abstrata do esquema seguida de uma versão que apresenta a instanciação sugerida pelos autores. Na apresentação a seguir usamos a notação

$$p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x)) \quad \text{e} \quad p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x)).$$

Criptossistema 7. *Blum-Goldwasser Abstrato* (BG−A)

Geração de chaves. Selecione uma permutação p_α com segredo associado s^*. A chave pública é (a descrição de) p_α e a chave privada é s^*.

Ciframento. Seja $M = m_0 m_1 \ldots m_{n-1}$ a mensagem a ser cifrada e seja $b^*(.) : \{0,1\}^k \to \{0,1\}$ um *hard-core* de p_α. Selecione aleatoriamente um elemento r do domínio de p_α e calcule o texto cifrado $C = (p^n(r), M \oplus G^n(r))$, onde

$$G_\alpha^n \stackrel{\text{def}}{=} b(r)||b(p_\alpha(r))||\ldots||b(p_\alpha^{n-1}(r)).$$

Deciframento. Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Utilizando s^*, calcule

$$M = \gamma_2 \oplus G_\alpha^{(|\gamma_2|)}(p_\alpha^{-|\gamma_2|}(\gamma_1)).$$

Como pode-se observar a seguir, o esquema é correto:

$$\begin{aligned} D_{s^*}(E_\alpha(M)) &= D_{s^*}(p_\alpha^{|M|}(r), M \oplus G_\alpha^{(|M|)}(r)) \\ &= (M \oplus G_\alpha^{|M|}(r)) \oplus G_\alpha^{|M|}(p_\alpha^{-|M|}(p_\alpha^{|M|}(r))) \\ &= M \oplus G_\alpha^{|M|}(r) \oplus G_\alpha^{|M|}(r) = M. \end{aligned}$$

Uma mensagem M de n bits é cifrada como uma string de $(n+k)$ bits neste esquema, gerando uma expansão de mensagem muito pequena e que tende a ser desprezível com o crescimento de n (afinal, k é constante). A segurança desta versão abstrata do esquema pode ser demonstrada com base nas propriedades das primitivas utilizadas.

Demonstração. Novamente, escolhemos a noção de segurança polinomial. É suficiente mostrar então que, para um par de mensagens arbitrárias $(M'_n, M''_n) \in \{0,1\}^{l(n)} \times \{0,1\}^{l(n)}$, as distribuições

$$\begin{aligned} D'_n &\stackrel{\text{def}}{=} (\alpha, p_\alpha^{|l(n)|}(X_n), M'_n \oplus G_\alpha^{l(n)}(X_n)) \text{ e} \\ D''_n &\stackrel{\text{def}}{=} (\alpha, p_\alpha^{|l(n)|}(X_n), M''_n \oplus G_\alpha^{l(n)}(X_n)), \end{aligned}$$

são polinomialmente indistinguíveis, onde X_n é uma variável uniformemente distribuída em $\{0,1\}^n$.

Na construção da nossa demonstração usaremos, sem demonstrar, o seguinte lema:

Lema 1. *Sejam k e n inteiros. Para toda permutação com segredo p_α e todo $x \in D_i$, seja $G_\alpha^n(.)$ como definido anteriormente:*

$$G_\alpha^n(x) \stackrel{def}{=} b(x)||b(p_\alpha(x))||\ldots||b(p_\alpha^{n-1}(x)).$$

Seja X_n uma variável aleatória uniformemente distribuída no domínio de p_α. Então, para todo $l(n) \in O(\text{poli}(n))$ as distribuições

$$\{(G_\alpha^{l(n)}(X_n), p_\alpha^{l(n)}(X_n)))\}_{n\in\mathbb{N}} \qquad e \qquad \{(U_{l(n)}, p_\alpha^{l(n)}(X_n)))\}_{n\in\mathbb{N}}$$

são indistinguíveis em tempo polinomial.

Este lema nos mostra basicamente que $G_\alpha^n(.)$ é um gerador de bits pseudo-aleatórios: mesmo recebendo todas as informações públicas disponíveis, nenhum algoritmo polinomial consegue distinguir a saída de $G_\alpha^{l(n)}(.)$ de uma variável uniformemente distribuída $U_{l(n)}$.

Seguimos então definindo as distribuições

$$D_n \stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M_n \oplus G_\alpha^{l(n)}(X_n)) \qquad \text{e} \qquad R_n \stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M_n \oplus U_{l(n)}),$$

onde $U_{l(n)}$ é uma variável uniformemente distribuída em $\{0,1\}^{l(n)}$.

Lema 2. *Para toda seqüência de M_i's, as distribuições D_i e R_i são indistinguíveis.*

Demonstração. A indistinguibilidade de $\{D_n\}_{n\in\mathbb{N}}$ e $\{R_n\}_{n\in\mathbb{N}}$ segue do lema 1. □

Para todos os possíveis pares $(M'_n, M''_n) \in \{0,1\}^{l(n)} \times \{0,1\}^{l(n)}$, definimos de forma análoga as distribuições

$$\begin{aligned}
D'_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M'_n \oplus G_\alpha^{l(n)}(X_n)) \\
D''_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M''_n \oplus G_\alpha^{l(n)}(X_n)) \\
R'_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M'_n \oplus U_{l(n)}) \\
R''_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M''_n \oplus U_{l(n)}).
\end{aligned}$$

Para provar que D'_n e D''_n são indistinguíveis, simplesmente aplicamos o lema 2 a (D'_n, R'_n) e notamos que elas são indistinguíveis. Analogamente, (D''_n, R''_n) são indistinguíveis. É fácil enxergar que R'_n e R''_n têm a mesma distribuição. Logo, D'_n e D''_n são (polinomialmente) indistinguíveis. □

Instanciando. Em [Blum e Goldwasser, 1985] os autores apresentam este esquema utilizando a operação de elevar ao quadrado módulo um inteiro de Blum como a permutação com segredo da definição de `BG-A`. Um inteiro de Blum é todo número n igual ao produto de dois primos p, q tais que $p \equiv q \equiv 3 \pmod 4$. Supondo que fatorar inteiros de Blum seja um problema difícil, a operação de elevar ao quadrado induz uma permutação com segredo sobre os resíduos quadráticos $\pmod n$. Esta instanciação do `BG-A`, referida aqui como `BG`, tem eficiência comparável à do RSA.

Criptossistema 8. *Blum-Goldwasser* (BG)

Geração de chaves. Selecione aleatoriamente dois primos P e Q de n bits tais que $P \equiv Q \equiv 3 \pmod 4$. Seja $N = PQ$. Calcule então:

$$\begin{aligned} d_P &= ((P+1)/4)^{l(n)} \pmod{(P-1)} \quad (\in \{0,\ldots,P-2\}) \\ d_Q &= ((Q+1)/4)^{l(n)} \pmod{(Q-1)} \quad (\in \{0,\ldots,Q-2\}) \\ c_P &= Q(Q^{-1} \pmod P) \quad (\in \{0,\ldots,N-Q\}) \\ d_P &= P(P^{-1} \pmod Q) \quad (\in \{0,\ldots,N-P\}). \end{aligned}$$

A chave pública é N e a chave privada (P,Q,c_P,d_P,c_Q,d_Q).

Ciframento. Para cifrar uma mensagem $M \in \{0,1\}^{l(n)}$ usando a chave N, selecione $s_0 \xleftarrow{r} \{1,\ldots,N\}$. Para $i = 1,\ldots,l(n)+1$, calcule

$$s_i \leftarrow s_{i-1}^2 \pmod N \qquad \text{e} \qquad b_i = \text{BMS}(s_i),$$

onde $\text{BMS}(s)$ representa o bit menos significativo de s. O texto cifrado será $(s_{l(n)+1}, M \oplus (b_1||b_2||\ldots||b_{l(n)}))$.

Deciframento. Seja $C = (\gamma_1,\gamma_2)$ o texto a ser decifrado e seja (P,Q,c_P,d_P,c_Q,d_Q) a chave privada. Comece por recuperar o valor de s_1 que é a $2^{l(n)}$-ésima raiz módulo N. Esta extração é tão eficiente quanto a extração de uma raiz quadrada:

$$\begin{aligned} s_P &= \gamma_1^{d_P} \pmod P \\ s_Q &= \gamma_1^{d_Q} \pmod P \\ s_1 &= c_P \cdot s_P + c_Q \cdot s_Q \pmod N. \end{aligned}$$

Lembrando que $\gamma_1 = s_1^{2^{l(n)}} \pmod N$, o resultado segue do TCR. Para todo $i = 1,\ldots,l(n)$, calcule $b_i = \text{BMS}(s_i)$ e $s_{i+1} \leftarrow s_i^2 \pmod N$. O texto em claro é $M = \gamma_2 \oplus b_1||b_2||\ldots||b_{l(n)}$.

É fácil observar que a eficiência deste esquema é comparável à do RSA. O ciframento requer $l(n)+1$ multiplicações modulares e o deciframento requer $l(n)+2$ multiplicações e duas exponenciações modulares. Supondo que exponenciações modulares têm um custo de $O(n)$ multiplicações, todo o processo de ciframento tem custo próximo de $2l(n)+3n$ multiplicações modulares. A versão aleatorizada do RSA apresentada aqui tem custo de $\lceil l(n)/n \rceil$ exponenciações modulares, aproximadamente $3l(n)$ multiplicações modulares (se $e = 3$, reduz-se a até $1{,}5l(n)$ multiplicações modulares).

A segurança do criptossistema BG é definida pelo teorema a seguir.

Teorema 6. *O criptossistema BG é seguro se a fatoração de inteiros de Blum for um problema difícil [Blum e Goldwasser, 1985].*

Enquanto não se sabe se a quebra do RSA é equivalente à fatoração, e se a conjetura que sustentaria a versão RSA-Aleatorizada-$n/2$ é verdadeira, a segurança

do BG deriva diretamente da dificuldade da fatoração, sem nenhuma suposição adicional, mantendo uma eficiência comparável à do RSA.

3.3.2. Ciframento e Adversários Ativos

Todas as demonstrações de segurança apresentadas até agora referem-se a ataques realizados de forma passiva: um adversário de posse de textos cifrados e informações públicas tenta descobrir alguma informação sobre as mensagens originais. Existem outros tipos de ataque, no entanto, onde adversários têm a capacidade de interagir com os usuários do sistema, requisitar deciframento de mensagens arbitrárias, e onde as demonstrações de segurança apresentadas aqui deixam de ser válidas. Estes cenários de ataque são importantes, e podem aparecer em muitas situações reais. Como lidar com eles? Basicamente, utilizamos três parâmetros para classificar os tipos de ataque:

1. *Que informação é conhecida pelo adversário?* Ele tem acesso apenas a textos cifrados ou a pares de texto em claro e cifrado?
2. *O adversário tem o poder de escolher a informação a que tem acesso?* Por exemplo, o adversário pode escolher o texto em claro a cuja versão cifrada terá acesso?
3. Em caso de poder escolher, *o adversário pode fazê-lo de forma adaptativa?* Ou seja, o adversário pode escolher os textos que deseja (de)cifrar durante o ataque ou precisa escolher todos *a priori*?

Dependendo então das respostas às perguntas acima obtemos os seguintes tipos de ataques: ataques de texto cifrado conhecido[7], ataques de texto em claro conhecido, ataques de texto em claro escolhido, ataques de texto cifrado escolhido, ataques adaptativos de texto em claro escolhido (CPA)[8] e ataques adaptativos de texto cifrado escolhido (CCA)[9].

Muitos destes cenários de ataque podem parecer artificiais, afinal por que algum usuário que valoriza o sigilo de suas mensagens se disporia a decifrar mensagens arbitrárias para um possível adversário (possibilitando assim um ataque adaptativo)? A verdade é que cenários em que isto pode ocorrer, por mais estranho que pareça, acontecem na prática. Portanto, é importante destacar a relevância da segurança contra estes ataques: eles podem surgir em muitas das situações onde esquemas de ciframento são utilizados.

Demonstrações por jogos. Na construção de provas de segurança contra ataques ativos é necessário, de alguma maneira, representar a possível interação entre usuário e adversário. A maneira mais difundida e provavelmente mais intuitiva de modelar esta interação é através de jogos. Digamos que queremos demonstrar que um certo criptossistema é CCA-seguro. Como sempre, utilizamos a noção de segurança polinomial. Propomos então o seguinte jogo entre duas partes: o adversário A, que tenta quebrar a segurança polinomial do criptossistema, e o simulador S_A que tenta resolver algum problema difícil caso A consiga quebrar o criptossistema. Para facilitar a discussão, definimos A por dois

[7] As demonstrações de segurança que vimos até agora tratam deste tipo de segurança, a noção de segurança mais fraca dentre as definidas.

[8] Do inglês *Chosen-Plaintext Attack*. Geralmente este acrônimo se refere a ataques adaptativos.

[9] Do inglês *Chosen-Ciphertext Attack*. Geralmente este acrônimo se refere a ataques adaptativos.

algoritmos (A_1, A_2) de modo que A_1 recebe os parâmetros do sistema e gera o par de mensagens-desafio, e posteriormente A_2 recebe este par de mensagens, tendo uma delas sido cifrada, e deve ser capaz de descobrir qual das duas foi cifrada. O jogo prossegue então da seguinte forma:

1. S_A gera os parâmetros do sistema `params` e o par de chaves (PK, SK), e executa $A(\texttt{params}, PK)$; se o ataque for de texto cifrado escolhido, A deve ter acesso a um oráculo de deciframento (simulado por S_A);
2. A_1 executa um número polinomial de passos e devolve um par de mensagens (m_0, m_1);
3. S_A escolhe aleatoriamente um $i \in \{0,1\}$ e calcula $\alpha = E_{SK}(m_i)$;
4. S_A executa $A_2(\texttt{params}, PK, m_0, m_1, \alpha)$; se o ataque for *adaptativo* de texto cifrado escolhido, A_2 também precisa ter acesso a um oráculo de deciframento (novamente, simulado por S_A).
5. A_2 executa um número polinomial de passos e retorna um chute i'.

A vence o jogo se $i = i'$.

Levou muito tempo até que os pesquisadores conseguissem propor um esquema prático e demonstravelmente seguro contra adversários adaptativos. Apresentamos, a seguir, o primeiro tal criptossistema, originalmente proposto em [Cramer e Shoup, 1998]. A sua segurança depende de um problema, menos padrão do que a fatoração ou o logaritmo discreto, chamado de problema de Diffie-Hellman de Decisão. Definimos brevemente este problema antes de prosseguir com a apresentação do `Cramer-Shoup`.

Partindo do protocolo de [Diffie e Hellman, 1976] para estabelecimento de chaves podemos propor dois problemas aparentemente difíceis:

> **O Problema de Diffie-Hellman Computacional (`DHC`).** Esta versão é exatamente equivalente à quebra do protocolo de estabelecimente de chaves. Seja $\mathbb{G}$ o grupo onde o problema está definido e g um gerador desse grupo. Dada a tupla $(g, g^a, g^b) \in \mathbb{G}^3$, o adversário deve calcular $g^{ab} \in \mathbb{G}$.
>
> **O Problema de Diffie-Hellman de Decisão (`DHD`).** Nesta versão do problema, o adversário precisa apenas *reconhecer* se o valor recebido é o valor correto: seja $\mathbb{G}$ o grupo onde o problema está definido e g um gerador desse grupo. Dada a tupla $(g, g^a, g^b, g^c) \in \mathbb{G}^3$, o adversário deve responder se $g^{ab} \equiv g^c$.

Ambos os problemas, assim como o protocolo de estabelecimento de chaves, dependem da dificuldade do cálculo de logaritmos discretos. No entanto, de maneira semelhante à relação entre RSA e fatoração, não existe qualquer demonstração de que `DHD` ou `DHC` sejam equivalentes ao problema do logaritmo discreto. Inclusive, no caso geral, o `DHD` é estritamente mais fácil que o `DHC`, existindo inclusive uma área de criptografia que estuda as propriedades de grupos onde o `DHD` é fácil e o `DHC` é difícil, os chamados *Gap Diffie-Hellman Groups*, e propõe criptossistemas baseados nesta diferença (e.g. [Boneh e Franklin, 2003]). No entanto, quando se define o problema em $\mathbb{Z}_n^*$, para valores apropriados de n, não se sabe resolver `DHD` ou `DHC` sem calcular o logaritmo discreto.

O criptossistema de [Cramer e Shoup, 1998] é baseado no `ElGamal`, que pode ser demonstrado seguro contra adversários passivos caso o `DHD` seja difícil. Mais do que isso, ele é dotado de alguma complexidade extra que o torna seguro também contra adversários adaptativos. Faremos uma apresentação progressiva do `Cramer-Shoup` baseada no tratamento dado por [Katz, 2004]: começaremos propondo uma versão ligeiramente alternativa do `ElGamal` básico, que não muda significativamente suas propriedades de segurança (continua semanticamente seguro contra adversários passivos), mas introduz uma técnica de demonstração que será crucial na apresentação do `Cramer-Shoup` completo. Considere então a seguinte versão do criptossistema ElGamal:

Criptossistema 9. *ElGamal-A*

Geração de Chaves. Seja $\mathbb{G}$ um grupo de ordem n onde o `DHD` é difícil e sejam g_1, g_2 geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x, y \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule $z = g_1^x g_2^y \pmod{n}$. A chave pública é (g_1, g_2, z) e a chave privada é (x, y).

Ciframento. Seja $M \in \{1, \ldots, n-1\}$ a mensagem a ser cifrada e seja (g_1, g_2, z) a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule o texto cifrado $(g_1^r, g_2^r, z^r M)$.

Deciframento. Seja (γ_1, γ_2, C) o texto cifrado e seja (x, y) a chave privada do usuário. Calcule $M = C/\gamma_1^x \gamma_2^y$.

Teorema 7. *`ElGamal-A` é polinomialmente seguro contra adversários passivos se o `DHD` é difícil em* $\mathbb{G}$.

Demonstração. Suponha que um algoritmo $A = (A_1, A_2)$ consegue quebrar a segurança polinomial de `ElGamal-A`. Construímos então o algoritmo S_A que consegue resolver o `DHD`. S_A recebe como entrada uma tupla (g_1, g_2, g_3, g_4) e deve ser capaz de descobrir se ela é uma tupla aleatória ou se é uma tupla de Diffie-Hellman. S_A prossegue então da seguinte forma:

1. escolhe $x, y \xleftarrow{r} \{0, \ldots, n-1\}$;
2. calcula $h = g_1^x g_2^y$;
3. define $PK \stackrel{\text{def}}{=} (g_1, g_2, z)$ como chave pública;
4. executa $(m_0, m_1) \leftarrow A_1(PK)$, recebendo as duas mensagens-desafio;
5. escolhe $i \xleftarrow{r} \{0, 1\}$;
6. calcula $C = (g_3, g_4, g_3^x g_4^y . m_b)$;
7. executa $i' \leftarrow A_2(PK, C)$
   - se $i = i'$, S_A devolve `ACEITA`;
   - caso contrário, S_A devolve `FALHA`.

A estrutura da prova será basicamente a seguinte: primeiro, mostraremos que, se S_A receber uma tupla de Diffie-Hellman como entrada ele simula perfeitamente um ataque real para A e detecta a tupla com probabilidade relacionada à probabilidade de A quebrar o criptossistema; por outro lado, se S_A recebe uma tupla aleatória, A não consegue qualquer informação sobre i e, na melhor das hipóteses, tem 50% de chance de acerto.

Lema 3. *Se S_A recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real.*

Prova do lema 3. Se (g_1, g_2, g_3, g_4) é uma tupla de Diffie-Hellman então existem $\alpha, \beta \in \mathbb{Z}_n$ tais que

$$(g_1, \quad g_2 \equiv g_1^{\alpha}, \quad g_3 \equiv g_1^{\beta}, \quad g_4 \equiv g_1^{\alpha\beta} \equiv g_2^{\beta}).$$

Logo, a chave pública e o texto cifrado têm a seguinte forma:

$$\begin{aligned} PK &= (g_1,\, g_2,\, z \equiv g_1^x g_2^y) \\ C &= (g_3 \equiv g_1^{\beta},\, g_4 \equiv g_2^{\beta},\, (g_1^{\beta})^x (g_2^{\beta})^y \cdot m_i) = (g_1^{\beta},\, g_2^{\beta},\, z^{\beta}.m_i), \end{aligned}$$

que é exatamente a distribuição esperada de uma execução "legítima" do protocolo. □

Observe que este lema implica que

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}].$$

Lema 4. *Se S_A recebe uma tupla aleatória, A não tem qualquer informação sobre i, mesmo que A seja ilimitado computacionalmente.*

Um corolário imediato deste lema é que

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)}$$

para todo polinômio $p(.)$ e n suficientemente grande. Seguimos então com a prova deste lema.

Prova do lema 4. Suponha que S_A recebeu uma tupla aleatória como entrada. Existem $\alpha, \beta, \omega \in \mathbb{Z}_n$ tais que

$$(g_1,\, g_2 = g_1^{\alpha},\, g_3 = g_1^{\beta},\, g_4 = g_1^{\omega}).$$

Suponha que $(\omega \neq \alpha\beta)$ e $(\alpha \neq 0)$[10], o que é verdadeiro com imensa probabilidade. Isso implica que existem r, r' tais que $g_3 = g_1^r$ e $g_4 = g_2^{r'}$. Analisemos agora o que A sabe sobre x e y. A partir da chave pública $PK = (g_1, g_2, h)$ sabe-se que $z = g_1^x g_2^y$ e que, portanto, x e y satisfazem

$$\log_{g_1} z = x + (\log_{g_1} g_2) \cdot y = x + \alpha y. \tag{4}$$

Para cada $x \in \mathbb{Z}_n$ existe um $y \in \mathbb{Z}_n$ único que satisfaz esta equação, logo existem exatamente n pares (x, y) válidos, igualmente prováveis.

Agora considere o termo $g_3^x g_4^y$. Analisemos a probabilidade de $g_3^x g_4^y = \mu$ para um $\mu \in \mathbb{G}$ arbitrário. Para que isto aconteça, temos que $\log_{g_1} \mu = \log_{g_1}(g_3^x g_4^y)$, ou seja:

$$\begin{aligned} \log_{g_1} \mu &= x.\log_{g_1} g_3 + y.\log_{g_1} g_4 \\ &= rx + r'\alpha y. \end{aligned} \tag{5}$$

[10]Observe que se $\alpha = 0$ então $g_2 \equiv 1$ e a tupla (g_1, g_2, g_3, g_4) é de Diffie-Hellman se e somente se $g_4 \equiv g_3$.

Sejam $y_1 = \log_{g_1} h$ e $y_2 = \log_{g_1} \mu$. Então as equações (4) e (5) formam um sistema de equações lineares em x e y dado por $B\vec{x} = \vec{z}$, onde

$$B = \begin{pmatrix} 1 & \alpha \\ r & r'\alpha \end{pmatrix}, \quad \vec{x} = [xy]^T, \quad \vec{z} = [z_1 z_2]^T .$$

Lembrando que $r \neq r'$ e $\alpha \neq 0$, o sistema acima sempre tem uma solução única em x, y. Mas como μ é um elemento arbitrário do grupo, isso implica que todo $\mu' \in \mathbb{G}$ é possível e igualmente provável. Em outras palavras, dados (g_1, g_2, g_3, g_4) e $z = g_1^x g_2^y$, para $x, y \in \mathbb{Z}_n$ escolhidos uniformemente, se $\log_{g_1} g_3 \neq \log_{g_2} g_4$, mesmo um algoritmo ilimitado computacionalmente não consegue prever o valor de $\mu = g_3^x g_4^y$ com probabilidade maior que $1/n$, já que todas as opções são igualmente prováveis.

Como, do ponto de vista de A, $g_3^x g_4^y$ é uniformemente distribuído em $\mathbb{G}$, ele não consegue obter nenhum informação sobre a mensagem cifrada e, conseqüentemente, sobre i. □

Combinando os dois lemas acima, sabemos que

$$X = \Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}], \text{ e}$$
$$Y = \Pr[S_A \text{ "ACEITA"} \mid \text{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)} .$$

Por conseguinte, a vantagem de S_A, $\varepsilon(k)$ é

$$\varepsilon(k) = X - Y = \lambda(k) ,$$

que é desprezível (em relação a k), provando o teorema. ■

Reiterando, o `ElGamal-A` tem as mesmas propriedades de segurança do `ElGamal` padrão, mas o utilizamos para apresentar a técnica de demonstração do lema 4, que será importante nos teoremas a seguir. Em seguida apresentamos mais um passo em direção ao `Cramer-Shoup`, uma versão simplificada do esquema que é segura somente contra ataques de texto escolhido não-adaptativos.

Criptossistema 10. *Cramer-Shoup-Lite*

Geração de Chaves. Seja $\mathbb{G}$ um grupo de ordem n onde o `DHD` é difícil e sejam g_1, g_2 geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x, y, a, b \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule $z = g_1^x g_2^y \pmod n$ e calcule $c = g_1^a g_2^b \pmod n$. A chave pública é (g_1, g_2, z, c) e a chave privada é (x, y, a, b).

Ciframento. Seja $M \in \mathbb{G}$ a mensagem a ser cifrada e seja (g_1, g_2, z, c) a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule o texto cifrado $(g_1^r, g_2^r, c^r, z^r M)$.

Deciframento. Seja $(\gamma_1, \gamma_2, \gamma_3, C)$ o texto cifrado e seja (x, y, a, b) a chave privada do usuário. Se $(\gamma_3 = \gamma_1^a \gamma_2^b)$, $M = C/\gamma_1^x \gamma_2^y$; caso contrário, $M = \perp$.

Teorema 8. *O `Cramer-Shoup-Lite` é seguro contra ataques de texto escolhido não-adaptativos se o `DHD` for difícil em* $\mathbb{G}$.

Demonstração. A prova de segurança deste esquema é muito semelhante à prova de segurança do `ElGamal-A`: supomos a existência de um algoritmo $A = (A_1, A_2)$ capaz de quebrar o `Cramer-Shoup-Lite` e construímos um algoritmo S_A que utiliza A para resolver o `DHD`. O fator complicador aqui é que estamos lidando com um adversário ativo, então temos que garantir que as respostas às consultas de decifrамento não revelem nada sobre a simulação. S_A prossegue da seguinte maneira:

1. escolhe $x, y, a, b \xleftarrow{r} \{0, \ldots, n-1\}$;
2. calcula $z = g_1^x g_2^y$ e $c = g_1^a g_2^b$;
3. faz a chave pública $PK = (g_1, g_2, z, c)$;
4. faz a chave privada $SK = (x, y, t, u)$;
5. executa $(m_0, m_1) \leftarrow A_1^{D_{SK}(.)}(PK)$, recebendo as duas mensagens-desafio;
6. escolhe $i \xleftarrow{r} \{0, 1\}$;
7. calcula $C = (g_3,\ g_4,\ g_3^a g_4^b,\ g_3^x g_4^y \cdot m_i)$;
8. executa $i' \leftarrow A_2(PK, C)$
9. se $i = i'$, S_A retorna `ACEITA`;
10. caso contrário, S_A retorna `FALHA`.

Denotamos por $A_1^{D_{SK}(.)}(.)$ a execução do algoritmo $A_1(.)$ com acesso ao oráculo $D_{SK}(.)$ que decifra mensagens utilizando a chave privada SK de acordo com a especificação do criptossistema. A estrutura desta prova é bastante semelhante à da demonstração anterior, para o `ElGamal-A`. As principais mudanças ocorrem na demonstração do segundo lema abaixo.

Lema 5. *Se S_A recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real.*

A prova deste lema é completamente análoga à presente na demonstração de segurança do `ElGamal-A`. Obtemos, inclusive, o mesmo corolário:

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}].$$

Lema 6. *Se S_A recebe uma tupla aleatória, A não tem qualquer informação sobre b, mesmo que A seja ilimitado computacionalmente, desde que A possa fazer apenas uma quantidade polinomial de consultas ao oráculo de decifрamento.*

Este lema, por sua vez, possui uma demonstração um pouco mais do que o seu análogo na demonstração de segurança anterior pois, aqui, temos que garantir que as consultas realizadas por A não trazem nenhuma informação adicional. Perceba que se este lema for verdadeiro o teorema 8 está provado pelo mesmo argumento usado para finalizar a demonstração do teorema 7.

Prova do lema 6. Seja (g_1, g_2, g_3, g_4) a tupla aleatória recebida por S_A. Podemos então escrevê-la como

$$(g_1,\ g_2 \equiv g_1^{\alpha},\ g_3 \equiv g_1^{\beta},\ g_4 \equiv g_1^{\omega}),$$

onde, com imensa probabilidade, $\alpha \neq 0$ e $\omega \neq \alpha\beta$ (assumimos esta situação no resto da prova). A partir da chave pública PK, A sabe que $z = g_1^x g_2^y$ e isso restringe (x,y) de acordo com

$$\log_{g_1} z = x + (\log_{g_1} g_2).y = x + \alpha y, \tag{6}$$

novamente, como na demonstração anterior. Consideremos agora que informações A pode obter a partir de suas consultas de deciframento. Dividimos as consultas $(\gamma_1, \gamma_2, \gamma_3, C)$ realizadas por A em duas categorias: se $\log_{g_1} \gamma_1 \equiv \log_{g_2} \gamma_2$, então a consulta é considerada *legal*; caso contrário, ela é considerada *ilegal*. Prosseguimos então para mostrar dois fatos:

1. A só ganharia informação com a resposta para uma consulta *ilegal*;
2. consultas *ilegais* são rejeitadas com altíssima probabilidade.

Estes dois fatos provam que A não obtém qualquer informação adicional através de suas consultas de deciframento.

Lema 7. *A obtém informação extra sobre (x,y) só se submeter uma consulta de deciframento $(\gamma_1, \gamma_2, \gamma_3, C)$ tal que:*

1. *o oráculo de deciframento não retorne $\perp$, e*
2. $\log_{g_1} \gamma_1 \neq \log_{g_2} \gamma_2$ *(a consulta for* ilegal*)*

Primeiro, suponha que o oráculo de deciframento retornou $\perp$ para um consulta. Isso significa que γ_3 não é da forma correta. Contudo, esta checagem não pode revelar qualquer informação extra sobre (x,y), porque só envolve a e b.

Suponha agora que A submete uma consulta tal que $\log_{g_1} \gamma_1 = \log_{g_2} \gamma_2 = \delta$, para algum δ arbitrário. Neste caso, de acordo com a resposta M do simulador, A sabe que $M = C/\gamma_1^x \gamma_2^y$, o que gera a seguinte restrição nos valores de x e y:

$$\begin{aligned} \log_{g_1} M &= \log_{g_1} C - (\log_{g_1} \gamma_1)x - (\alpha \log_{g_2} \gamma_2)y \\ &= \log_{g_1} C - \delta(x - \alpha y), \end{aligned}$$

que é linearmente dependente da equação (6). Logo, esta equação não introduz qualquer restrição aos valores de x e y, o que nos permite concluir que consultas rejeitadas pelo oráculo e consultas legais não trazem qualquer informação adicional (sobre x e y) para o adversário A. Falta-nos apenas provar que a probabilidade de uma consulta ilegal não ser rejeitada é desprezível, e a prova do teorema estará completa.

Lema 8. *A probabilidade de A submeter uma consulta de deciframento $(\gamma_1, \gamma_2, \gamma_3, C)$ tal que $\log_{g_1} \gamma_1 \neq \log_{g_2} \gamma_2$ e o oráculo de deciframento não a rejeitar é desprezível.*

Sejam $r_1 = \log_{g_1} \gamma_1$ e $r_2 = \log_{g_2} \gamma_2$. Para que o oráculo de deciframento não rejeite a consulta, A deve "prever" o valor $\gamma_3 = \gamma_1^a \gamma_2^b$. Mostramos a seguir que isso não pode ser feito com probabilidade não-desprezível. Considere o que A sabe sobre (a,b). A partir da chave pública, sabe-se que $c = g_1^a g_2^b$ e, conseqüentemente,

$$\log_{g_1} c = a + \alpha b. \tag{7}$$

Seja γ_3' um elemento arbitrário de $\mathbb{G}$. Temos que $\gamma_3' = \gamma_1^a \gamma_2^b$ se e somente se

$$\begin{aligned} \log_{g_1} \gamma_3' &= a \log_{g_1} \gamma_1 + b \log_{g_1} \gamma_2 \\ &= r_1 a + \alpha r_2 b. \end{aligned} \tag{8}$$

As equações (7) e (8) são linearmente independentes e têm solução única em termos de a e b. Como γ_3' é arbitrário, existe solução para qualquer γ_3'. Portanto A tem probabilidade $1/n$ de acertar o γ_3 correto.

Esta análise é válida para a *primeira* consulta realizada por A. Perceba que cada consulta rejeitada adicional revela *alguma* informação adicional sobre (a,b), em particular que $\gamma_3' \neq \gamma_1^a \gamma_2^b$: na melhor das hipóteses isso elimina uma possibilidade de (a,b). Logo, após q_d' consultas rejeitas, ainda existem $n - q_d'$ soluções possíveis para (a,b). Se durante a execução do ataque forem realizadas um total de q_d consultas ao oráculo de deciframento, temos então que a probabilidade de alguma delas não ser rejeitada é $q_d/(n-q_d)$. Como q_d é polinomial em k, e n é exponencial em k, temos que essa probabilidade é desprezível. □

Juntando os lemas 5 e 6 concluímos que A não consegue mais informação sobre (x,y) do que a implícita pela equação (6). Neste caso, um argumento análogo ao da prova de segurança do teorema 7 mostra que $g_3^x g_4^y$ é uniformemente distribuído (do ponto de vista de A) e que A não obtém qualquer informação sobre i. Isto completa a prova do lema 7 e do teorema. ■

Estamos prontos para analisar agora a versão completa do `Cramer-Shoup`, conforme proposto em [Cramer e Shoup, 1998]. A grande diferença entre o cenário da prova anterior e o da seguinte é que A agora é adaptativo, ou seja, ele pode fazer consultas ao oráculo de deciframento mesmo depois de receber o texto cifrado-desafio. Temos que garantir então que também nestas consultas ele não possa conseguir qualquer informação adicional, da mesma maneira que com as consultas antes do desafio ser feito. Para atingir isto, adicionam-se mais duas variáveis desconhecidas (para A) de maneira que o número de incógnitas permaneça maior do que o número de equações conhecidas. Observe que o `Cramer-Shoup` utiliza uma função de *hash* resistente a colisões, conforme a definição 6, *mas não supõe o modelo do oráculo aleatório*.

Criptossistema 11. *Cramer-Shoup*

Geração de Chaves. Seja $\mathbb{G}$ um grupo de ordem n onde o `DHD` é difícil, seja $H : \{0,1\}^* \to \mathbb{G}$ uma função de *hash* resistente a colisões e sejam g_1, g_2 geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x,y,a,b,a',b' \xleftarrow{r} \{1,\ldots,n-1\}$. Calcule $z = g_1^x g_2^y \pmod n$, $c = g_1^a g_2^b \pmod n$ e calcule $d = g_1^{a'} g_2^{b'} \pmod n$. A chave pública é (g_1,g_2,z,c,d,H) e a chave privada é (x,y,a,b,a',b').

Ciframento. Seja $M \in \{1,\ldots,n-1\}$ a mensagem a ser cifrada e seja (g_1,g_2,z,c,d,H) a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1,\ldots,n-1\}$. Seja $h = H(g_1^r, g_2^r, z^r M)$. Calcule o texto cifrado $(g_1^r, g_2^r, (cd)^{hr}, z^r M)$.

Deciframento. Seja $(\gamma_1,\gamma_2,\gamma_3,C)$ o texto cifrado, seja (x,y,a,b,a',b') a chave privada do usuário e seja $h = H(\gamma_1,\gamma_2,C)$. Se $(\gamma_3 = \gamma_1^{a+ha'} \gamma_2^{b+hb'})$, $M = C/\gamma_1^x \gamma_2^y$; caso contrário, $M = \perp$.

Teorema 9. *O* `Cramer-Shoup` *é polinomialmente seguro contra adversários adaptativos se o* `DHD` *é difícil em* $\mathbb{G}$.

Demonstração. Procedemos de forma bastante semelhante à prova anterior. De fato, S_A é definido de maneira idêntica, exceto pelo fato de que A_2 agora também tem acesso ao oráculo de deciframento:

1. escolhe uma função de *hash* resistente a colisões $H(.)$;
2. escolhe $x, y, a, b, a', b' \stackrel{r}{\leftarrow} \{0, \ldots, n-1\}$;
3. calcula $z = g_1^x g_2^y$, $c = g_1^a g_2^b$, e $d = g_1^{a'} g_2^{b'}$;
4. faz a chave pública $PK = (g_1, g_2, z, c, d, H)$;
5. faz a chave privada $SK = (x, y, a, b, a', b')$;
6. executa $(m_0, m_1) \leftarrow A_1^{D_{SK}(.)}(PK)$, recebendo as duas mensagens-desafio;
7. escolhe $i \stackrel{r}{\leftarrow} \{0,1\}$;
8. calcula $h = H(g_3,\ g_4,\ g_3^x g_4^y \cdot m_i)$
9. calcula $C = (g_3,\ g_4,\ g_3^{a+ha'} g_4^{b+hb'},\ g_3^x g_4^y \cdot m_i)$;
10. executa $i' \leftarrow A_2^{D_{SK}(.)}(PK, C)$
11. se $i = i'$, S_A retorna `ACEITA`;
12. caso contrário, S_A retorna `FALHA`.

Lema 9. *Se S_A recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real. Conseqüentemente,*

$$\Pr[S_A \text{ "}\texttt{ACEITA}\text{"} \mid \textit{é tupla DH}] = \Pr[i' = i \mid \textit{A quebra o criptossistema}].$$

A prova deste lema é exatamente como na demonstração anterior. O lema seguinte, no entanto, tem uma demonstração mais complicada dado que A agora é adaptativo.

Lema 10. *Se S_A recebe uma tupla aleatória, A não tem qualquer informação sobre i, mesmo que A seja ilimitado computacionalmente, desde que A possa fazer apenas uma quantidade polinomial de consultas ao oráculo de deciframento. Logo,*

$$\Pr[S_A \text{ "}\texttt{ACEITA}\text{"} \mid \textit{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)}.$$

Prova do lema. A estrutura geral desta prova é semelhante à da prova do lema 6 e assumimos o conhecimento desta última na apresentação da prova atual. Novamente, nosso objetivo é demonstrar que

1. A não consegue fazer qualquer consulta "ilegal" que não é rejeitada pelo oráculo, e
2. que A não consegue obter qualquer informação adicional a não ser através de consultas "ilegais".

Como o ponto 2 é exatamente igual ao `Cramer-Shoup-Lite`, focaremos aqui o ponto 1. Vejamos o que A consegue de informação sobre (a,b,a',b'). A partir da chave pública PK, sabe-se que:

$$\log_{g_1} c = a + b.\alpha; \quad (9)$$

$$\log_{g_1} d = a' + b'.\alpha, \quad (10)$$

onde, como anteriormente, $\alpha = \log_{g_1} g_2$. Sejam $g_3 = g_1^r$ e $g_4 = g_2^{r'}$. Novamente, com imensa probabilidade, $r \neq r'$. Perceba que a análise de possíveis consultas realizadas durante a execução de $A_1(.)$ é análoga à análise presente na demonstração anterior. Vamos nos preocupar, portanto, com as consultas realizadas por $A_2(.)$. Quando A recebe o texto cifrado-desafio, $(\gamma_1^* = g_3,\ \gamma_2^* = g_4,\ \gamma_3^* = g_3^{a+ha'} g_4^{b+hb'},\ C^* = g_3^x g_4^y \cdot m_i)$, sabe que:

$$\log_{g_1} \gamma_3^* = (a+ha')r + (b+hb')\alpha r'. \quad (11)$$

Por um argumento análogo ao presente na demonstração anterior, consultas "legais" (i.e., com $\log_{g_1} \gamma_1 = \log_{g_2} \gamma_2$) não trazem informação adicional para A. Queremos mostrar então que qualquer consulta $(\gamma_1,\gamma_2,\gamma_3,C)$ ilegal será rejeitada com imensa probabilidade pelo oráculo. Analisemos três casos para a consulta $(\gamma_1,\gamma_2,\gamma_3,C)$, lembrando que A não pode submeter a consulta $(\gamma_1,\gamma_2,\gamma_3,C) = (\gamma_1^*,\gamma_2^*,\gamma_3^*,C^*)$:

Caso 1. Se $(\gamma_1,\gamma_2,C) = (\gamma_1^*,\gamma_2^*,C^*)$, mas $\gamma_3 \neq \gamma_3^*$, a consulta certamente é rejeitada (porque o valor do *hash* não será correto).

Caso 2. Se $(\gamma_1,\gamma_2,C) \neq (\gamma_1^*,\gamma_2^*,C^*)$, mas $H(\gamma_1,\gamma_2,\gamma_3) = H(\gamma_1^*,\gamma_2^*,\gamma_3^*)$ então A encontrou uma colisão na função de *hash* (e como a função é resistente a colisões, isso só pode acontecer com probabilidade desprezível).

Caso 3. Se $H(\gamma_1,\gamma_2,C) \neq H(\gamma_1^*,\gamma_2^*,C^*)$ então uma análise mais cuidadosa, feita a seguir, é necessária.

Sejam $h^* = H(\gamma_1^*,\gamma_2^*,C^*)$, e $h = H(\gamma_1,\gamma_2,C)$. Sejam $\log_{g_1} \gamma_1 = r_1$ e $\log_{g_2} \gamma_2 = r_2$. Já que estamos tratando de consultas ilegais, sabemos que $r_1 \neq r_2$. Vamos nos focar na primeira consulta ilegal feita por A. Esta não será rejeitada se

$$\log_{g_1} \gamma_3 = (a+ha')r_1 + (b+hb')\alpha r_2.$$

Esta equação é linearmente independente das equações (9), (10) e (11) em relação às incógnitas (a,b,a',b'). De maneira semelhante à prova anterior, existem n possíveis, e igualmente prováveis, $\gamma_3 \in \mathbb{G}$, então a primeira consulta ilegal será aceita com probabilidade apenas $1/n$. Continuando então de maneira análoga, chega-se à conclusão de que consultas ilegais não são rejeitadas pelo oráculo com probabilidade desprezível. □

O teorema segue diretamente dos lemas acima. ■

Isto conclui então a prova de segurança do `Cramer-Shoup`. Veja em particular que, apesar de fazer uso de uma função de *hash*, a prova de segurança supõe apenas a resistência a colisões, e não uma função de *hash* ideal (como no modelo

do oráculo aleatório, discutido em §3.4). O `Cramer-Shoup` foi assim o primeiro criptossistema prático e seguro contra adversários adaptativos proposto na literatura. Apesar de ser baseado na dificuldade do problema de Diffie-Hellman de Decisão, uma suposição razoavelmente forte, ele representou um resultado de grande importância para a criptografia moderna.

3.3.3. Assinaturas

Depois de desenvolver com certo de grau de detalhe noções de segurança para ciframento de chave pública, voltaremos nossa atenção para o problema de gerar assinaturas digitais e avaliar a sua segurança. O que significa "ser seguro" para um esquema de assinaturas? A noção intuitiva de segurança é a de "não-falsificação": estudaremos então o quão difícil é falsificar uma assinatura numa mensagem arbitrária. A primeira divisão importante é o quanto de informação um possível adversário tem sobre o usuário (digamos, Alice) cujas assinaturas deseja falsificar:

> **Ataque apenas com a chave.** O adversário conhece apenas os parâmetros do sistema e a chave pública de Alice.
>
> **Ataque de mensagem conhecida.** O adversário conhece também uma lista de mensagens anteriormente assinadas por Alice.
>
> **Ataque de mensagem escolhida.** O adversário, além das informações acima, tem acesso a um oráculo de assinatura, a quem ele pode submeter mensagens arbitrárias e que devolve uma assinatura de Alice válida na mensagem.

A outra grande classificação que se pode fazer é: o que, exatamente, é considerado "sucesso" para o adversário?

> **Quebra total.** O adversário consegue descobrir a chave privada de Alice.
>
> **Falsificação *universal*.** O adversário é capaz de gerar falsificações para mensagens arbitrariamente escolhidas sem ter observado uma assinatura de Alice para esta mensagem.
>
> **Falsificação *existencial*.** O adversário é capaz de gerar um par assinatura-mensagem (σ, M), mas não é capaz de determinar nada *a priori* sobre a mensagem.

Observe que, enquanto concentramos nosso tratamento inicial da segurança de criptossistemas em adversários passivos, nosso tratamento de esquemas de assinaturas será focado em segurança contra adversários adaptativos. Definimos então, formalmente, o que daqui para frente consideraremos um esquema de assinaturas (existencialmente) seguro (contra ataques de mensagem escolhida).

Definição 5. *Seja A um algoritmo polinomial arbitrário com acesso a um oráculo O. Denotamos por $Q_A^O(x)$ as consultas feitas por A ao oráculo O quando recebe x como entrada. Seja $A^O(x)$ a saída de A quando recebe x como entrada. Um esquema de assinatura (G,S,V) é seguro se para todo algoritmo polinomial A, todo polinômio $p(.)$, e todo k suficientemente grande*

$$\Pr\left[(V_v(\sigma,M)=1)\wedge(M\notin Q_M^{S_s}(v)) \mid (\sigma,M)\leftarrow A^{S_s}(v)\right] < \frac{1}{p(k)},$$

onde $(s,v) \leftarrow G(1^k)$.

Um maneira mais intuitiva de formular a definição de segurança acima segue:

1. Seja (G,S,V) um esquema de assinatura como na definição 3.
2. Dado um par de chaves (s,v) gerado de acordo com o parâmetro de segurança em questão (k).
3. Nenhum algoritmo A que receba a chave pública v como parâmetro e tenha acesso a um oráculo que calcule assinaturas corretamente (S_s) pode ser capaz de gerar, com probabilidade não-desprezível, um par assinatura-mensagem (σ,M) tal que:
   - o par seja aceito pelo algoritmo de verificação ($V_v(\sigma,M) = 1$);
   - M nunca tenha sido consultada ao oráculo ($M \notin Q_M^{S_s}(v)$).

Perceba que, quando se trata de esquemas de assinaturas, a relevância de adversários ativos é muito mais clara do que no caso de esquemas de ciframento: naturalmente um possível adversário terá acesso a várias mensagens assinadas por um dado usuário pois estas costumam estar publicamente disponíveis (enquanto textos decifrados por usuários não são, em geral, facilmente disponibilizadas). Portanto realmente não faz muito sentido pensar em segurança contra adversários passivos no contexto de esquemas de assinatura.

3.3.3.1. *"Hash and Sign"*

Não discutiremos nenhum esquema de assinatura concreto até abordarmos o modelo do oráculo aleatório. No entanto, discutiremos um resultado muito interessante que apóia a prática comum de assinar o *hash* de uma mensagem. Em primeiro lugar, vamos definir precisamente o tipo de função de *hash* ao qual nos referimos na discussão a seguir.

Definição 6. *(Funções de hash resistentes a colisões) Seja* $l : \mathbb{N} \to \mathbb{N}$. *Uma coleção de funções* $\{h_s : \{0,1\}^* \to \{0,1\}^{l(|s|)}\}_{s\in\{0,1\}^*}\}$ *é chamada de* `hash resistente a colisões` *se:*

1. ***Cálculo eficiente.*** *Existe um algoritmo polinomial que, dados s e x, calcula* $h_s(x)$.
2. ***Difícil de gerar colisões.*** *Diz-se que o par* (x,x') *forma uma colisão na função h se* $h(x) = h(x')$ *mas* $x \neq x'$. *Para que h seja resistente a colisões, tem-se que para todo algoritmo polinomial A, todo polinômio* $p(.)$ *e todo n grande o suficiente,*

$$\Pr[A_{h_s}(1^n) \text{ é uma colisão de } h_s] < \frac{1}{p(n)},$$

onde a probabilidade é tomada em relação à escolha de h_s *e em relação às escolhas aleatórias de* $A(.)$.

Definimos então esquemas de assinaturas *de tamanho restrito*. Intuitivamente, um esquema de assinaturas l-restrito só permite a assinatura de documentos de tamanho $l(n)$, onde n é o parâmetro de segurança do sistema.

Definição 7. *(Esquema de assinaturas para documentos de tamanho fixo) Seja* $l : \mathbb{N} \to \mathbb{N}$. *Um esquema de assinatura l-restrito é uma tupla* (G,S,V) *de algoritmos probabilísticos de tempo polinomial tais que, analogamente à definição 3:*

1. *ao receber* 1^n *o algoritmo G retorna um par de strings;*
2. *para todo n e todo par* (s,v) *imagem de* $G(1^n)$*, para todo* $\alpha \in \{0,1\}^{l(n)}$*, os algoritmos S e V satisfazem*

$$\Pr[V_v(\alpha, S_s(\alpha)) = 1] = 1\,.$$

Tal esquema é considerado seguro *se preenche os requisitos (análogos aos) presentes na definição 5 para adversários restritos a fazer consultas de tamanho* $l(n)$ *e geram uma falsificação* $\varsigma = (M,\sigma)$ *onde* $|M| = l(n)$*.*

Faz todo o sentido, na busca de um esquema de assinatura seguro, o projeto inicial de um esquema que opere em blocos de tamanho fixo, de maneira semelhante ao que fizemos para criptossistemas: posteriormente pode-se tentar estender este esquema para assinar mensagens arbitrárias. Trataremos nesta seção de como fazer esta extensão, ou seja, suponha que você conseguiu um esquema de assinatura seguro (contra ataques de mensagem escolhida) mas apenas para mensagens de tamanho $l(n)$ (onde, digamos, $l(n) = n$). Como utilizá-lo para assinar mensagens de tamanho arbitrário?

A prática comum é utilizar um função de *hash*: escolhemos uma função de *hash* $H(.)$ que mapeia mensagens de tamanho arbitrário para strings de $l(n)$ bits e, dada M para ser assinada, assinamos $H(M)$. Este é o conhecido paradigma do `Hash-And-Sign`. Provamos a seguir que, se a função $H(.)$ for resistente a colisões, o paradigma do `Hash-And-Sign` provê um esquema de assinaturas seguro.

Esquema de Assinatura 1. *Assinaturas sem restrição de tamanho* (`Hash-And-Sign`)

Sejam l e (G,S,V) como definição 7 e $H_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ uma função de *hash* resistente a colisões como na definição 6. Construímos então o esquema de assinaturas (G',S',V') a seguir:

Geração de Chaves (G')**.** Recebendo 1^n como entrada, G' faz $(s,v) \leftarrow G(1^n)$. Escolha $H_n : \{0,1\}^* \to \{0,1\}^n$ e devolva o par $((H_n,s),(H_n,v))$.

Assinatura (S')**.** Seja (H_n,s) a chave de assinatura e seja $M\{0,1\}^*$ a mensagem a ser assinada. S' faz $\sigma \leftarrow S_s(H_n(M))$ e devolve σ.

Verificação (V')**.** Seja (H_n,v) a chave de verificação e seja (M,σ) o par mensagem-assinatura. V' devolve o mesmo que $V_v(H_n(M),\sigma)$.

Teorema 10. *Suponha que* (G,S,V) *é um esquema de assinaturas l-restrito seguro. Suponha que* H_n *é uma função de hash livre de colisões. Então* (G',S',V')*, como definido acima, é um esquema de assinatura seguro sem restrição no tamanho das mensagens assinadas.*

Demonstração. Suponha que exista um adversário A capaz de atacar o esquema (G',S',V'). Construímos então um algoritmo S_A que utiliza A e que ou forja assinaturas do esquema l-restrito (G,S,V) ou gera colisões na função de *hash* $H_n(.)$. S_A tem tempo de execução e probabilidade de sucesso relacionados aos de A. Descrevemos S_A a seguir:

1. S_A recebe como entrada uma chave pública $PK = v$ do esquema l-restrito;

2. escolhe uma função de *hash* aleatória H_n;
3. executa $A^{S_{s,H_n}}(H_n, v)$; (o oráculo S_{s,H_n} é descrito a seguir)
4. A devolve um par mensagem assinatura (M^*, σ^*) no esquema "complexo" (G', S', V');
5. S_A devolve o par $(H_n(M^*), \sigma^*)$ como uma falsificação do esquema l-restrito.

As distribuições das chaves simuladas por S_A são exatamente como num ataque real. A simulação do oráculo S_{s,H_n} é feita da seguinte forma (lembre que S_A, que é um adversário do esquema l-restrito, tem acesso a um oráculo S_s que gera assinaturas para este esquema):

> $S_{s,H_n}(M)$. S_A simplesmente utiliza o seu oráculo para simular este, devolvendo $S_s(H(M))$.

Suponha agora que com probabilidade $\varepsilon'(n)$ A consegue gerar uma falsificação do esquema complexo. Seja M_i a i-ésima consulta realizada por A ao oráculo S_{s,H_n} e σ_i a respectiva resposta. Lembre que (M^*, σ^*) é saída de A. Analisemos dois casos:

> **Caso 1.** $H_n(M^*) \neq H_n(M_i)$ para todo i. Neste caso o par $(H(M^*), \sigma^*)$ é uma falsificação válida para o esquema l-restrito porque este valor de *hash* $(H(M^*))$ não havia sido consultado ainda ao oráculo S_s.
>
> **Caso 2.** $H_n(M^*) = H_n(M_i)$ para algum i. Neste caso, conseguimos uma colisão da função de *hash*.

Seja S_A^* o evento de o caso 1 acontecer e seja C_A^* o evento de o caso 2 acontecer. Temos então que

$$\Pr[A^*] = \Pr[S_A^*] + \Pr[C_A^*].$$

Como a função de *hash* é resistente a colisões e o esquema l-restrito é seguro, temos que

$$\Pr[S_A^*] + \Pr[C_A^*] < \frac{1}{p(n)},$$

para n suficientemente grande. Isso mostra que $\Pr[A^*]$ é desprezível em n, contradizendo a hipótese de que A era capaz de atacar o esquema (com probabilidade não-desprezível). □

Perceba que, na realidade, não fornecemos um esquema de assinatura com a construção acima. Provamos apenas que, se existir um esquema de assinatura seguro para mensagens de tamanho $l(n)$ e uma função de *hash* resistente a colisões com imagem em $\{0,1\}^{l(n)}$, podemos estender este esquema para assinar mensagens de tamanho arbitrário de maneira extremamente eficiente, pagando apenas o preço do *hash*. Este fato justifica, de alguma maneira, a técnica largamente utilizada de assinar o *hash* de mensagens: o problema é que, na prática, não são utilizados esquemas demonstravelmente seguros, invalidando assim a construção geral do `Hash-And-Sign`.

3.3.4. Próximos Passos

Os anos 80 trouxeram importantíssimas contribuições teóricas para a criptografia moderna. A formalização de noções fortes de segurança e o projeto de esquemas que as atingiam proporcionaram à criptografia uma base sólida sobre a qual continuar o seu desenvolvimento. O problema é que quando se trata da ameaça de adversários ativos estes resultados geralmente se restringiam a resultados de possibilidade: os esquemas propostos estavam longe de serem práticos. As técnicas que realmente se utilizavam na prática, por outro lado, constituíam-se principalmente de heurísticas, esquemas que conquistaram a confiança geral da comunidade pela incapacidade dos pesquisadores em quebrá-los.

Claramente o grande desafio para os pesquisadores no início dos anos 90 seria a aproximação entre os esquemas efetivamente utilizados na prática e as noções fortes de segurança desenvolvidas nos anos anteriores, especificamente quando se trata de adversários ativos (já que, como vimos aqui, existiam esquemas eficientes e demonstravelmente seguros contra adversários passivos). Estudaremos a seguir um paradigma de projeto de algoritmos criptográficos que tem esta aproximação entre a prática e a teoria como principal objetivo e foi largamente adotado ao longo da década de 90: o paradigma do oráculo aleatório.

3.4. O Paradigma do Oráculo Aleatório

O paradigma do oráculo aleatório surgiu basicamente como uma tentativa de formalizar um conjunto de práticas heurísticas utilizadas pela comunidade criptográfica para lidar com adversários adaptativos. O uso de funções de *hash* neste cenário tornou-se cada vez mais corriqueiro, mas apenas a suposição de que estas seriam resistentes a coliões não parecia ser suficiente para provar muitas das propriedades que se esperavam delas e que, ao menos aparentemente, estas efetivamente possuíam na prática. A intuição geral era a de que uma boa função de *hash* se comportaria efetivamente como uma função pseudo-aleatória, não permitindo o cálculo de qualquer correlação entre suas entradas e saídas. Seguindo essa linha heurística de pensamento, propõe-se que funções de *hash* sejam encaradas como caixas-pretas que implementam funções efetivamente aleatórias, apesar de estas certamente não o serem.

No paradigma do oráculo aleatório o projeto de protocolos segue portanto três passos:

1. suponha que todos os potenciais participantes do protocolo têm acesso a um oráculo público e aleatório;
2. prove que o protocolo é correto (e seguro) neste *Modelo do Oráculo Aleatório*;
3. instancie o protocolo na prática simulando o oráculo aleatório com um objeto tal como uma função de *hash*.

Certamente não se pode afirmar que a demonstração obtida no passo (2) prova que o protocolo é efetivamente seguro na prática. Espera-se, no entanto, que quaisquer vulnerabilidades que possam existir sejam problemas na instanciação (i.e., escolha de uma função de *hash* inapropriada) e não uma falha "estrutural" do protocolo. Divide-se então o problema de projetar um protocolo complexo em (1) projetar uma versão "ideal" do protocolo e (2) projetar uma boa implementação para o oráculo aleatório. Este paradigma

traduz, de alguma maneira, uma opinião geral da comunidade de que uma boa função de *hash* não deve possibilitar a detecção de correlações entre suas saídas, agindo, até certo ponto, como uma função pseudo-aleatória. Nas próximas subseções discutiremos alguns dos protocolos que foram desenvolvidos neste modelo "ideal" e, finalmente, discutiremos as principais objeções e controvérsias que cercam o paradigma do oráculo aleatório.

3.4.1. Ciframento no Modelo do Oráculo Aleatório

Na seção anterior, vimos alguns exemplos de criptossistemas polinomialmente seguros, mas sempre contra adversários passivos. Agora mostraremos que, no modelo do oráculo aleatório, existe um criptossistema extremamente eficiente e que é polinomialmente seguro contra ataques de texto cifrado escolhido. Ele pode ser encarado como uma extensão do criptossistema 7 onde, ao invés de sucessivas aplicações da permutação p, utilizamos uma função de *hash* (G) como gerador pseudo-aleatório (uma vez que estamos no modelo do oráculo aleatório) e utilizamos uma outra função de *hash* (H) para "validar" textos cifrados legítimos: esta validação protege o esquema de adversários adaptativos. Este criptossistema, assim como sua prova de segurança, foi originalmente apresentado em [Bellare e Rogaway, 1993].

Criptossistema 12. *Bellare-Rogaway contra ataques ativos* (BR)

Geração de chaves. Seja $p : \mathbb{G} \to \mathbb{G}$ uma permutação com segredo s^* e sejam $G : \{0,1\}^* \to \{0,1\}^\infty$ e $H : \{0,1\}^* \to \{0,1\}^k$ duas funções aleatórias. A chave pública PK será (p, G, H) e a chave privada SK será s^*.

Ciframento. Seja $M \in \{0,1\}^*$ a mensagem a ser cifrada e seja $PK = (p, G, H)$ a chave pública do destinatário. Escolha $r \xleftarrow{r} \mathbb{G}$. O texto cifrado será então $C = p(r)||x \oplus G(r)||H(rx)$.

Deciframento. Seja $C = \gamma_1||\gamma_2||\gamma_3$ o texto a ser decifrado. Utilizando s^*, calcula-se $r' = p^{-1}(\gamma_1)$. Seja então $X = \gamma_2 \oplus G(r')$. Se $H(r'||X) = \gamma_3$, então $M = X$; caso contrário, $M = \perp$. Devolva M.

Teorema 11. *O criptossistema* BR *é seguro contra ataques de texto cifrado escolhido.*

Demonstração. Seja $A = (A_1, A_2)$ um adversário capaz de quebrar o BR. Construímos então um algoritmo S_A que utiliza A para inverter a permutação p. S_A funcionará da seguinte maneira:

1. recebe como entrada $(\mathbb{G}, p, Y)$ e deve encontrar $x \in \mathbb{G}$ tal que $Y = p(x)$;
2. executa $(m_0, m_1) \leftarrow A_1(PK)$;
3. escolhe $i \xleftarrow{r} \{0,1\}$;
4. calcula $C = Y||\gamma_2||\gamma_3$ para $\gamma_2 \xleftarrow{r} \{0,1\}^{|m_0|}$ e $\gamma_3 \xleftarrow{r} \{0,1\}^k$;
5. executa $i' \leftarrow A_2(PK, C)$;

Ao longo de todo o ataque, S_A precisa simular os seguintes oráculos:

G(r_i). Se $p(r_i) = Y$, S_A pára e devolve r_i. Caso contrário, devolve uma string aleatória do tamanho apropriado.

H($r_i||M_i$). Se $p(r_i) = Y$, S_A pára e devolve r_i. Caso contrário, devolve uma string aleatória do tamanho apropriado.

Decifra($C_i = (\gamma_1, \gamma_2, \gamma_3)$). S_A busca, entre as consultas já feita a $G(r)$ e $H(r||x)$, por (r', x') tais que:

- $H(r'||x') = \gamma_3$;
- $p(r') = \gamma_1$;
- $G(r') \oplus x' = \gamma_2$.

Caso um par (r', x') apropriado seja encontrado, o oráculo devolve x'; caso contrário, devolve `inválido`.

Perceba então que S_A tem sucesso quando A faz uma consulta que contenha r_i, tal que $p(r_i) = Y$, a um dos oráculos. Precisamos então mostrar duas coisas:

1. S_A simula corretamente uma execução legítima do criptossistema;
2. a probabilidade de A conseguir adivinhar i mas não fazer a consulta que inverte Y é essencialmente $1/2$.

Para estruturar um pouco melhor a prova, vamos dividi-la em lemas.

Lema 11. *S_A simula corretamente uma execução legítima do criptossistema.*

Prova do Lema 11. Os oráculos $G(.)$ e $H(.)$ são simulados perfeitamente. O oráculo de deciframento também é correto, exceto quando devolve "`inválido`" para uma consulta válida, i.e., onde $(\gamma_1, \gamma_2, \gamma_3)$ são tais que

$$\gamma_3 = H(p^{-1}(\gamma_1)||\gamma_2 \oplus G(p^{-1}(\gamma_1))) . \tag{12}$$

Seja D_{err} este evento em que o oráculo devolve "`inválido`" erroneamente: exceto quando D_{err} ocorre, a simulação é correta. Temos então que

$$\Pr[\, A^* \mid \overline{D_{\text{err}}} \,] = \Pr[\mathbf{SUCC}_A] = \lambda(k) + \frac{1}{2} ,$$

onde $\Pr[A^*]$ é a probabilidade de sucesso de A no ambiente simulado por S_A, e $\Pr[\mathbf{SUCC}_A]$ é a probabilidade de sucesso de A num ataque real. Perceba, no entanto, que por causa da aleatoriedade de $H(.)$ e $G(.)$, a probabilidade de D_{err} ocorrer é igual à probabilidade de A adivinhar corretamente a saída destas funções aleatórias sem as consultar (de maneira a satisfazer a eq. (12)). Ou seja,

$$\Pr[D_{\text{err}}] \leqslant q_d 2^{-k} ,$$

onde q_d é o total de consultas feitas ao oráculo de deciframento. Como q_d é polinomial em k, essa probabilidade é desprezível. Logo, com imensa probabilidade, a simulação executada por S_A é correta. □

Lema 12. *A probabilidade de A conseguir adivinhar i mas não fazer a consulta que inverte Y é essencialmente* $1/2$.

Prova do Lema 12. Devido à aleatoriedade de $G(.)$ e $H(.)$, $\gamma_2 = G(r) \oplus m_i$ é essencialmente uma cifra perfeita: a única maneira através da qual A pode obter *qualquer* informação sobre qual m_i foi cifrada é consultando r tal que $p(r) = \gamma_1$. Seja S_A^* o evento

em que S_A inverte com sucesso a permutação p (porque A realizou a consulta "esperada" a $G(.)$ ou $H(.)$). Temos então que

$$\Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge \overline{S_A^*} \,] = \frac{1}{2}.$$

□

Segue então que

$$\begin{aligned}
\Pr[A^*] &= \Pr[\, A^* \mid D_{\text{err}} \,]\Pr[D_{\text{err}}] + \Pr[\, A^* \mid \overline{D_{\text{err}}}]\Pr[\overline{D_{\text{err}}}] \\
&= \Pr[\, A^* \mid D_{\text{err}} \,]\Pr[D_{\text{err}}] + \Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge \overline{S_A^*} \,].\Pr[\overline{D_{\text{err}}} \wedge \overline{S_A^*}] + \\
&\quad \Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge S_A^* \,]\Pr[\overline{D_{\text{err}}} \wedge S_A^*] \\
&\leqslant \Pr[D_{\text{err}}] + \frac{1}{2} + \Pr[S_A^*] \\
&\leqslant q_s 2^{-k} + \frac{1}{2} + \Pr[S_A^*].
\end{aligned}$$

Lembrando que $\Pr[A^*] = 1/2 + \lambda(k)$ para algum $\lambda(k)$ não-desprezível e que a simulação falha com probabilidade no máximo $q_s 2^{-k}$, temos que

$$\Pr[S_A^*] \geqslant \frac{1}{2} + \lambda(k) - \frac{1}{2} - 2.q_s 2^{-k} = \lambda(k) - q_s 2^{-k+1}.$$

que é não-desprezível (em k), provando assim o teorema. ■

3.4.2. Assinando no Modelo do Oráculo Aleatório

Suponha a existência de uma permutação com segredo. Provamos que, no Modelo do Oráculo Aleatório, o simples esquema de Hash-And-Sign usando qualquer permutação com segredo é seguro. Para facilitar nossa apresentação, utilizaremos a função RSA (definição 2) como exemplo. Provamos então que, no Modelo do Oráculo Aleatório, o esquema de Hash-And-Sign com RSA é seguro.

Esquema de Assinatura 2. *Hash-And-Sign com RSA* (`HaS-RSA`)

Geração de Chaves. Selecionam-se aleatoriamente dois primos de n bits, p e q; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par (e,d) tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par (N,e) e a chave privada será o par (N,d).

Assinatura. Seja (d,N) a chave privada do usuário, seja M a mensagem a ser assinada e seja $H(.)$ a função de *hash* apropriada. A assinatura σ é calculada como

$$\sigma = H(M)^d \pmod{N}.$$

Verificação. Seja (e,N) a chave pública do usuário, seja (σ,M) o par assinatura-mensagem a ser verificado e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura σ se e somente se

$$\sigma^e = H(M) \pmod{N}.$$

A estrutura da prova de segurança é bastante simples: vamos supor a existência de um algoritmo A que é capaz de (com alta probabilidade) quebrar o `HaS-RSA`. Construiremos então um outro algoritmo, S_A, que internamente utiliza A e é capaz de inverter a função RSA sem conhecer o segredo correspondente. Em outras palavras, A representa um algoritmo probabilístico de tempo polinomial que:

1. Recebe como entrada a chave pública (e,N) de um usuário arbitrário;
2. Tem acesso a um oráculo `H(.)` que calcula $H(x)$ para qualquer x;
3. Tem acesso a um oráculo `Assina` que calcula uma assinatura válida para qualquer x (i.e., calcula σ tal que $\sigma^e \equiv H(x) \pmod N$);
4. Devolve um par (σ^*, M^*) tal que $(\sigma^*)^e \equiv H(M^*) \pmod N$.

O algoritmo S_A, por sua vez, funcionará da seguinte maneira:

1. Recebe como entrada uma instância do Problema-RSA, i.e., uma tupla (Y,e,N) tal que $(\exists X)\,(Y \equiv X^e \pmod N)$.
2. A partir de (Y,e,N) gera uma chave pública (e',N');
3. Executa $A(e',N')$ e simula os dois oráculos a que A tem acesso: `H(.)` e `Assina`;
4. Seja (σ^*, M^*) o par devolvido por A. S_A usa esta resposta para calcular X tal que

$$X^e \equiv Y \pmod N.$$

Teorema 12. *Se a Função-RSA induzir uma permutação com segredo no seu domínio, o esquema de assinatura `HaS-RSA` é existencialmente seguro contra ataques de mensagem escolhida no modelo do oráculo aleatório.*

Demonstração. Seja A um algoritmo polinomial capaz de gerar uma falsificação existencial do `HaS-RSA` com um ataque de mensagem escolhida no modelo do oráculo aleatório. Contruímos um algoritmo S_A capaz de inverter a Função-RSA. S_A recebe (Y,e,N) como entrada. Seja $k = |N|$ o parâmetro de segurança e seja a chave pública $PK = (e,N)$. S_A executa $A(e,N)$ como parâmetro. Seja $\lambda(k)$ a probabilidade de sucesso de A. Sabe-se que $\lambda(k)$ é não-desprezível, ou seja, para todo polinômio $p(.)$ e k suficientemente grande,

$$\lambda(k) \geqslant \frac{1}{p(k)}.$$

Seja q_H (resp. q_S) a quantidade de consultas feitas por A ao oráculo `H(.)` (resp. `Assina`) durante a execução. Assumimos, sem perda de generalidade, que não são feitas consultas repetidas e que qualquer consulta a `Assina` é precedida de uma consulta a `H(.)` para a mesma mensagem. S_A escolhe $t \xleftarrow{r} \{1,\ldots,q_H\}$. S_A simula então os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H(M_i).** Se esta é a t-ésima consulta realizada por A, retorne $H(M_t) = Y$. Caso contrário, escolha $r_i \xleftarrow{r} \{0,1\}^k$, calcule $y_i = r_i^e \mod N$, e devolva $H(M_t) = y_i$.
- **Assina(M_j).** Se $M_j = M_t$, S_A pára e retorna `FALHA`. Caso contrário, S_A encontra i tal que $M_i = M_j$ e retorna r_i.[11]

[11] Perceba que como supomos que consultas a `Assina` são sempre precedidas de consultas a `H(.)` para a mesma mensagem, sempre poderemos encontrar $i, M_i = M_j$.

Se A falhar, S_A também falha. Com probabilidade $\lambda(k)$, A devolve um par (σ^*, M^*) válido. Se $M^* \neq M_t$, S_A pára e devolve `FALHA`. Caso contrário, a saída de S_A é $\sigma^* \equiv Y^e \pmod{N}$, invertendo a Função-RSA com sucesso.

Precisamos analisar agora a probabilidade $\varepsilon(k)$ de sucesso de S_A. Seja A^* o evento em que A tem sucesso ($\Pr[A^*] = \lambda(k)$). Qual a probabilidade de M^* não ter sido consultada a `H(.)`? Como a resposta de `H(.)` é completamente aleatória, a chance A adivinhá-la é basicamente $1/2^k$. Logo,

$$\begin{aligned}\Pr[A^*] &= \Pr[(A^*) \wedge (M^* \in \{M_1, \ldots, M_{q_H}\})] + \Pr[(A^*) \wedge (M^* \notin \{M_1, \ldots, M_{q_H}\})] \\ &= \sum_{i=1}^{q_H} \Pr[(A^*) \wedge (M^* = M_i)] + 2^{-k} = \lambda(k)\,.\end{aligned}$$

Como t é escolhido aleatoriamente, tem-se que

$$\begin{aligned}\varepsilon(k) = \Pr[(A^*) \wedge (M^* = M_t)] &\geqslant \sum_{i=1}^{q_H} \Pr[(A^*) \wedge (M^* = M_i)]/q_H \\ &\geqslant (\lambda(k) - 2^{-k})/q_H, \qquad (13)\end{aligned}$$

e, como q_H é polinomial (em k), se $\lambda(k)$ é não-desprezível, $\varepsilon(k)$ também o é. □

Perceba que esta redução em momento algum utiliza características específicas do RSA. Na verdade, uma redução análoga a esta pode ser usada para provar que *qualquer* permutação com segredo gera um esquema de assinatura baseado no `Hash-and-Sign` que é seguro no modelo do oráculo aleatório, o que nos leva ao teorema a seguir.

Teorema 13. *Qualquer permutação com segredo dá origem a um esquema de assinaturas existencialmente seguro contra ataques de mensagem escolhida no modelo do oráculo aleatório.*

A prova deste teorema é análoga à prova do teorema 12 e fica como exercício para o leitor.

3.4.2.1. Eficiência da Redução

O leitor mais atento vai perceber que a redução apresentada na sub-seção anterior, apesar de manter o caráter polinomial do algoritmo, tem uma grande perda de eficiência. Relembrando a equação (13), $\varepsilon(k) \geqslant (\lambda(k) - 2^{-k})/q_H$, onde q_H pode ser um número bastante grande, limitado apenas pelo número de passos de A. Quando tratamos de reduções em teoria da complexidade de algoritmos, geralmente a eficiência da redução não é um tema importante: o crucial é manter o caráter polinomial. Em criptografia, no entanto, a aplicação destas reduções de segurança requer um pouco mais de cuidado.

Para facilitar um pouco a discussão que segue introduziremos uma notação mais apropriada: dizemos que uma primitiva é (t, ε)-segura se todo algoritmo A limitado a t passos tem probabilidade no máximo ε de "quebrar" a primitiva[12]. Nessa notação

[12] Perceba que as noções exatas de "quebrar" e "seguro" (e.g. que tipo de operações podem ser realizadas por A, oráculos aos quais tem acesso) dependem da primitiva em análise.

podemos então dizer que, se a Função-RSA é (t,ε)-segura, a equação (13) nos garante que o HaS−RSA é (t,λ)-seguro para

$$t \approx t' \quad \text{e} \quad \lambda \geqslant \varepsilon \cdot q_H .$$

Provamos que HaS−RSA é seguro se a Função-RSA for de difícil inversão fornecendo um algoritmo S_A capaz de usar qualquer algoritmo A que quebre HaS−RSA para inverter a Função-RSA. Supondo que inverter a Função-RSA é difícil, HaS−RSA é difícil. Mas, no mundo real, o que significa a afirmação "inverter a Função-RSA é difícil"? Significa que, para um conjunto de parâmetros factíveis, os melhores algoritmos conhecidos para inverter a Função-RSA têm chance desprezível de sucesso quando executados por um tempo factível. Usemos então a notação acima e digamos que um adversário que executa 2^{60} passos tem chance 2^{-100} de inverter a Função-RSA para um parâmetro de segurança k específico, ou seja, esta é $(2^{60},2^{-100})$-segura. Digamos ainda que consideramos um esquema seguro se ele é $(2^{60},\varepsilon)$-seguro para $\varepsilon \leqslant 2^{-70}$. Se tentarmos então usar a redução acima para deduzir o nível de segurança do HaS−Sign quando instanciado com este mesmo parâmetro de segurança temos um problema. A primeira questão é: qual o valor de q_H, o limite superior no número de consultas a $H(.)$ executadas por A? O único limite indiscutível seria 2^{60}, o número de passos do adversário, mas façamos o exercício de supor que uma em cada mil operações de A seja uma consulta a $H(.)$. Isso nos dá, por exemplo, $q_H \approx 2^{50}$. A equação (13) nos garante então que esta instanciação do HaS−RSA é $(2^{60},2^{-50})$-segura, o que é insatisfatório pela nossa definição de segurança.

Os valores exatos nesta discussão não são relevantes; o importante é o questionamento do significado de uma redução ineficiente. Certamente a polinomialidade da redução é essencial. Mas apenas isso é suficiente para garantir a segurança do esquema resultante? Existe uma forte polêmica na comunidade criptográfica em relação a o quão relevante é a eficiência de uma redução e a como demonstrações como a apresentada anteriormente devem ser encaradas. Se tentarmos utilizar a redução para deduzir bons parâmetros para instanciar o esquema na prática, teremos que aumentar bastante o parâmetro de segurança (tornando as operações do esquema menos eficientes) para compensar a ineficiência da redução. Tendo isso em mente, surgiram novas propostas de esquemas que mantivessem as propriedades de segurança do HaS−RSA mas tivessem reduções de segurança eficientes. Analisamos a seguir uma variação de uma das mais importantes destas propostas, o PSS−RSA.

3.4.2.2. Assinaturas Aleatorizadas com o RSA

Seguindo o espírito da discussão anterior sobre a eficiência de reduções de segurança, em [Bellare e Rogaway, 1996] os autores propuseram um novo esquema de assinatura baseado no RSA, chamado de PSS−RSA, que tem uma redução de segurança eficiente. Analisaremos aqui uma versão simplificada deste esquema, seguindo [Coron, 2002], que ilustra muito bem as idéias presentes no PSS−RSA: manteremos o nome original do esquema, PFDH−RSA *(Probabilistic Full Domain Hash RSA)*. Esta é uma versão aleatorizada do HaS−RSA onde a mensagem é concatenada a um fator aleatório r antes

de ser assinada. Este fator r tem então que ser enviado junto à assinatura para que ela possa ser posteriormente verificada. Seja $k_0 < k$ um parâmetro do sistema. Nesta versão simplificada assumimos que $k_0 \geqslant \log_2 q_s$, onde q_s é o número de assinaturas que permitiremos a um potencial adversário observar durante um ataque. Os resultados obtidos seriam muito próximos para $k_0 < \log_2 q_s$, mudando apenas a análise final quanto à probabilidade de sucesso, presente no artigo original [Coron, 2002].

Esquema de Assinatura 3. *Probabilistic Full Domain Hash RSA* (`PFDH-RSA`)

Geração de Chaves. Selecionam-se aleatoriamente dois primos de n bits, p e q; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par (e,d) tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par (N,e) e a chave privada será o par (N,d).

Assinatura. Seja (N,d) a chave privada do usuário, seja M a mensagem a ser assinada e seja $H(.)$ a função de *hash* apropriada. Escolha $r \xleftarrow{r} \{0,1\}^{k_0}$. A assinatura σ é calculada como

$$\sigma = H(M||r)^d \pmod{N}.$$

Verificação. Seja (N,e) a chave pública do usuário, seja (σ,M,r) a tupla a ser verificada e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura σ se e somente se:

$$\sigma^e = H(M||r) \pmod{N}.$$

Perceba que a geração de chaves aqui é exatamente como em `HaS-RSA`, e o que muda na geração/verificação de assinaturas é o uso do fator de aleatorização r.

Teorema 14. *Se a Função-RSA é (t',ε')-segura, o esquema de assinaturas `PFDH-RSA`$[k_0]$ é (t,q_h,q_s,ε)-existencialmente-seguro no modelo do oráculo aleatório, onde*

$$k_0 \geqslant \log_2 q_s \tag{14}$$
$$t = t' - (q_h + q_s - q_h q_s) O(poli(k)) \tag{15}$$
$$\varepsilon = 4.\varepsilon' \tag{16}$$

Demonstração. Seja A um algoritmo capaz de (t,q_h,q_s,ε)-quebrar o `PFDH-RSA`. Construímos então um algoritmo S_A que utiliza A para (t',ε')-quebrar a Função-RSA. S_A recebe (Y,e,N) como entrada e deve calcular X tal que $X^e \equiv Y \pmod{N}$. Seja $k = |N|$ um parâmetro de segurança e seja a chave pública $PK = (N,e)$. S_A manterá um contador i ao longo da simulação, inicialmente valendo zero. S_A executa A com PK como parâmetro. Assumimos, sem perda de generalidade, que A não repete consultas ao oráculo $H(.)$.

Quando uma certa mensagem M aparece pela primeira vez em uma consulta (seja de *hash* ou de assinatura), o contador i é incrementado e $M_i = M$. S_A gera então uma lista L_i contendo q_s números aleatórios em $\{0,1\}^{k_0}$. S_A simula então os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H(M_i, r_j).**
  Escolhe um $x \xleftarrow{r} \{1, N-1\}$ e o associa a r_j.

Se $r_j \in L_i$ então devolva $H(M_i, r_j) = x^e \pmod N$;
caso contrário, devolva $H(M_i, r_j) = Yx^e \pmod N$.

- **Assina(M_i).**
  S_A escolhe o próximo $r' \in L_i$ aleatório. Se $H(M_i, r')$ não foi consultado ainda,
  - escolhe um $x \xleftarrow{r} \{1, N-1\}$ e o associa a r';
  - faz $H(M_i, r_j) = x^e \pmod N$.

  Seja x o valor associado a r'. Devolva a assinatura $(\sigma = x, r')$.

Finalmente, A retorna a falsificação $\varsigma = (\sigma^*, M^*, r^*)$. A consulta $H(M^*, r^*)$ foi realizada com alta probabilidade[13]. Seja então x o valor associado ao par (M^*, r^*). Seja L^* a lista relativa à mensagem M^*. Então, se r^* não pertencer a L^*, temos que $H(M^*, r^*) \equiv Yx^e$ e a falsificação nos fornece

$$\sigma^* \equiv H(M^*, r^*)^d \equiv Y^d x^{ed} \equiv Y^d x \pmod N .$$

É fácil então obter $X = \sigma^* x^{-1}$, invertendo assim a instância da Função-RSA em questão.

Precisamos agora calcular a probabilidade de sucesso $\varepsilon(k)$ de S_A. Como por definição A não pode ter consultado o oráculo `Assina(.)` na mensagem M^*, ele não pode ter obtido qualquer informação sobre L^*. Logo, a probabilidade de $r^* \notin L^*$ é $(1 - 2^{-k_0})^{q_s}$. Se $k_0 > \log_2 q_s$ e $q_s \geqslant 2$, obtém-se

$$\left(1 - 2^{-k_0}\right)^{q_s} \geqslant \left(1 - \frac{1}{q_s}\right)^{q_s} \geqslant \frac{1}{4} .$$

Como a probabilidade de sucesso de A é ε, temos que $\varepsilon' \geqslant \varepsilon/4$. Além disso, o tempo de execução é dominado pela geração das listas L_i ($q_h q_s$, q_h listas de tamanho q_s), mais as rotinas de resposta para $(q_s + q_h)$ consultas de oráculo. □

Os resultados de [Coron, 2002] para o `PSS-RSA` e para o `PFDH-RSA` implicam que $k_0 \approx \log_2 q_s$, digamos 40 ou 50 bits. Apesar da eficiência da demonstração, permitindo o uso de um parâmetro de segurança k menor, gostaríamos de não "desperdiçar" tanto espaço com a aleatorização da mensagem. É exatamente isso que nos traz ao próximo esquema de assinaturas que abordamos.

3.4.2.3. Katz & Wang

Em [Katz e Wang, 2003] os autores apresentam um resultado surpreendente: eles provam que a aleatorização utilizada no `PSS-RSA` e no `PFDH-RSA` pode ser reduzida a apenas um bit. Utilizando apenas este bit adicional, eles conseguem uma redução eficiente entre a Função-RSA e o esquema de assinatura. Apresentamos primeiro a definição do esquema de assinaturas, seguida pela demonstração de sua segurança.

[13] Devido à aleatoriedade de $H(.)$, a chance de esta consulta não ser feita é $\leqslant 1/2^k$.

Esquema de Assinatura 4. *Katz-Wang RSA* (`KW-RSA`)

Geração de Chaves. Selecionam-se aleatoriamente dois primos de n bits, p e q; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par (e,d) tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par (N,e) e a chave privada será o par (N,d).

Assinatura. Seja (N,d) a chave privada do usuário e seja M a mensagem a ser assinada. Se M já foi assinada antes, devolva a assinatura anteriormente calculada. Seja $H(.)$ a função de *hash* apropriada. Escolha um bit aleatório $b \xleftarrow{r} \{0,1\}$. A assinatura σ é calculada como

$$\sigma = H(M,b)^d \pmod{N}.$$

Verificação. Seja (N,e) a chave pública do usuário, seja (σ,M) o par assinatura-mensagem a ser verificado e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura σ se e somente se

$$\sigma^e \equiv H(M,0) \pmod{N} \quad \text{ou} \quad \sigma^e \equiv H(M,1) \pmod{N}.$$

Assumimos aqui que, apesar de haver duas assinaturas válidas para cada mensagem, usuários assinam apenas uma vez cada mensagem (i.e., se a assinatura da mesma mensagem for solicitada repetidas vezes, a mesma assinatura é devolvida). Este pode parecer um requisito complicado, requerendo que os usuários tenham "memória" de todas as mensagens assinadas, mas este não é o caso[14].

Teorema 15. *Se a Função-RSA é (t',ε')-segura, o esquema de assinaturas `KW-RSA` é (t,q_h,q_s,ε)-existencialmente-seguro no modelo do oráculo aleatório, onde*

$$t = t' - (q_h + q_s)O(\text{poli}(k)) \tag{17}$$
$$\varepsilon = 2.\varepsilon' \tag{18}$$

Demonstração. Seja A um algoritmo capaz de (t,q_h,q_s,ε)-quebrar o `KW-RSA`. Construímos então um algoritmo S_A que utiliza A para (t',ε')-quebrar a Função-RSA. S_A recebe (Y,e,N) como entrada e deve calcular X tal que $X^e \equiv Y \pmod{N}$. Seja $k = |N|$ o parâmetro de segurança e seja a chave pública $PK = (N,e)$. S_A executa A com PK como parâmetro. Assumimos, sem perda de generalidade, que A não repete consultas ao oráculo $H(.)$. S_A simula os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H(M_i,b).**
  Escolhe um bit aleatório $c \in \{0,1\}$ e $t_1,t_2 \xleftarrow{r} \{1,N-1\}$.
  Se $b = c$
  - $H(M_i,b) = Yt_{i1}^e \pmod{N}$
  - $H(M_i,\hat{b}) = t_{i2}^e \pmod{N}$;

  caso contrário,
  - $H(M_i,b) = t_{i2}^e \pmod{N}$

[14] Perceba que a escolha do bit b pode ser feita de forma determinística para uma dada mensagem, por exemplo, escolhendo $b = G(s||M)$, para uma função de hash G e a chave privada s.

– $H(M_i, \hat{b}) = Yt_{i1}^e \mod N$.

- **Assina(M_i).**
  Se M_i ainda não foi consultada ao oráculo $H(.)$,

  – consulta $H(M_i, b)$, para $b \xleftarrow{r} \{0,1\}$.

  Devolva a assinatura $\sigma = t_{i2}$.

Finalmente, A retorna a falsificação $\varsigma = (\sigma^*, M^*)$. A consulta $H(M^*, b)$, para algum $b \in \{0,1\}$, foi realizada com alta probabilidade. A assinatura σ^* será então a e-ésima raiz de t_{i2}^e ou de Yt_{i1}^e. Então, se $\sigma^* = t_{i2}^e$, S_A "FALHA". Caso contrário, $X = \sigma^* t i 1^{-1}$ e S_A consegue inverter a Função-RSA. Como as distribuições de t_{i2}^e ou de Yt_{i1}^e são indistinguíveis para A, a probabilidade de isso acontecer é $\frac{1}{2}$, ou seja $\varepsilon = 2\varepsilon'$. □

Este é um resultado realmente curioso e que pede uma discussão mais cuidadosa. Como destaca [Koblitz e Menezes, 2004], é, até certo ponto, um desafio à intuição que um bit possa fazer tanta diferença: esta redução é mais eficiente do que a apresentada para `HaS-RSA` por um fator de q_H, digamos 2^{40}. Por outro lado, não podemos ignorar os fatos: a redução está aí e é correta. Há pelo menos uma década pesquisadores tentam obter resultados semelhantes para a versão simples do `HaS-RSA` e nada foi conseguido. Aparentemente `KW-RSA` é mais seguro que `HaS-RSA` — pelo menos no modelo do oráculo aleatório.

3.4.3. Controvérsias ao Redor do Modelo do Oráculo Aleatório

Um dos tópicos de maior controvérsia atualmente na comunidade criptográfica é a validade do modelo do oráculo aleatório: de um lado, ele obviamente não representa um modelo fiel da realidade mas, por outro lado, nenhum exemplo *prático* de problema de segurança foi encontrado nesses quinze anos em que ele vem sendo largamente utilizado. Tentaremos aqui sucintamente esclarecer um pouco o que está por trás desta discussão sem ter qualquer pretensão, no entanto, de poder chegar a uma resposta definitiva em um texto de caráter introdutório como este.

Certamente todo o estudo formal da criptografia está baseado em modelos: o próprio modelo padrão não é uma descrição fiel da realidade, nem poderia ser. Existe um consenso geral, no entanto, de que as suposições feitas na sua concepção são bastante razoáveis e que vulnerabilidades que não se encaixem na sua "visão de mundo" (e.g. *side-channel attacks*[15]) devem ser analisadas através de outras ferramentas. No cerne do modelo do oráculo aleatório está, no entanto, uma suposição completamente irreal: a existência de funções *realmente* aleatórias é impossível de ser realizada na prática. Espera-se, no entanto, que este requisito seja "satisfatoriamente" satisfeito por funções de *hash* adequadamente escolhidas: que um protocolo seguro no modelo do oráculo aleatório não contivesse falhas "estruturais", e qualquer ataque a ele seria necessariamente um ataque às funções de *hash* utilizadas na sua implementação real.

[15] Ataques deste tipo buscam obter informações de ou então injetar falhas nos dispositivos físicos onde o processamento criptográfico é executado ou onde informações criptográficas são armazenadas. O leitor interessado poderá encontrar mais informações em [Koeune e Standaert, 2005] e [Bar-El et al., 2006]

Em [Canetti et al., 1998] os autores fornecem um esquema de assinaturas que é demonstrado seguro no modelo do oráculo aleatório mas que é *não-instanciável* no seguinte sentido: sua instanciação através de qualquer família de funções é insegura. O esquema é altamente artificial e não é em nada parecido com qualquer uso "normal" do modelo do oráculo aleatório, mas o resultado é inegável: existe uma distância explorável entre o modelo do oráculo aleatório e o modelo padrão. A este resultado seguiu-se outro de [Bellare et al., 2004], razoavelmente mais realista mas que ainda viola certas "regras práticas" de protocolos criptográficos[16], o que leva muitos pesquisadores a desacreditarem ambos os resultados como meramente teóricos.

A situação do modelo do oráculo aleatório é, portanto, incerta. Construções que não baseiam sua segurança neste modelo são ainda bastante ineficientes, ou dependem de suposições ainda menos estudadas. Por outro lado, os resultados de [Canetti et al., 1998] e [Bellare et al., 2004], apesar de encarados por muitos pesquisadores como distantes da realidade, são, no mínimo, preocupantes. O futuro próximo da pesquisa em segurança demonstrável deve encarar o problema de encontrar vulnerabilidades mais palpáveis no modelo do oráculo aleatório, ou melhor limitar o cenário onde elas podem existir às já conhecidas (e irreais).

3.5. Considerações finais

Procuramos, através deste texto cobrir os tópicos introdutórios mais importantes da segurança demonstrável, seguimos principalmente uma abordagem histórica dos resultados mais importantes obtidos na área, começando com os artigos seminais do início dos anos 80, que estabeleceram as bases formais sob as quais a criptografia viria a ser estudada nos anos vindouros, passando por avanços relevantes que surgiram nos anos seguintes e chegando ao paradigma do oráculo aleatório. Povoamos o texto com demonstrações de segurança dos mais variados esquemas, contra os mais variados tipos de ataques. Esperamos assim ajudar o leitor a se sentir mais confortáveis com a linguagem e as principais técnicas utilizadas nestas demonstrações.

Certamente não foi possível cobrir a totalidade dos assuntos relevantes à pesquisa nesta área e tópicos de extrema importância tiveram que ser deixados de lado: não tratamos de criptossistemas baseados em emparelhamentos bilineares ou esquemas baseados em identidade (e variações), não abordamos o modelo de grupo genérico ou as técnicas de replay de oráculo (e o lema da bifurcação), lidamos exclusivamente com esquemas de ciframento e de assinaturas, ignorando assim classes de protocolos mais complexos. Todos estes tópicos são de extrema importância e cabem em textos mais avançados sobre o assunto.

Sentimos, no entanto, que o conteúdo coberto no texto capacita pesquisadores a entender boa parte das provas de segurança que são produzidas atualmente, e isto é de extrema importância para o desenvolvimento da criptografia enquanto ciência. É extremamente negativo, por exemplo, que resultados de extrema expressividade, como

[16]Especificamente, a construção de [Bellare et al., 2004] é de um criptossistema híbrido, usando um criptossistema de chave pública para cifrar a chave de um esquema de chave privada que é usado para efetivamente cifrar os dados. Neste exemplo eles utilizam informação do esquema de chave pública no esquema de chave privada, o que é evitado em qualquer implementação realista deste tipo de criptossistema.

o RSA-OAEP [Bellare e Rogaway, 1994], possam ter falhas em suas demonstrações de segurança descobertas sete anos após a sua publicação e utilização em diversos padrões internacionais [Shoup, 2001]. Sentimos que isto acontece, em parte, pela não-familiaridade de boa parte da comunidade de pesquisa com as técnicas utilizadas na construção destas demonstrações. Esperamos, no entanto, que quanto mais pesquisadores sintam-se confortáveis com estas técnicas, e sintam-se aptos a entender as demonstrações de resultados que lhe parecem interessantes, maior será a probabilidade de erros nas mesmas serem mais rapidamente encontrados.

Referências Bibliográficas

[Bar-El et al., 2006] Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., e Whelan, C. (2006). The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2).

[Bellare et al., 2004] Bellare, M., Boldyreva, A., e Palacio, A. (2004). An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. Em [Cachin e Camenisch, 2004], p. 171–188.

[Bellare e Rogaway, 1993] Bellare, M. e Rogaway, P. (1993). Random oracles are practical: a paradigm for designing efficient protocols. Em *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, p. 62–73, New York, NY, USA. ACM Press.

[Bellare e Rogaway, 1994] Bellare, M. e Rogaway, P. (1994). Optimal asymmetric encryption. Em *EUROCRYPT*, p. 92–111.

[Bellare e Rogaway, 1996] Bellare, M. e Rogaway, P. (1996). The exact security of digital signatures - how to sign with rsa and rabin. Em *EUROCRYPT*, p. 399–416.

[Blum e Goldwasser, 1985] Blum, M. e Goldwasser, S. (1985). An *efficient* probabilistic public key encryption scheme which hides all partial information. Em *Proceedings of CRYPTO 84 on Advances in cryptology*, p. 289–302, New York, NY, USA. Springer-Verlag New York, Inc.

[Boneh e Franklin, 2003] Boneh, D. e Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615.

[Cachin e Camenisch, 2004] Cachin, C. e Camenisch, J., editores (2004). *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 de *Lecture Notes in Computer Science*. Springer.

[Canetti et al., 1998] Canetti, R., Goldreich, O., e Halevi, S. (1998). The random oracle methodology, revisited (preliminary version). Em *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, p. 209–218, New York, NY, USA. ACM Press.

[Chor e Goldreich, 1985] Chor, B. e Goldreich, O. (1985). Rsa/rabin least significant bits are 1-2- + 1/poly(log n) secure. Em *Proceedings of CRYPTO 84 on Advances in cryptology*, p. 303–313, New York, NY, USA. Springer-Verlag New York, Inc.

[Coron, 2002] Coron, J.-S. (2002). Optimal security proofs for pss and other signature schemes. Em *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, p. 272–287, London, UK. Springer-Verlag.

[Cramer e Shoup, 1998] Cramer, R. e Shoup, V. (1998). A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Em *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, p. 13–25, London, UK. Springer-Verlag.

[Diffie e Hellman, 1976] Diffie, W. e Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

[Goldreich, 2003] Goldreich, O. (2003). *Foundations of Cryptography: Volume I Basic Tools*. Cambridge University Press, Cambridge.

[Goldreich, 2004] Goldreich, O. (2004). *Foundations of Cryptography: Volume II Basic Applications*. Cambridge University Press, Cambridge.

[Goldwasser e Micali, 1982] Goldwasser, S. e Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. Em *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, p. 365–377, New York, NY, USA. ACM Press.

[Goldwasser e Micali, 1984] Goldwasser, S. e Micali, S. (1984). Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299.

[Katz, 2004] Katz, J. (2004). Lecture notes — advanced topics in cryptography. `http://www.cs.umd.edu/~jkatz/gradcrypto2/scribes.html`.

[Katz e Wang, 2003] Katz, J. e Wang, N. (2003). Efficiency improvements for signature schemes with tight security reductions. Em *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, p. 155–164, New York, NY, USA. ACM Press.

[Koblitz e Menezes, 2004] Koblitz, N. e Menezes, A. (2004). Another look at "provable security". Cryptology ePrint Archive, Report 2004/152.

[Koeune e Standaert, 2005] Koeune, F. e Standaert, F.-X. (2005). *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures*, volume 3655 de *Lecture Notes in Computer Science*, capítulo A Tutorial on Physical Security and Side-Channel Attacks, p. 78–108. Springer, Berlin Heidelberg.

[Lipton, 1981] Lipton, R. J. (1981). How to cheat at mental poker. Proceedings of the AMS Short Course on Cryptology.

[Micali et al., 1988] Micali, S., Rackoff, C., e Sloan, B. (1988). The notion of security for probabilistic cryptosystems. *SIAM J. Comput.*, 17(2):412–426.

[Rabin, 1979] Rabin, M. O. (1979). Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT-LCS-TM-212, Massachusetts Institute of Technology.

[Rivest et al., 1978] Rivest, R. L., Shamir, A., e Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

[Shamir et al., 1979] Shamir, A., Rivest, R. L., e Adleman, L. M. (1979). Mental poker. Technical Report MIT-LCS-TM-125, Massachusetts Institute of Technology.

[Shoup, 2001] Shoup, V. (2001). Oaep reconsidered. Em *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, p. 239–259, London, UK. Springer-Verlag.

[Stinson, 2006] Stinson, D. R. (2006). *Cryptography: Theory and Practice*. Chapman & Hall/CRC, Boca Raton, London, New York, 3 edição.

[Yao, 1982] Yao, A. C. (1982). Theory and applications of trapdoor functions. Em *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, p. 80–91.

Capítulo 4

Soluções para o desenvolvimento de sistemas seguros

Milene Fiorio[1], Carlo O. Emmanoel[1], Paulo F. Pires[2] e Flávia C. Delicato[2]

[1]Instituto de Matemática – Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 2324 – 20.001-970 – Rio de Janeiro – RJ – Brasil

[2]Centro de Ciências Exatas, Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Campus Universitário – Lagoa Nova – 59072-970 – Natal – RN – Brasil

milenefc@posgrad.nce.ufrj.br, carlo@nce.ufrj.br,
paulo.pires@dimap.ufrn.br, flavia.delicato@dimap.ufrn.br

Abstract

Modern Information systems are highly concerned with security and it is not rare the occurrence of security failures and vulnerabilities. Most of these problems are due to the dissociation of security requisites from the initial system conception. Security must be aggregated to the software development process to guarantee safety by design as an integral part of system construction. The development of security design patterns embedded into a model driven approach is an answer to help architects and designer to build systems with improved security.

Resumo

Apesar de grande parte dos sistemas de informação possuir a segurança como requisito fundamental, constantemente há noticias sobre vulnerabilidades e falhas de segurança. Isto se deve principalmente pelo fato deste requisito ser raramente considerado nos estágios iniciais do desenvolvimento de software e ser delegado a segundo plano ao longo do mesmo. Portanto, são necessárias novas formas de desenvolver software seguro, baseadas não somente na aplicação das teorias existentes como na adoção de um processo de desenvolvimento que considere os requisitos de segurança como parte integral do projeto de construção de software. Neste contexto, a utilização de padrões de projeto de segurança e de uma abordagem orientada a modelos pode auxiliar arquitetos e projetistas a construir sistemas seguro.

4.1. Introdução

No desenvolvimento de *software*, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento sendo comum, portanto, que a busca por um *software* de maior qualidade passe necessariamente por uma melhoria nesse processo. [TSUKOMO 1997].

Nos últimos anos, empresas têm buscado certificações ISO ou CMM como meio de comprovar a qualidade no seu processo de desenvolvimento de *software* e, desta maneira, tornarem-se competitivas em um mercado cada vez mais exigente. Com isso, torna-se de grande importância o desenvolvimento de métodos e técnicas que permitam uma avaliação abrangente da qualidade dos processos e dos produtos de *software*, para garantir que o usuário receba produtos dentro das especificações por ele definidas e esperadas. Isto pode ser alcançado através da definição e especificação de características relevantes de qualidade do produto, com as respectivas avaliações, sempre que possível, usando métricas válidas e aceitas.

Segundo [ISO/IEC 9126-1 (2001)], a qualidade de *software* deve ser medida através da funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade do *software*. Cada fator possui uma série de subfatores que também devem ser medidos, entre eles, a segurança e proteção do *software*. Para garantir a qualidade de segurança é comum utilizar um arcabouço de segurança, onde um arcabouço representa uma solução para um conjunto de problemas com características similares o qual disponibiliza componentes com capacidade de capturar funcionalidades comuns a várias aplicações. Ou seja, um arcabouço representa uma solução quase completa para um determinado problema, de forma independente do domínio da aplicação, feito normalmente por desenvolvedores experientes. Um arcabouço deve ser extensível, bem documentado e principalmente reutilizável.

Aplicações de comércio eletrônico e outras similares no âmbito da internet possuem a segurança como requisito fundamental e apesar da grande variedade de arcabouços de segurança disponíveis, constantemente há noticias sobre vulnerabilidades e falhas de segurança em sistemas de software [FINK et al 2004]. Á medida que os arcabouços aparecem, eles se tornam rapidamente ultrapassados por diferentes motivos. Na maioria das vezes porque eles não oferecem um modelo arquitetural suficientemente flexível para acompanhar as necessidades do negócio, o qual está em desenvolvimento ou porque eles limitam o escopo da arquitetura de que o negócio necessita. Devido à diversidade de requisitos de negócio de cada aplicação, arquitetos de software e analistas necessitam cada vez mais criar seus próprios arcabouços e manter suas integridades e coerência em relação às necessidades do negócio as quais eles atendem.

Além disso, existe uma grande lacuna entre as soluções teóricas e o que é de fato implementado na área de segurança. Um dos problemas que contribuem para a construção de software com segurança fraca é o fato de este requisito ser raramente considerado nos estágios iniciais do desenvolvimento de *software* e ser relegado a segundo plano ao longo do mesmo, provocando problemas de segurança tanto na arquitetura da aplicação quanto na lógica implementada.

De acordo com os problemas relatados, são necessárias novas formas de desenvolver software seguro, baseadas não somente na aplicação das teorias existentes como na adoção de um processo de desenvolvimento que considere os requisitos de

segurança como parte integral do projeto de construção de *software* [FERNANDEZ 2000]. Além disso, o aumento da complexidade dos sistemas aliado à dinamicidade inerente às regras de negócio das aplicações atuais faz com que o processo de desenvolvimento de segurança necessite cada vez mais de arcabouços flexíveis e maior gerência de requisitos e mudanças. Por outro lado, a necessidade de baixo tempo de resposta no desenvolvimento de soluções faz com que os riscos de sucesso dos projetos se tornem mais críticos.

Uma das abordagens que pode auxiliar arquitetos e projetistas a construir sistemas seguros consiste no uso de padrões de projetos. Padrões de projeto foram introduzidos como uma forma de identificar e apresentar soluções a problemas recorrentes na programação orientada a objetos. Joseph Yoder e Jeffrey Barcalow [YODER, BARCALOW 1997] foram uns dos primeiros a utilizar esta abordagem como uma tentativa de criar arcabouços de segurança bem projetados. O uso de padrões nesse contexto justifica-se, pois enquanto muitos problemas de segurança são novos ou complicados, uma grande parte de outros problemas são bem conhecidos e possuem soluções bem estabelecidas. No espírito do provérbio “É melhor ensinar a pescar do que dar o peixe”, é melhor explicar como utilizar padrões de projeto para criar arquiteturas de segurança próprias a utilizar arquiteturas prontas que não atendem as necessidades do negócio e que muitas vezes o arquiteto e o projetista têm que modificar para fazer o melhor acoplamento possível.

Contudo, para obter os benefícios da utilização de padrões de projeto é necessário saber quando e como utilizá-los a partir de suas definições. Neste contexto, uma abordagem orientada a modelo pode auxiliar o processo de desenvolvimento e de criação de arcabouços de segurança, promovendo uma implementação controlada dos padrões de projeto e, desta forma, garantindo aderência às diretrizes arquiteturais da organização e às questões de segurança.

O objetivo desse capítulo é abordar aspectos teóricos e práticos das soluções de segurança existentes para o desenvolvimento de sistemas de software. Para atingir esse objetivo, serão apresentados os modelos básicos de segurança, integração de soluções de segurança na modelagem de sistemas incluindo a aplicação de padrões de projeto de segurança e o novo paradigma de desenvolvimento orientado a modelos denominado MDD (*Model Driven Development*).

Na seção 1.2, apresentamos os fundamentos de segurança em ambiente computacional, onde são descritos alguns conceitos e definições, como por exemplo, a definição de modelo de segurança e os principais modelos existentes. Na seção 1.3, será feita uma análise de modelagem de sistemas seguros, apresentando as técnicas de integração de soluções de segurança nas fases de análise e projeto de sistemas de software. Na seção 1.4, será apresentada uma conclusão deste trabalho.

4.2. Fundamentos de Segurança em Ambiente Computacional

Com a interligação dos computadores, através de redes físicas ou *wireless* (conexões de equipamentos sem fio), estes computadores se tornam mais suscetíveis a acessos não autorizados e o conceito de segurança, importante em ambientes físicos se torna relevante para ambientes distribuídos. Desta forma é necessário que os usuários de um sistema computacional tenham acesso somente ao que lhe seja pertinente. Pois caso isso

não ocorra, o sistema se torna mais suscetível a ataques podendo causar sérios danos ao sistema.

Nesta seção iremos introduzir uma série de conceitos, modelos e técnicas utilizados com o objetivo de garantir a segurança da informação em sistemas computacionais.

4.2.1. Conceito de segurança

Nos últimos anos os sistemas computacionais de segurança têm sido alvo de muito interesse pela grande maioria das pessoas que se utilizam deles direta ou indiretamente. Na mídia, não são poucos os relatos encontrados sobre notícias de pessoas, instituições ou empresas que tiveram enormes prejuízos causados por ações intrusivas executadas pelos especialistas deste tipo de delito, que são conhecidos como *hackers*. Devido ao conhecimento de fatos como estes e muitas vezes à própria convivência com os mesmos é que chegamos à conclusão de que a principal finalidade da "segurança" consiste em promover a defesa de ataques externos como forma de evitar prejuízos de qualquer natureza que em sua grande parte geram perdas financeiras e até morais ao prejudicado.

A segurança em sistemas computacionais consiste em uma disciplina que busca, através de todos os seus conceitos, metodologias e técnicas empregadas, manter as propriedades de um sistema de forma que ele não seja alvo de possíveis ações danosas praticadas por entidades não autorizadas junto às informações e recursos nele existentes. De acordo com [BISHOP 2003], a segurança de sistemas computacionais visa à proteção contra a indisponibilidade, vazamento da informação e a leitura ou modificação não-autorizada das informações. Além de garantir dados e informações, a segurança visa prevenir, detectar, conter e documentar eventuais ameaças aos sistemas computacionais.

Existem hoje na literatura várias definições para segurança e, em sua grande maioria, elas incluem a necessidade de se manter no sistema um conjunto de propriedades [DENNING 1982]:

- **Confidencialidade**: A informação poderá ser acessada somente por usuários autorizados. Este acesso é restrito a usuários devidamente cadastrados para que impeça usuários não autorizados de terem acesso à informação
- **Integridade**: Prover a garantia que a informação deve ser retornada em sua forma original no momento em que foi armazenada, protegendo contra modificações não autorizadas
- **Disponibilidade**: A informação ou sistema de computador deve estar disponível no momento em que a mesma seja requisitada

Alguns autores tratam que um sistema será seguro se abordar outros dois itens [BISHOP 2003, LANDWEHR 2001]:

- **Autenticidade:** Garante que a informação ou o usuário da mesma é autêntico; Atesta a origem do dado ou informação
- **Não repúdio:** Não é possível negar (no sentido de dizer que não foi feito) uma operação ou serviço que modificou ou criou uma informação

Outros três itens podem ser desejáveis para se ter um ambiente seguro [BISHOP 2003]:

- **Legalidade:** Garante a legalidade (jurídica) da informação; Aderência de um sistema à legislação; Característica das informações que possuem valor legal dentro de um processo de comunicação, onde todos os ativos estão de acordo com as cláusulas contratuais pactuadas ou a legislação política institucional, nacional ou internacional vigentes
- **Privacidade:** Uma informação pode ser considerada confidencial, mas não privada. Uma informação privada deve ser vista/lida/alterada somente pelo seu proprietário. Consiste em garantir que a informação não será disponibilizada para outras pessoas (neste caso é atribuído o caráter de confidencialidade a informação)
- **Auditoria:** Verificar e mapear os passos que um determinado processo realizou ou que uma informação foi submetida, identificando os participantes, os locais e horários de cada etapa. Auditoria em software significa uma parte da aplicação, ou conjunto de funções do sistema, que viabiliza uma auditoria, consistindo no exame do histórico dos eventos dentro de um sistema para determinar quando e onde ocorreu uma violação de segurança

4.2.2. Política de Segurança

O termo “Política de Segurança” pode ter vários significados dependendo do nível em que é aplicado. Vendo sob o ponto de vista administrativo de uma instituição, podemos definir como sendo um conjunto de leis e práticas utilizadas pela instituição para gerenciar, proteger e distribuir suas informações. E esta mesma ótica administrativa em relação à segurança da informação deve-se refletir nos ambientes computacionais, onde a política de segurança passa a ser definida como sendo o conjunto de regras e serviços que visam especificar como um sistema provê os seus recursos mantendo sempre as propriedades de confidencialidade, integridade e disponibilidade.

Quando um administrador de um ambiente computacional necessita tomar decisões para realizar a proteção do ambiente computacional, o mesmo necessita estabelecer regras; definindo as funcionalidades que irá oferecer, e qual será a facilidade de utilizá-la. Às vezes se torna complicado realizar decisões sobre segurança, sem que tenham determinado quais são as suas metas de segurança.

O estabelecimento de um conjunto de regras irá definir as políticas de segurança, o principal objetivo de uma política de segurança é informar aos usuários, equipe e gerentes, as suas obrigações para a proteção da tecnologia e do acesso à informação. Os objetivos determinados devem ser comunicados a todos os usuários, pessoal operacional, e gerentes através da política de segurança adotada.

As políticas de segurança são classificadas em duas categorias em relação ao controle de acesso: as discricionárias e as obrigatórias. Nas discricionárias os acessos a cada recurso ou informação são manipulados sem restrição pelo proprietário ou responsável do sistema, baseando-se na idéia de que este proprietário deve determinar quem tem acesso a estas informações segundo a sua vontade (à sua discrição). Já nas obrigatórias, as autorizações de acesso são definidas através de um conjunto incontornável de regras que expressam algum tipo de organização envolvendo a segurança das informações no sistema como um todo [MACKENZIE 1997], baseando-se em uma administração centralizada de segurança, a qual dita qual serão as regras de acesso à informação.

O objetivo da utilização de políticas de segurança no ambiente computacional é tornar o sistema mais seguro contra intrusos. Com a adoção de políticas de segurança tanto discricionárias quanto obrigatórias tornam o ambiente mais seguro, mas não livre de ações que tentam contornar as políticas de segurança adotadas.

4.2.3. Violações de Segurança

As regras definidas pela política de segurança determinam as entidades que serão autorizadas e responsáveis pelas ações executadas sobre todas as informações mantidas no sistema. Essas entidades são normalmente identificadas como "principais"**.** Dependendo do nível de aplicação desta política pode-se caracterizar como sendo principal um usuário, um processo ou ainda uma máquina dentro de uma rede de computadores. Uma entidade que ganha acesso a recursos de um sistema computacional de forma ilícita, violando assim a política de segurança, é denominada de intruso.

As violações de segurança em sistemas computacionais se traduzem como sendo a arte de burlar de alguma forma a política de segurança. A Tabela 1.1. mostra os tipos de violação alocados em contraposição às propriedades de segurança não verificadas [AMOROSO 1994].

Tabela 1.1. Relação de violações as propriedades de segurança

Tipo de Violação (TV)		Propriedade de Segurança Violada
I	revelação não autorizada	confidencialidade
II	modificação não autorizada	integridade
III	negação de serviço	disponibilidade

Antes de verificarmos as principais violações existentes no ambiente distribuído, devemos fazer algumas definições que serão importantes para o nosso entendimento. Uma ameaça é caracterizada como sendo uma ação possível que, uma vez concretizada, produz efeitos indesejáveis sobre os dados ou recursos de sistema. Uma ameaça, quando posta em ação, é identificada como um ataque à segurança do sistema. Entende-se por vulnerabilidade como sendo uma falha ou característica indevida existente no sistema oriunda de falhas de concepção, implementação ou de configuração, a qual expõe os recursos deste sistema computacional a ataques e que podem ser exploradas para concretizar uma ameaça.

Uma maneira de ilustrarmos o relacionamento entre ameaças, vulnerabilidades e ataques é fazendo uma analogia com uma casa. Uma ameaça associada a uma casa é o roubo de móveis, dinheiro e eletrodomésticos. Vulnerabilidades podem ser comparadas a uma janela aberta ou uma porta que não esteja trancada. O ataque consiste na invasão propriamente dita com o conseqüente roubo dos bens existentes.

A Tabela 1.2 relaciona as ameaças mais comumente presentes em relatos de violações de segurança em sistemas distribuídos [AMOROSO 1994].

Tabela 1.2. Ameaças mais comuns ao sistema de informática

Ameaça	TV	Descrição
Mascaramento (masquerade/spoofing)	II	Técnica utilizada para "se fazer passar"/forjar uma identidade. Utilizada por exemplo em capturadores de senha, IP spoofing, etc.

Bypassing control	I	Técnica utilizada para explorar vulnerabilidades do sistema. É comum a utilização desta técnica em invasões de sistemas com versões (patchs, service packs) desatualizadas.
Código maléfico (malicious code)	I, II, III	Técnica que utiliza software com código contendo partes aparentemente inofensivas ou invisíveis. Estas, quando executadas, comprometem a segurança dos recursos do sistema. Utilizada por exemplo em cavalos de tróia, vírus, bombas lógicas, etc.
Backdoor/Trapdoor	I, II	Técnica utilizada para inserir/criar funções/escutas no sistema, que aceitam entradas específicas e permitem contornar/driblar os mecanismo s de segurança do sistema.
Inspeção de lixeira (midia scavenging)	I	Técnica que consiste em bisbilhotar/revirar lixeiras pro curando papéis, discos, etc, com informações importantes que não foram adequadamente destruídas.
Scanning	I	Técnica utilizada para bisbilhotar recursos do sistema com algum objetivo específico. Um ataque deste tipo utiliza-se de um software scanner para ser executado.
Negação de Serviço	III	Técnica utilizada para impedir o acesso legítimo ao sistema ou a algum recurso deste. Alguns exemplos desta técnica são Distributed Denial of Service, Worms, repetidor de discagem e diversas variações de Denial of Service.
Ataque nas Comunicações	I, II	Técnica utilizada para escuta, interceptação, inserção ou alteração de mensagens durante a transmissão. Geralmente, a escuta da comunicação com um software sniffer, por exemplo, precede o uso de ataques como Man in-the-middle ou roubo da sessão autenticada.

4.2.4. Modelos de Segurança

Os modelos de segurança são formas de descrever as políticas de segurança, determinando o comportamento de entidades administradas pela política e também as regras que definem a evolução desta política.

Quando alguns modelos matemáticos formais de segurança são utilizados para descrever políticas de autorização, os mesmos permitem de alguma maneira, verificações de que a política é coerente, e servem como guia para implementações de esquemas de autorização, correspondentes às especificações contidas no modelo [LANDWEHR 1981].

Os modelos de segurança correspondem a descrições mais detalhadas do comportamento de um sistema. Eles atuam sempre segundo regras de uma política de segurança estabelecida. Estes modelos são representados na forma de um conjunto de entidades e relacionamentos [GOGUEN 1982]. Na literatura é comum encontrarmos os modelos divididos em três tipos básicos [SANDHU 1996, OSBORN 2000]:

- Baseados em identidade ou discricionários (*discretionary)*;
- Baseados em regras ou obrigatórios (*mandatory)*;
- Baseados em papéis (*roles*);

4.2.4.1. Modelos discricionários (discretionary)

Nos modelos discricionários, os direitos de acesso aos recursos ou informações são especificados para cada sujeito, pelo proprietário da informação ou recurso. As requisições para utilização de recursos são analisadas por um mecanismo de segurança discricionário e o acesso é concedido de acordo com as regras de autorização [LANDWEHR 1981, AMOROSO 94].

Os modelos discricionários se baseiam em conceder e retirar privilégios. O controle poderá ser centralizado, isto é, a autorização é administrada por um controle central, normalmente o administrador do sistema. Podem também ter um controle descentralizado de maneira que o proprietário da informação possui o direito de conceder ou retirar os privilégios.

A adoção destes modelos para aplicação em políticas de segurança tem a vantagem de torná-las flexíveis, fazendo com que tais políticas possam ser utilizadas por vários tipos de sistemas e aplicações comerciais e industriais e, conseqüentemente, atualmente serem as mais utilizadas. Vários modelos foram propostos e implementados em vários sistemas, porém certos requisitos de segurança podem não ser totalmente satisfeito, como no caso em que um usuário com permissão de leitura a determinada informação poderá transferi-la para um objeto de outro usuário que não possui esta permissão. As políticas de controle de acesso discricionárias não permitem classificar os objetos e sujeitos no que diz respeito a confidencialidade. Com isto, não é possível criar uma política que possa restringir os privilégios de acesso a informações confidenciais com base na classificação dos sujeitos, independentemente de especificações determinadas pelo administrador. Este tipo de política é importante quando se deseja limitar a autoridade do administrador, protegendo informações confidenciais, muitas vezes vitais, como no caso dos sistemas militares. Mas sua aplicação pode ser bem mais abrangente, sendo fundamental em qualquer sistema onde a confidencialidade das informações for importante.

Os modelos discricionários, na sua essência, são baseados no modelo matriz de acesso proposto por Lampson [LAMPSON 1971]. Neste modelo, o estado de proteção do sistema é representado através de uma matriz, onde as linhas correspondem aos sujeitos e as colunas correspondem aos objetos do sistema. Esta matriz relaciona-se através dos objetos, que podem ser definidos como sendo os recursos do sistema, dos sujeitos, que são as entidades ativas existentes no sistema e dos atributos de acesso, que são os direitos ou permissões de acesso (*read*, *write*) cabíveis ao sistema. Cada sujeito (S1, S2, S3, etc.) é representado por uma linha da matriz e cada objeto (O1, O2, etc.) por uma coluna. Na célula de intersecção das duas, onde ambas (linha x coluna) se encontram são especificados os atributos de acesso do respectivo sujeito em relação ao objeto considerado. O modelo de matriz de acesso pode ser visto na Figura 1.1.

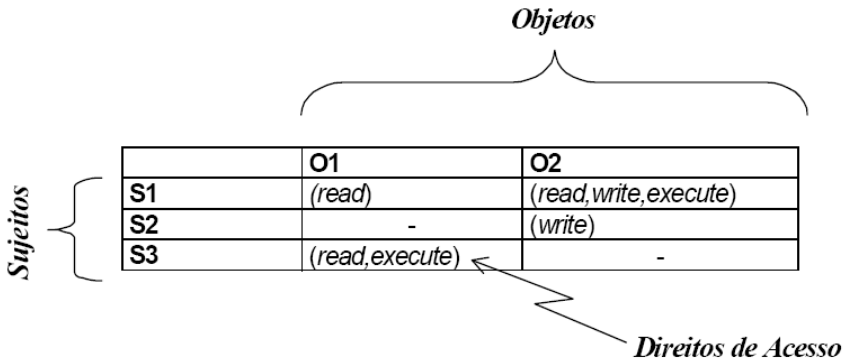


Figura 1.1. Modelo Matriz de Acesso

Se considerarmos uma coluna da matriz de acesso, veremos que a relação de todos os sujeitos existentes, com seus respectivos direitos de acesso sobre o objeto, correspondem à coluna, formando o que é identificado como sendo uma lista de controle de acesso (LCA) do objeto considerado. As LCAs correspondem a uma forma de apresentação do modelo matriz de acesso. Na Tabela 1.3, é apresentado um conjunto de LCAs onde cada lista corresponde a uma entrada do objeto correspondente.

Tabela 1.3. Listas de Controle de Acesso

Objeto	Listas de Controle de Acesso
O1	S1(read), S3(read, execute)
O2	S2(write), S1(read, write, execute)

A LCA de um objeto permite uma fácil revisão dos acessos autorizados a um objeto. Outra operação bastante simples com o uso de LCAs é a revogação de todos os acessos a um objeto, bastando para isso substituir a LCA corrente por outra vazia. Por outro lado, a determinação dos acessos aos quais um sujeito está autorizado é bastante problemática; é necessário percorrer todas as LCAs do sistema para fazer este tipo de revisão de acesso. A revogação de todos os acessos de um sujeito também requer que todas as LCAs sejam inspecionadas e, eventualmente, modificadas.

A matriz de acesso pode ainda apresentar uma outra representação através de Listas de Competência. Neste tipo de representação cada sujeito é associado a uma lista que indica, para cada objeto no sistema, os acessos que o sujeito está autorizado a efetuar. Isso corresponde a armazenar a matriz de acesso por linhas. Na Tabela 1.4, cada lista é representada em uma entrada por sujeito da matriz.

Tabela 1.4. Listas de Competências

Sujeitos	Listas de Competências
S1	O1(read), O2(read, write, execute)
S2	O2(write)
S3	O1(read, execute)

As competências permitem uma fácil verificação e revogação dos acessos autorizados para um determinado sujeito. Porém, em contrapartida, a determinação de quais sujeitos poderão acessar um objeto requer a inspeção de todas as competências do sistema. As vantagens e desvantagens de LCAs e competências são, como as próprias estratégias, ortogonais entre si.

Competências são vantajosas em sistemas distribuídos. A posse de uma competência é suficiente para que um sujeito obtenha o acesso autorizado por esta competência. Em um sistema distribuído, isso possibilita que um sujeito se autentique uma vez, obtenha a sua lista de competência e apresente estas competências para obter os acessos aos quais ele está autorizado; os servidores precisam apenas verificar a validade dessa competência para liberar o acesso.

4.2.4.2. Modelos Obrigatórios (mandatory)

Os Modelos obrigatórios têm em seu esquema de autorização um conjunto de regras incontornáveis que expressam um tipo de organização envolvendo a segurança das informações no sistema como um todo. O modelo obrigatório prevê que os usuários, objetos e recursos do sistema sejam rotulados; os rótulos dos objetos seguem uma classificação específica enquanto os usuários de acesso possuem níveis de habilitação. Os controles que determinam as autorizações de acesso são baseados numa comparação da habilitação do usuário com a classificação do objeto [AMOROSO 1994].

Os modelos obrigatórios que caracterizam as políticas obrigatórias baseiam-se em uma administração centralizada de segurança, a qual dita regras de acesso à informação. As mesmas definem regras e estruturas válidas no âmbito de um sistema, normalmente especificando algum tipo de política multinível. As políticas multiníveis estão baseadas em algum tipo de classificação ao qual estão submetidos os acessos dos sujeitos aos objetos. Elas visam ajudar na decisão de acesso em um ambiente com classificações de segurança, ou seja, ambientes onde informações e documentos possuem níveis de segurança, como por exemplo, secreto, confidencial, etc.

A seguir serão descritos os principais modelos obrigatórios constantes na literatura.

4.2.4.2.1. Modelo Bell-LaPadula (BLP) de confidencialidade

O modelo Bell e Lapadula (BLP) tem este nome devido aos cientistas David Bell e Leonard Lapadulla que desenvolveram este modelo no inicio da década de 70. O referido modelo é baseado nos procedimentos de manipulação de informação em áreas ligadas à segurança nacional americana [AMOROSO 94, BELL 1776].

A idéia principal deste modelo é acrescentar controles de acesso obrigatório aos controles de acesso discricionário, desta forma, aplicando políticas de segurança que impeçam o fluxo de informações de níveis mais altos aos níveis mais baixos de segurança. As permissões de acesso são definidas através de uma matriz de acesso e de rótulos de segurança. A matriz de acesso armazena os direitos de cada sujeito sobre os objetos do sistema e pode ser modificada pelos sujeitos através de regras específicas.

O modelo Bell-LaPadula priva a confidencialidade e está baseado na classificação dos elementos de segurança, que definem a política de acesso ao sistema. Esta classificação é expressa por níveis de segurança, onde cada nível é definido por

dois componentes: uma classificação e um conjunto de categorias. O modelo BLP classifica as informações em quatro níveis hierárquicos de sensibilidade (não-classificada, confidencial, secreta e ultra-secreta – respectivamente do menor para o maior nível). O conjunto de categorias não possui nenhuma hierarquia e os seus elementos são definidos de acordo com o ambiente ou área de aplicação do modelo.

Ao utilizar este método de classificação reduz-se a complexidade das regras, aproximando o modelo BLP dos ambientes computacionais de uso corrente, sem enfraquecê-lo em nenhum aspecto.

O sistema é descrito em termos de sujeitos que acessam objetos, onde cada sujeito possui uma habilitação e cada objeto possui uma classificação. A cada sujeito está associado também um rótulo corrente de segurança - representando a classificação mais alta dentre as informações já consultadas pelo sujeito no sistema, sendo portanto, uma classificação flutuante (dinâmica). Para evitar a revelação da informação a sujeitos não autorizados, basicamente, o modelo BLP impõe duas propriedades que devem ser satisfeitas para que a segurança do sistema seja garantida.

- **Propriedade de Segurança Simples *ou No Read Up (NRU):*** um sujeito pode somente observar informações para as quais esteja habilitado, evitando assim que um sujeito de nível inferior leia informações de um nível mais elevado que o seu nível de segurança (habilitação e compartimento)*;*
- **Propriedade Estrela ou No Write Down (NWD):** um sujeito só pode escrever em objetos cujos níveis de segurança dominam o nível de segurança do sujeito.

Estas propriedades devem ser satisfeitas para que se possa evitar que a informação de um nível mais alto acabe fluindo para níveis baixos de segurança, caracterizando a revelação não autorizada de informação. Além das duas propriedades descritas o modelo também mantém um controle discricionário por nível de segurança (*discretionary security propriety*) que reflete os princípios de autorização expressos no modelo matriz de acesso.

O modelo de *BLP* apresenta entre as suas principais limitações [MCLEAN 1990, MACKENZIE 1997], a escrita às cegas e a superclassificação da informação.

A Escrita às Cegas é assim denominada porque a propriedade estrela permite que um sujeito escreva num objeto de nível de sensibilidade superior a sua habilitação, o que pode determinar a destruição de informações sem caracterizar uma violação de segurança e das regras do modelo propriamente dito. O cenário de escritas cegas torna-se uma preocupação na medida em que o mesmo sujeito considerado inadequado para ver o conteúdo de um objeto possui permissão para fazer modificações arbitrárias neste mesmo objeto. Isto pode causar problemas de integridade que só podem ser resolvidos através de alterações nas regras do modelo BLP. Por exemplo, escritas em níveis de segurança mais altos que o corrente podem ser proibidas, ou seja, um sujeito somente poderia escrever em um objeto que tivesse o mesmo nível de segurança. Entretanto, tal modificação restringe, de certa forma, o modelo BLP e muda seu enfoque, que deixa de ser exclusivamente a ameaça de revelação não autorizada e passa a ser uma combinação de revelação e integridade. Por outro lado, a adoção de propriedade-* revisada é bastante comum em implementações de sistemas computacionais que seguem o modelo BLP.

A superclassificação *da informação* é determinada pelas regras do modelo que definem uma espécie de fluxo da informação dos níveis de segurança mais baixos para os mais altos, o que dificulta, com o tempo, a manipulação da informação no sistema. Por exemplo, se um sujeito com rótulo corrente de segurança SECRETO deseja copiar um arquivo CONFIDENCIAL, a propriedade-* impõe que a cópia tenha classificação SECRETO, mesmo que as informações ali contidas possuam classificação CONFIDENCIAL. Ao longo do tempo, isso faz com que as informações subam no reticulado de rótulos de segurança, recebendo classificações sucessivamente maiores. A superclassificação da informação provoca a necessidade de reclassificações periódicas dos objetos (através de sujeitos de confiança) apenas para garantir a usabilidade de sistemas baseados no modelo BLP.

4.2.4.2.2. O Modelo Biba de Integridade

O modelo obrigatório de integridade BIBA [AMOROSO 1994, BIBA 1977] é descrito como o inverso do modelo BLP [AMOROSO 1994]. Esta é uma descrição razoável, pois as regras básicas do modelo BIBA são bem próximas das regras do modelo BLP. No modelo de BIBA são definidas regras onde um sujeito estando em um nível de integridade mais elevado não pode ler um objeto que esteja em um nível de integridade inferior ao seu (*no read down* NRD). Também estabelece que um sujeito estando em um baixo nível de integridade não poderá escrever em um objeto em um nível de integridade superior ao seu (*no write up* NWU) [AMOROSO 1994].

O modelo obrigatório BIBA está baseado na integridade e tanto uma regra quanto outra deste modelo, são opostas ao modelo BLP que em sua proposta visa a confidencialidade. Deste modo, os níveis mais elevados de integridade devem ser vistos como uma associação daqueles sujeitos e objetos que devem ter um nível de integridade mais elevado, e os níveis mais baixos devem ser vistos como uma associação daqueles sujeitos e objetos que podem tolerar um menor nível de integridade.

O modelo BIBA pode ser representado através de um diagrama de níveis de maneira mais objetiva. As linhas horizontais do diagrama representam os níveis de integridade e as ações que são negadas pelo contexto geral do modelo. O diagrama do modelo BIBA pode ser melhor visualizado na Figura 1.2.

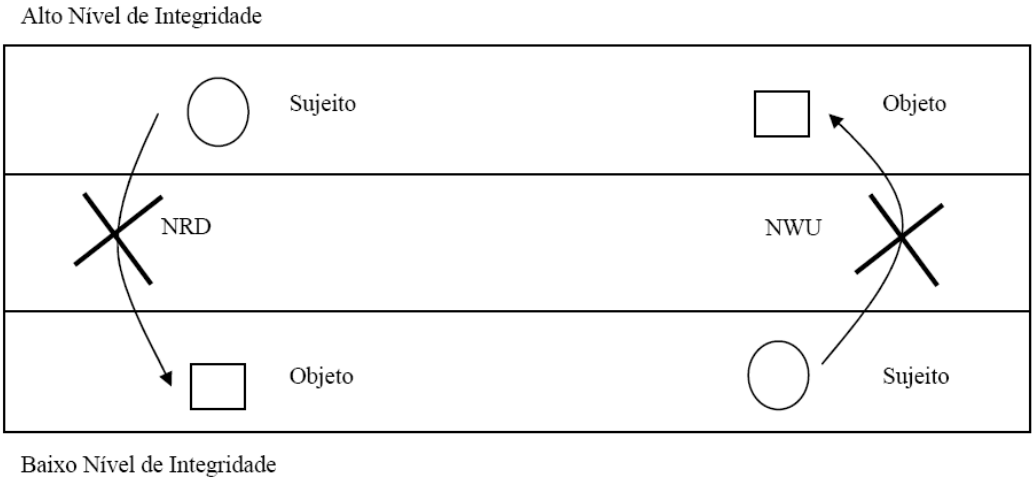


Figura 1.2. Regras do Modelo Obrigatório de Integridade BIBA

Uma das vantagens do modelo BIBA, é a incorporação de algumas características das regras do modelo BLP incluindo a simplicidade e atributos intuitivos. Isto é, os desenvolvedores de sistemas podem facilmente entender as regras de NWD e NRU, podendo incorporá-las em projetos de decisões de sistemas.

O Modelo da marca d´água baixa do sujeito [AMOROSO 1994] introduz um leve relaxamento nas regras de leitura de sujeito mais íntegros em objetos menos íntegros. A revisão do modelo obrigatório BIBA de integridade não permite que os sujeitos de alta integridade leiam os objetos de baixa integridade. Isto pretende assegurar que a informação do sujeito de alta integridade, não seja corrompida pela baixa integridade do objeto. Entretanto no modelo da marca d´água baixa do sujeito é permitida a leitura de objetos de menor integridade por sujeitos mais íntegros, mas o resultado de tal leitura é o rebaixamento do nível de integridade do sujeito ao nível do objeto lido [AMO94]. As características deste modelo são mostradas na Figura 1.3.

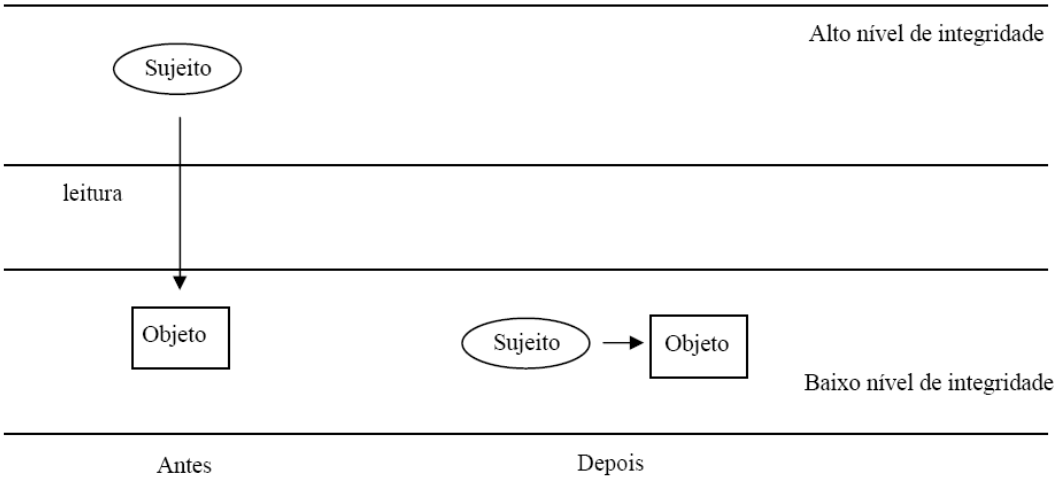


Figura 1.3. Modelo BIBA da Marca D`Água Baixa do Sujeito

Os modelos descritos são relativamente simples e intuitivos, e conseguem demonstrar de uma maneira bastante intuitiva a sua capacidade de conservar a propriedade de segurança considerada (confidencialidade no BLP e integridade no Biba).

Entretanto, o modelo Biba, assim como o BLP, depende em demasia de sujeitos de confiança em situações práticas: a necessidade de um processo de confiança para aumentar ou reduzir a integridade de sujeitos ou objetos é especialmente problemática para a integridade. Outra crítica ao modelo Biba é a ausência de provisão de mecanismos para a promoção da integridade de um sujeito ou objeto. Cabe notar que todas as mudanças possíveis no modelo Biba preservam a integridade de todos os sujeitos ou rebaixam a integridade de algum sujeito ou objeto. Isso permite imaginar que, com o passar do tempo, os sistemas sofrem um rebaixamento do nível de integridade, proporcional a quanto os sujeitos e objetos migram para o nível mais baixo. Esta degradação da integridade da informação é análoga ao problema de superclassificação da informação no modelo BLP.

4.2.4.3. Modelos Baseados em Papéis (Role Based Access Control - RBAC)

Os modelos Baseados em Papéis (RBAC) têm como objetivo intermediar o acesso dos usuários à informação, com base nas atividades que são por eles desenvolvidas no sistema. A idéia central é que o usuário desempenhe diferentes papéis (*roles*) em um sistema. Um papel pode ser definido como um conjunto de atividades e responsabilidades associadas a um determinado cargo ou função em uma organização. Assim, no RBAC, as permissões são conferidas aos papéis e os usuários são autorizados a exercer papéis (Figura 1.4). O controle de acesso baseado em papéis facilita a gerência de autorização, porque quando o usuário muda de atribuição – sendo, portanto, desassociado de um papel e assumindo um outro – a manutenção das permissões dos papéis não sofre mudanças. Em geral, o RBAC trabalha com o princípio do mínimo privilégio, um usuário ativa apenas o subconjunto de papéis que precisa para executar uma operação, e esta ativação pode ou não estar sujeita a restrições.

Um modelo unificado denominado RBAC-NIST foi criado para tentar padronizar as várias tendências que têm surgido em modelos baseado a papéis [SANDHU 2000]. Esta família de modelos RBAC-NIST se apresenta estratificada a partir do modelo RBAC básico (*Flat RBAC*), evoluindo para outros modelos da família – RBAC Hierárquico (*Hierarchical RBAC*), RBAC com Restrições (*Constrained RBAC*) e RBAC Simétrico (*Symmetric RBAC*) – adicionando funcionalidades ao mesmo.

4.2.4.3.1. RBAC Básico

Os aspectos essenciais e fundamentais do RBAC estão definidos no RBAC Básico. Como visto, o conceito básico do RBAC é a associação de usuários a papéis, a associação de permissões a papéis e a aquisição de permissões do usuário pelo papel que o mesmo desempenha. Além do conceito básico do RBAC, o modelo RBAC Básico ainda incorpora as seguintes características: (i) exige que as associações usuário-papel e permissão-papel possam ser muitos-para-muitos; (ii) provê suporte à revisão usuário-papel; e (iii) permite a ativação múltipla de papéis.

O RBAC Básico mostrado na Figura 1.4. possui três conjuntos de entidades: usuários (U), papéis (R, de *roles*) e permissões (P). Neste modelo um usuário é uma pessoa ou um processo agindo em nome de uma pessoa. Um papel é uma função ou cargo dentro da organização que possui uma semântica representando a autoridade e a responsabilidade conferidas aos membros desse papel. Uma permissão é um direito específico de acesso a um ou mais objetos do sistema; a natureza exata de uma permissão é dependente da implementação e não é especificada pelo modelo RBAC-NIST [SANDHU 2000].

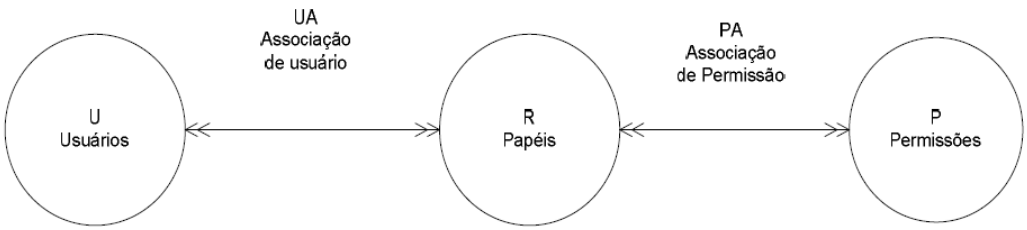


Figura 1.4. RBAC Básico

O RBAC Básico exige que a associação usuário-papel (UA, de *user-role assignment*) e a associação permissão-papel (PA, de *permission-role assignment*) sejam relações do tipo muitos-para-muitos (isto é indicado pelas setas duplas na Figura 1.4). Este é um aspecto essencial do RBAC pois permite que um usuário exerça as permissões de vários papéis. O princípio do mínimo privilégio dificilmente pode ser respeitado quando todos os papéis de um usuário são ativados em uma sessão, mas esse princípio pode ser respeitado mais facilmente permitindo-se o usuário ativar e desativar os papéis que deseja utilizar, pois o usuário pode ativar apenas os papéis necessários à execução de determinada tarefa.

O suporte à revisão usuário-papel possibilita determinar de maneira eficiente quais os usuários associados a um papel e a quais papéis um usuário está associado. A revisão permissão-papel também é muito importante no RBAC para responder quais permissões estão associadas a um papel e a quais papéis uma permissão está associada, mas pela dificuldade de se implementar este mecanismo em sistemas distribuídos de larga escala, ele é requerido somente no RBAC Simétrico [SANDHU 2000].

4.2.4.3.2. RBAC Hierárquico

A inclusão da relação de hierarquia de papéis (RH, de *role hierarchy*), mostrada na Figura 1.5. é a principal diferença do RBAC Hierárquico para o Básico. Uma hierarquia é matematicamente uma ordem parcial definindo uma relação de precedência de papéis, por meio da qual papéis de categoria superior adquirem as permissões dos seus subordinados. Hierarquias de papéis é uma forma natural de estruturar os papéis de modo a refletir as linhas de autoridade e responsabilidade em uma organização.

Existem duas interpretações distintas para uma hierarquia de papéis. Na primeira delas, papéis superiores (*senior roles*) herdam as permissões dos papéis inferiores (*junior roles*); neste caso, tem-se uma hierarquia de herança.

Na outra interpretação para a hierarquia de papéis, a ativação de um papel superior não implica a ativação automática das permissões dos papéis inferiores; neste caso, tem-se uma hierarquia de ativação. Para que as permissões dos papéis inferiores sejam ativadas, estes papéis devem ser explicitamente ativados.

É possível aplicar as duas interpretações simultaneamente. Em tais casos, a hierarquia de ativação pode estender a hierarquia de herança ou ser independente desta [SANDHU 1998]. O modelo RBAC-NIST não define uma interpretação específica para as hierarquias de papéis [SANDHU 2000].

O modelo do NIST divide a RBAC Hierárquico em dois níveis:

- RBAC Hierárquico Geral: neste caso qualquer tipo de ordem parcial pode constituir uma hierarquia de papéis.
- RBAC Hierárquico Limitado: neste caso existem restrições em relação à estrutura da hierarquia de papéis. Geralmente, as hierarquias são limitadas a estruturas simples como árvores ou árvores invertidas.

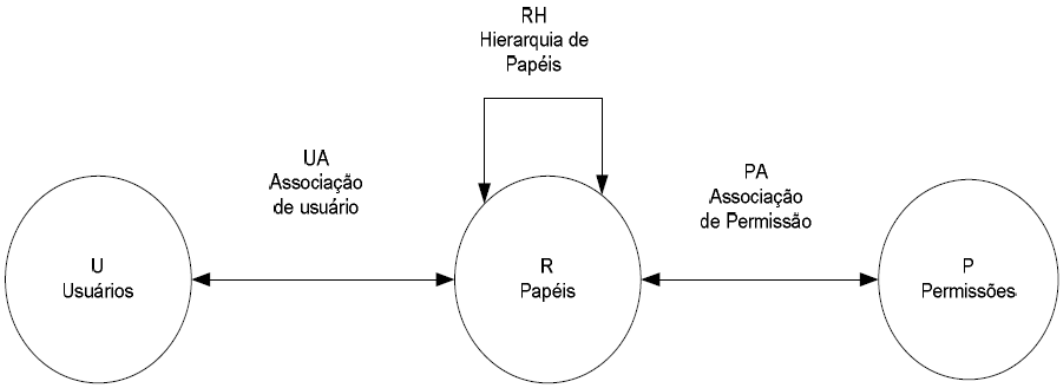


Figura 1.5. RBAC Hierárquico

O mecanismo de revisão usuário-papel do RBAC Básico deve ser estendido para suportar hierarquias de papéis de modo a permitir a identificação tanto dos papéis associados diretamente a um usuário (definidos pelo administrador de segurança) como dos papéis associados indiretamente ao usuário (cuja associação se dá através de herança).

4.2.4.3.3. RBAC com Restrições

Um conceito bastante importante para o modelo de restrições RBAC, no sentido de minimizar a ocorrência de erros e fraudes na manipulação da informação, é a separação de tarefas (*separation of duty*). Este conceito consiste em dividir tarefas que podem gerar conflito de interesses em várias subtarefas menores, executadas por pessoas diferentes, reduzindo, desta maneira, o poder individual de cada usuário. A separação de tarefas teve a sua importância para a segurança da informação reconhecida e discutida em detalhes por Clark e Wilson [CLARK 1987]. O RBAC facilita a implantação de separação de tarefas, utilizando-se, para isso, de relações de exclusão mútua entre papéis. A separação de tarefas é suportada pelo princípio do mínimo privilégio na definição dos papéis, ou seja, os papéis têm que estar associados ao mínimo de permissões necessárias ao cumprimento de suas tarefas.

Existem duas formas de separação de tarefas: estática e dinâmica. Na separação estática de tarefas, dois papéis R1 e R2 que são mutuamente exclusivos não podem ter usuários em comum; em outras palavras, um mesmo usuário não pode ser associado a R1 e a R2. Por outro lado, na separação dinâmica de tarefas uma exclusão mútua entre dois papéis R1 e R2 significa que um usuário pode ser associado a ambos, desde que apenas um deles (R1 ou R2) esteja ativo em um dado momento [SANDHU 2000].

4.2.4.3.3.1. Separação Estática de Tarefas

O conflito de interesses em um sistema baseado em papéis pode surgir como resultado da obtenção de permissões associadas com papéis conflitantes [SANDHU 2000]. Um modo de prevenir esta forma de conflito de interesse é através da separação estática de tarefas (SSD – *Static Separation of Duty*), ou seja, impondo restrições à associação de usuários a papéis (Figura 1.6). Nesta abordagem, usuários associados a um papel não podem ser associados a um segundo papel, de acordo com restrições definidas pelo administrador de segurança.

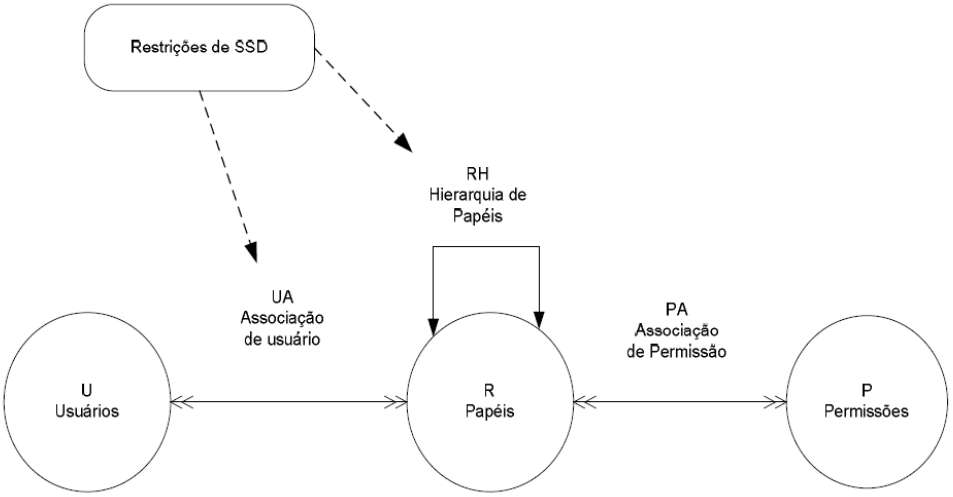


Figura 1.6. RBAC com Restrições – Separação Estática de Tarefas

4.2.4.3.3.2. Separação Dinâmica de Tarefas

O uso de políticas de separação dinâmica de tarefas (DSD – *Dynamic Separation of Duty*) (Figura 1.7) também é permitido no RBAC com Restrições. Nesta abordagem, os usuários podem ser associados a papéis que só constituem um conflito de interesse quando ativados simultaneamente, ou seja, é perfeitamente possível e seguro que um usuário ative mais de um papel do conjunto, desde que estas ativações não sejam simultâneas.

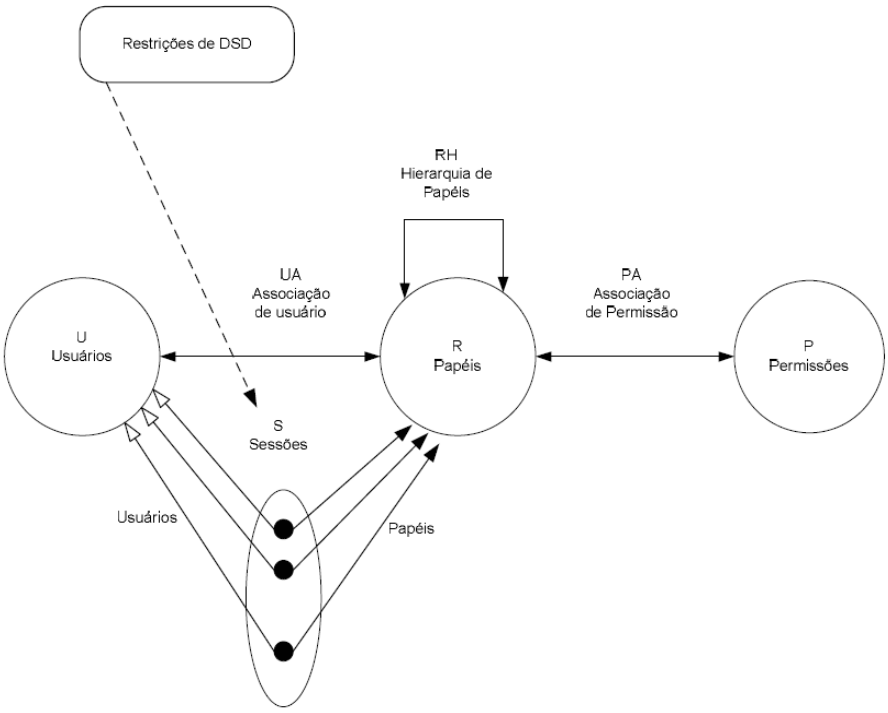


Figura 1.7. RBAC com Restrições – Separação Dinâmcia de Tarefas

A separação dinâmica de tarefas pode ser perfeitamente aplicada a papéis que tenham uma relação de herança, ao contrário do que acontece na separação estática [SANDHU 2000].

4.2.4.3.4. RBAC Simétrico

Um componente essencial para qualquer esquema de gerenciamento de autorização é a manutenção apropriada e precisa das associações permissão-papel. O RBAC Simétrico possui como único requisito o suporte à revisão de associações permissão-papel similar à revisão usuário-papel do RBAC Básico. O desempenho de revisões permissão-papel deve ser comparável ao desempenho de revisões usuário-papel.

A revisão permissão-papel possibilita identificar, de maneira eficiente, as permissões associadas a um papel e, também, quais papéis possuem determinadas permissões. A possibilidade de identificar as associações permissão-papel é um dos principais fatores que diferenciam papéis de grupos.

4.2.4.3.5. Outras características do modelo RBAC-NIST

O modelo unificado RBAC-NIST não abrange todas as características do RBAC. Primeiro, porque não julga apropriado especificar em um modelo detalhes dependentes da aplicação tais como: escalabilidade, permissões negativas, natureza das permissões (granularidade), ativação seletiva de papéis, revogação de papéis. Segundo, porque não existe consenso suficiente da comunidade para justificar a inclusão de algumas características no modelo como, por exemplo, administração e definição de outras restrições além da separação de tarefas.

O modelo RBAC-NIST reconhece apenas restrições de separação de tarefas (tanto estática como dinâmica). O RBAC pode suportar outros tipos de restrições, como pré-condições (determinam as condições que devem ser satisfeitas para que as associações usuário-papel e papel-permissão possam ser efetuadas [SANDHU 1998]), restrições de cardinalidade (determinados papéis podem ter um número máximo de usuários associados [FERRAIOLO 1999]) e obrigações (atributos funcionais que denotam as exigências que um sujeito deve atender antes ou durante um acesso [AHN 1999, SANDHU and PARK 2003]). Embora estes tipos de restrições sejam igualmente importantes, as restrições de cardinalidade e pré-condições não foram incluídas no modelo unificado RBAC-NIST por não existir ainda consenso sobre a sua real utilização.

4.2.4.4. RBAC com contextos

O objetivo do modelo RBAC com contextos, uma extensão do RBAC, é se adequar a um tipo de sistema utilizado por várias divisões dentro da mesma organização, tornando a administração de autorizações e papéis mais simples e mais intuitiva para o usuário. Por outro lado, não se deseja perder em funcionalidade, de forma a tornar o sistema de controle de acesso inadequado em relação aos requisitos de segurança.

O primeiro passo para tornar o sistema de controle de acesso adaptado ao tipo de aplicação foi a introdução de uma nova entidade além das quatro já existentes no RBAC. Essa nova entidade reflete as divisões existentes dentro da organização, para as quais se deseja a separação da atribuição de papéis, pois essa atribuição somente será válida dentro de um contexto. Por este motivo, a esta nova entidade foi dado o nome de Contexto (C), e na Figura 1.8. pode-se perceber como ela se relaciona com as outras entidades. A interpretação do contexto também vai depender da aplicação, mas algumas das interpretações feitas em sistemas já implementados são as seguintes: filiais, unidades e departamentos. Dessa forma, assim como foi feito com as outras entidades do RBAC, a interpretação para os contextos será deixada em aberto. Nesse novo modelo, uma sessão é composta apenas por um usuário e um contexto por vez. Para que um usuário ative um novo contexto em uma sessão, é necessário que ele desative o contexto anterior. Dessa forma, o que existe é uma troca de contextos. Ou seja, uma sessão em um determinado momento é formada por apenas um usuário e um contexto.

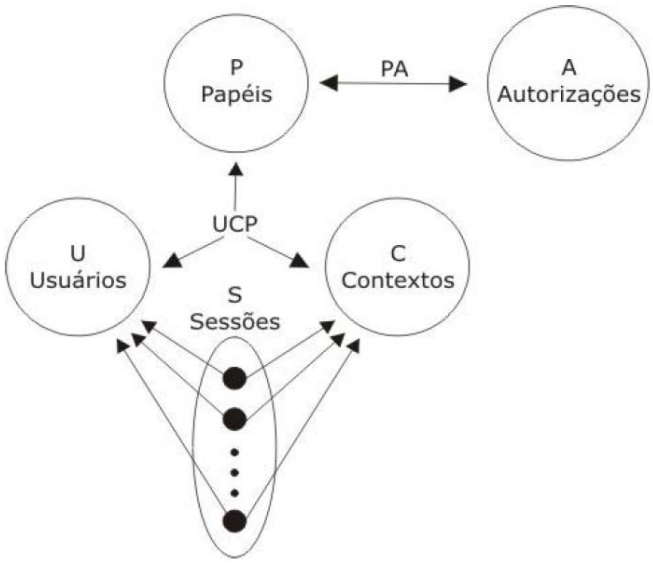


Figura 1.8. Modelo RBAC com Contextos

Com a utilização dos contextos é possível realizar facilmente a implementação da autonomia desejada para as divisões da organização no que diz respeito à administração de papéis. Para os usuários, o acúmulo de papéis fica mais claro, pois os papéis são atribuídos por contexto, algo que acaba sendo mais intuitivo para o mesmo: “Eu possuo o papel de Administrador na Filial A e o papel de Supervisor na Filial B”. Pode-se dizer que o grande ganho do modelo RBAC com contextos, do ponto de vista do usuário, é realizar uma simplificação que atende os requisitos de segurança, ao mesmo tempo em que facilita a compreensão do sistema. O impacto da facilidade de implementação se reflete em menor tempo de desenvolvimento, menor custo e, conseqüentemente, em melhor nível de manutenibilidade. Com os contextos, as autorizações também ficam mais fáceis de serem definidas. As autorizações são o resultado do relacionamento entre os objetos protegidos e as formas de acesso ao mesmo. Os contextos ajudam na definição de quais são os objetos para os quais uma determinada autorização está sendo concedida. Se, por exemplo, o objeto é o acesso às entradas e saídas na conta, o contexto ajuda na divisão dessas entradas e saídas, de forma que a autorização correspondente fica definida somente para as entradas e saídas relacionadas com aquele contexto, ou seja, daquela filial ou unidade da organização. Isto ajuda na divisão dos objetos não só pelo seu tipo, como no exemplo de entradas e saídas na conta, mas também pelo contexto ao qual está relacionado.

O modelo RBAC com contextos não foi a primeira abordagem na literatura para controle de acesso dependente de contexto. O modelo de autorização contextual para controle de acesso baseado em papéis proposto por Motta e Furuie [MOTTA, FURUIE 2002a] faz com que uma autorização seja positiva ou negativa, de acordo com regras que relacionam informações sobre o contexto em que aquela autorização está sendo solicitada. Esse modelo vem sendo utilizado no MACA, uma ferramenta de autorização e controle de acesso para o prontuário eletrônico de pacientes [MOTTA, FURUIE, 2002b], que se encontra funcionando no Instituto do Coração do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo.

A diferença entre o modelo de autorizações contextuais e o RBAC com contextos está na forma com que o contexto se encaixa no modelo. No sistema de autorizações contextuais o contexto é algo dinâmico e extremamente granular, onde o mesmo é constituído de informações tais como hora, local e qual o paciente. Nesse modelo, o contexto se encaixa na autorização, que será positiva ou negativa, dependendo de regras que utilizam variáveis do contexto.

Já o modelo RBAC com contextos trabalha com contextos fixos e mais amplos, tais como filiais e unidades de uma organização. No modelo proposto, o contexto se encaixa no relacionamento entre usuários e papéis, funcionando como um divisor entre o controle de acesso dos diferentes contextos. Como o contexto é algo bem mais amplo, faz sentido atribuir papéis dentro de cada contexto.

Apesar do objetivo de ambos os modelos ser o de diferenciar autorizações de acesso do mesmo tipo de informação de acordo com o contexto, os ambientes de utilização que tornam os modelos adequados para utilização são diferentes. O modelo de autorizações contextuais trabalha com contextos dinâmicos e granulares, bem como com uma definição dinâmica da autorização, enquanto o modelo RBAC com contextos trabalha com contextos fixos e amplos, bem como com a atribuição de papéis por contexto.

4.2.4.5. Considerações sobre os Modelos

Basicamente os modelos de segurança foram definidos em períodos distintos com tendências e características específicas. No começo dos anos 60, o desenvolvimento desses modelos foi estimulado pelo crescente surgimento dos sistemas de tempo compartilhado. Depois, nos anos 70, seu desenvolvimento foi impulsionado por fins e propósitos militares, e nos anos 80 caracterizou-se pelo enfoque mais comercial. Dos anos 90 em diante, a tônica está sendo em modelos que envolvem os ambientes distribuídos, sem um propósito específico, mas geralmente para suportar políticas de segurança múltiplas. A diversidade de origem e objetivos dos modelos de segurança não permite que eles sejam diretamente comparados entre si. Entretanto, é possível abstrair-se de detalhes específicos de cada modelo e concentrar-se nas suas características gerais, estabelecendo uma base comum a partir da qual é possível tecer uma comparação. A Tabela 1.5. ilustra as principais diferenças entre os modelos de controle de acesso discutidos neste capítulo.

O modelo matriz de acesso é caracterizado pela sua flexibilidade, o que facilita a gerência descentralizada. Como neste modelo são os próprios sujeitos que determinam quais os acessos que outros sujeitos possuem sobre seus objetos, diz-se que a administração da política de segurança do modelo matriz de acesso é descentralizada. Ao contrário, os modelos obrigatórios exigem a presença no sistema de um administrador de segurança, que é assumido como único.

O preço da flexibilidade e da descentralização é a complexidade envolvida no controle à propagação de direitos no sistema. Os modelos obrigatórios geralmente definem um conjunto de regras não contornáveis no sistema, o que diminui as possibilidades de propagação de direitos. Estes modelos são próprios para gerências centralizadas de segurança, representando estruturas pouco flexíveis. Os modelos baseados em papéis são tidos como modelos intermediários entre os discricionários e os obrigatórios. Tais modelos apresentam a estrutura flexível dos primeiros e a

possibilidade de gerência centralizada característica dos obrigatórios. Todos os modelos descritos neste texto podem ser implementados em sistemas distribuídos, com maior ou menor dificuldade [WESTPHALL 2000].

A separação de tarefas é um conceito suportado somente pelo modelo baseado em papéis. O princípio do mínimo privilégio, por sua vez, é suportado tanto pelos modelos baseados em papéis quanto pelos modelos BLP e Biba, através do uso criterioso dos rótulos correntes de segurança. É importante observar, porém, que a granularidade do suporte a mínimo privilégio nos modelos baseados em papéis é bem mais fina do que nos modelos baseados em rótulos. Os modelos baseados em papéis são os únicos que permitem incorporar, de alguma forma, a hierarquia natural das organizações ao controle de acesso; nenhum dos outros modelos fornece esta importante facilidade ao administrador de segurança. O último critério considerado na comparação da tabela é a facilidade ou viabilidade de implementação.

O modelo matriz de acesso é o mais simples de ser implementado dentre todos. Modelos baseados em papéis vêm em segundo lugar em termos de simplicidade, não trazendo grandes complicações à implementação. Os modelos baseados em rótulos já foram implementados em vários sistemas, mas requerem um razoável esforço para identificar todas as estruturas de um sistema que precisam ser rotuladas para evitar as violações de confiabilidade (BLP) ou integridade (Biba).

Tabela 1.5. Comparação entre os modelos de segurança

Características	Modelo			
	Matriz de Acesso	BLP	Biba	RBAC
Flexibilidade	Sim	Não	Não	Sim
Política de Controle de Acesso Centralizada	Não	Sim	Sim	Sim
Administração da Política Centralizada	Não	Sim	Sim	Sim ou Não
Separação de tarefas	Não	Não	Não	Sim
Suporte a mínimo Privilégio	Não	Sim	Sim	Sim
Suporte à Hierarquia Organizacional	Não	Não	Não	Sim
Facilidade de Implementação	Fácil	Médio	Médio	Fácil

4.2.5. Mecanismos de Segurança

Os mecanismos de segurança são responsáveis pela implementação das políticas de segurança específicas, expressas pelos modelos de segurança. Como exemplo, podemos dizer que uma política de segurança pode exigir que todos os usuários de um sistema sejam identificados univocamente para fins de contabilidade, e os mecanismos para implantar esta política incluem o uso de senhas, de cartões magnéticos e de dispositivos de reconhecimento de impressões digitais. Para viabilizar a implantação de tais políticas, os mecanismos são construídos a partir de controles de acesso e controles criptográficos.

No que se refere a controle de acesso, podemos dizer que virtualmente todos os sistemas computacionais podem ser descritos em termos de sujeitos acessando objetos. O controle de acesso é, portanto, a mediação das requisições de acesso a objetos iniciadas pelos sujeitos. Um monitor de referência é um modelo conceitual do subsistema responsável pelo controle de acesso. Ou seja, é a entidade que recebe todas

as requisições de acesso dos sujeitos e autoriza ou nega o acesso de acordo com a política de segurança implantada.

O monitor de referência, tendo como função intermediar todas as requisições de acesso aos objetos de um sistema, deve ter algumas propriedades, tais como: deve ser inviolável, incontornável (sempre invocado) e pequeno o suficiente para permitir a verificação de sua correção. A noção de núcleo de segurança foi definida em [LANDWEHR 1983] como o conjunto de recursos de hardware e software que permitem a realização de um monitor de referências.

A implementação do monitor de referência é realizada através de mecanismos de controle, que podem ser discricionários (DAC – *Discretionary Access Control*), obrigatórios (MAC – *Mandatory Access Control*) ou baseados em papéis (*RBAC - Role-Based Access* Control).

4.2.5.1. Controles de Acesso Discricionário

Os mecanismos de controle de acesso discricionário (DAC) implementam políticas discricionárias, permitido ao usuário atribuir direitos de acesso sobre seus recursos computacionais de acordo com a sua necessidade. Tais mecanismos baseiam-se na idéia de que o proprietário da informação deve determinar quem terá acesso a essa informação. O controle discricionário permite que os dados sejam livremente copiados de objeto para objeto, de modo que, mesmo que o acesso aos dados originais seja negado, pode-se obter acesso a uma cópia. Porém, se o usuário não atribuir corretamente estes direitos, ou mesmo se o fizer permitindo acesso de cópia a outros sujeitos, a disseminação de suas informações no sistema não pode ser controlada. O controle de acesso discricionário não impõe nenhuma restrição à disseminação de direitos e à própria evolução da matriz de acesso. Apesar dessa limintação, na prática, devido talvez à facilidade de implementação, o controle de acesso discricionário é largamente utilizado nos sistemas atuais.

4.2.5.2. Controle de Acesso Obrigatório

Os mecanismos de controle de acesso obrigatório (MAC) implementam políticas obrigatórias, as regras de controle de acesso são impostas por uma autoridade central. Baseiam-se em uma administração centralizada de segurança, a qual dita regras incontornáveis de acesso à informação. Estes mecanismos, como descrito anteriormente, implementam políticas multinível. Dos vários relatos de implementação de mecanismos MAC, observa-se que os mesmos são bem mais difíceis de viabilizar que os de DAC, devido à rigidez de suas regras e às limitações de seus modelos, além de outras dificuldades de caráter computacional [LANDWEHR 1984] que geram limitações à implementação deste modelo. Deste modo os modelos conceituais necessitam de relaxamento de suas regras para poderem ser viabilizados em mecanismos comerciais.

4.2.5.3. Controle de Acesso Baseado em Papéis

Para os mecanismos que implementam RBAC a identidade no sistema é o papel, uma vez que estes encapsulam as políticas na forma de permissões. Tais mecanismos têm como base que os direitos de acesso sejam atribuídos a papéis e não a usuários, assim como acontece no DAC, já que os usuários obtêm estes direitos em virtude de terem papéis a si atribuídos. Por ser independente das políticas, o RBAC é facilmente

ajustável a mudanças no ambiente e é largamente utilizado, porque não é tão flexível quando o DAC e nem tão rígido quanto o MAC. Por mais que os modelos RBAC ainda estejam em desenvolvimento, existem vários relatos de implementações do mesmo, um deste pode ser encontrado em [GLENN 1999].

4.2.5.4. Controles Adicionais de Segurança

Os mecanismos de segurança têm como seu principal objetivo implantar políticas de segurança, utilizando para isso ações, técnicas ou dispositivos. Mas indo um pouco além, existem ainda outros controles (internos) que não atuam diretamente nas requisições de acesso, mas que devem necessariamente estar presentes nos sistemas. Dentre eles podemos citar:

1. A auditoria de vestígio, ligada à geração periódica de registros de eventos associados à segurança, coletados para uso potencial em detecção de intrusão e/ou auditoria de segurança.
2. A auditoria de segurança, inspeção independente (por terceiros) dos procedimentos e registros do sistema com intuito de verificar a adequação da política de segurança e as possíveis violações do sistema.
3. A detecção de intrusão, que usa os mesmos registros das auditorias em métodos automatizados de análise em tempo real, envolvendo muitas vezes uma seqüência de eventos relacionados ou não, com o intuito de identificar atividades anormais no sistema.

Mecanismos que façam uso de técnicas de *backups*, replicações e que permitam recuperar o sistema em situações onde as violações não puderam ser evitadas completam os controles adicionais.

Outros controles (externos) necessitam ser adicionados aos internos já comentados. Por exemplo, é necessário que se leve ao conhecimento de cada usuário suas atribuições e responsabilidades, para que esse saiba o que está autorizado a fazer. Se este não estiver treinado e convencido da importância da segurança no ambiente computacional as demais medidas podem se tornar sem eficácia [AMOROSO 94].

4.3. Modelagem de Sistemas Seguros

Aplicações de comércio eletrônico e outras similares no âmbito da internet possuem a segurança como requisito fundamental e apesar da grande variedade de arcabouços de segurança disponíveis, constantemente há noticias sobre vulnerabilidades e falhas de segurança em sistemas de software [FINK et al 2004]. Á medida que os arcabouços aparecem, eles se tornam rapidamente ultrapassados por diferentes motivos. Na maioria das vezes porque eles não oferecem um modelo arquitetural suficientemente flexível para acompanhar as necessidades do negócio, o qual está em desenvolvimento ou porque eles limitam o escopo da arquitetura de que o negócio necessita. Devido à diversidade de requisitos de negócio de cada aplicação, arquitetos de software e analistas necessitam cada vez mais criar seus próprios arcabouços e manter suas integridades e coerência em relação às necessidades do negócio as quais eles atendem.

Além disso, existe uma grande lacuna entre as soluções teóricas e o que é de fato implementado na área de segurança. Um dos problemas que contribuem para a construção de software com segurança fraca é o fato de este requisito ser raramente

considerado nos estágios iniciais do desenvolvimento de *software* e ser relegado a segundo plano ao longo do mesmo, provocando problemas de segurança tanto na arquitetura da aplicação quanto na lógica implementada.

De acordo com os problemas relatados, são necessárias novas formas de desenvolver software seguro, baseadas não somente na aplicação das teorias existentes como na adoção de um processo de desenvolvimento que considere os requisitos de segurança como parte integral do projeto de construção de *software* [FERNANDEZ 2000]. Além disso, o aumento da complexidade dos sistemas aliado à dinamicidade inerente às regras de negócio das aplicações atuais faz com que o processo de desenvolvimento de segurança necessite cada vez mais de arcabouços flexíveis e maior gerência de requisitos e mudanças.

Uma das abordagens que pode auxiliar arquitetos e projetistas a construir sistemas seguros e integrar soluções de segurança nas fases de análise e projeto de sistemas de software consiste no uso de padrões de projetos.

Contudo, para obter os benefícios da utilização de padrões de projeto, é necessário saber quando e como utilizá-los a partir de suas definições. Neste contexto, uma abordagem orientada a modelo pode auxiliar o processo de desenvolvimento e de criação de arcabouços de segurança, promovendo uma implementação controlada dos padrões de projeto e desta forma, garantindo aderência às diretrizes arquiteturais da organização e às questões de segurança.

Além disso, uma abordagem orientada a modelo como MDA (*Model Driven Architecture*) permite que segurança esteja integrada na modelagem de sistemas à medida que modelos de projeto são combinados com modelos de segurança e técnicas de geração automática de código são utilizadas para automatizar a construção de sistemas a partir desses modelos. Isto auxilia a criação de arcabouços de segurança flexíveis e garante que os requisitos de segurança serão levados em consideração durante todas as fases do processo de desenvolvimento do sistema.

4.3.1. Padrões de Projeto de Segurança

Padrões de projeto foram introduzidos como uma forma de identificar e apresentar soluções a problemas recorrentes na programação orientada a objetos. Joseph Yoder e Jeffrey Barcalow [YODER, BARCALOW 1997] foram uns dos primeiros a utilizar esta abordagem como uma tentativa de criar arcabouços de segurança bem projetados. O uso de padrões nesse contexto justifica-se, pois enquanto muitos problemas de segurança são novos ou complicados, uma grande parte de outros problemas são bem conhecidos e possuem soluções bem estabelecidas. No espírito do provérbio “É melhor ensinar a pescar do que dar o peixe”, é melhor explicar como utilizar padrões de projeto para criar arquiteturas de segurança próprias a utilizar arquiteturas prontas que não atendem as necessidades do negócio e que muitas vezes o arquiteto e o projetista têm que modificar para fazer o melhor acoplamento possível.

O uso de padrões é uma boa ferramenta para ajudar arquitetos e projetistas a construir sistemas seguros, pois enquanto muitos problemas de segurança são novos ou complicados, uma grande parte de outros problemas são bem conhecidos e possuem soluções bem estabelecidas.

4.3.1.1. Padrões de Projeto

Os padrões de projeto de software, também muito conhecidos pelo termo original, *Design Patterns*, descrevem soluções para problemas recorrentes no desenvolvimento de sistemas de *software* orientados a objetos. Essas soluções são desenvolvidas e conhecidas por especialistas, tornando-se padrões por serem reutilizadas várias vezes em vários projetos, além de terem eficácia comprovada. Os padrões de projeto visam facilitar a reutilização de soluções de *design* - isto é, soluções na fase de projeto do software, sem considerar reutilização de código.

O conceito de padrão de projeto foi criado na década de 70 pelo arquiteto Christopher Alexander. Em seus livros *Notes on the Synthesis of Form*, *The Timeless Way of Building* [Alexander 1979] e *A Pattern Language* [Alexander 1978], ele estabelece que um padrão deve possuir, idealmente, as seguintes características:

- **Encapsulamento**: um padrão encapsula um problema/solução bem definido. Ele deve ser independente, específico e formulado de maneira a ficar claro onde ele se aplica.
- **Generalidade**: todo padrão deve permitir a construção de outras realizações a partir dele.
- **Equilíbrio**: quando um padrão é utilizado em uma aplicação, o equilíbrio estabelece a razão, relacionada com cada uma das restrições envolvidas, para cada passo do projeto. A forma de encontrar este equilíbrio pode ser uma análise racional que envolva uma abstração de dados empíricos, uma observação da aplicação de padrões em artefatos tradicionais, uma série convincente de exemplos e uma análise de soluções ruins ou fracassadas.
- **Abstração**: os padrões representam abstrações da experiência empírica ou do conhecimento cotidiano.
- **Abertura**: um padrão deve permitir a sua extensão para níveis mais baixos de detalhamento.
- **Combinatoriedade**: os padrões são relacionados hierarquicamente. Padrões de alto nível podem ser compostos ou relacionados com padrões que endereçam problemas de nível mais baixo.

Além da definição das características de um padrão, Alexander definiu o formato que a descrição de um padrão deve ter. Ele estabeleceu que um padrão deve ser descrito em cinco partes:

- **Nome**: Descrição da solução, mais do que do problema ou do contexto.
- **Exemplo**: Uma ou mais figuras, diagramas ou descrições que ilustrem um protótipo de aplicação.
- **Contexto**: Descrição das situações sob as quais o padrão se aplica.
- **Problema**: Descrição das forças e restrições envolvidas e como elas interagem.
- **Solução**: Relacionamentos estáticos e regras dinâmicas descrevendo como construir artefatos de acordo com o padrão, freqüentemente citando variações e formas de ajustar a solução segundo as circunstâncias. Inclui referências a outras

soluções e o relacionamento com outros padrões de nível mais baixo ou mais alto.

Partindo do trabalho de Alexander, profissionais de software começaram a incorporar esses princípios na criação das primeiras documentações de padrões de projetos como um guia para desenvolvedores iniciantes. O resultado prático foi a publicação de *Design Patterns: Elements of Reusable Object-Oriented Software* [Gamma et al 1995]. Os autores desse livro são Eric Gamma, Richard Helm, Ralph Johnson e John Vlissides, conhecidos como a "Gangue dos Quatro" (*Gang of Four*) ou simplesmente "GoF". Este livro é a referência principal no assunto para a comunidade de software. Nele são descritos 23 padrões que foram baseados na experiência dos autores.

Posteriormente, vários outros livros tratando do tema foram publicados, como *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* [LARMAN 2004], que introduziu um conjunto de padrões conhecidos como GRASP (*General Responsibility Assignment Software Patterns*).

Dentre as principais vantagens da utilização de padrões de projetos estão:

- Capturar o conhecimento e a experiência de especialistas em projeto de software.
- Especificar abstrações que estão acima do nível de classes ou objetos isolados ou de componentes.
- Definir um vocabulário comum para a discussão de problemas e soluções de projeto.
- Facilitar a documentação e manutenção da arquitetura do software.
- Auxiliar o projeto de arquiteturas mais complexas.
- Fornecer soluções que já foram testadas e aprovadas.
- Tornar o sistema mais fácil de entender e manter

4.3.1.1.1. Representação de padrões de projeto

A representação de um padrão de projeto deve conter no mínimo o nome (identificação), o problema (quando o uso do padrão pode ser interessante), a solução (como aplicar o padrão de projeto) e as conseqüências (perdas e ganhos ao se aplicar o padrão). Não há um formato único e padronizado para representar padrões de projeto. Com isso, diferentes formatos têm sido utilizados por diferentes autores. Porém, alguns formatos se tornaram mais conhecidos que outros e, consequentemente, se tornaram padrões na documentação de novos padrões. Um exemplo de formato comumente utilizado é o da gangue dos quatro, descrito em [Gamma et al 1995], que contém:

- **Nome:** Um nome descritivo e único que ajuda a identificar e referenciar um padrão
- **Intenção:** Uma descrição do objetivo do padrão e a razão para utilizá-lo
- **Também conhecido como:** Outros nomes para o padrão

- **Motivação:** Um cenário contendo um problema e um contexto onde esse padrão possa ser utilizado
- **Aplicabilidade:** Situações onde este padrão possa ser utilizado, o contexto do padrão
- **Estrutura:** Uma representação gráfica do padrão. Diagramas de classe e diagramas de interação podem ser utilizados para este propósito.
- **Participantes:** Uma listagem das classes e objetos utilizados no padrão e seus papéis no projeto.
- **Colaboração:** Uma descrição de como classes e objetos utilizados no padrão interagem entre si.
- **Conseqüências:** Uma descrição dos resultados, efeitos causados pela utilização do padrão.
- **Implementação:** Uma descrição de uma implementação do padrão.
- **Código Exemplo:** Uma ilustração de como o padrão pode ser utilizado em uma linguagem de programação
- **Usos conhecidos:** Exemplos de utilizações reais do padrão

4.3.1.2. Metodologia de utilização de padrões de projeto

Em [Gamma et al 1995] há um passo a passo para utilizar um padrão de projeto efetivamente:

1. Dedique atenção particular na aplicabilidade e nas conseqüências do padrão para ter certeza de estar utilizando o padrão correto para o seu problema.
2. Entenda perfeitamente as classes e os objetos no padrão e como eles se relacionam entre si.
3. Estude o código como forma de entender como implementar o padrão
4. Os nomes dos participantes nos padrões de projeto são geralmente muito abstratos para aparecer diretamente na aplicação. Contudo, deve-se incorporar o nome do participante no nome que aparece na aplicação. Isto faz com que o padrão se torne mais explicito na implementação. Resumindo, escolha nomes para os participantes que sejam significativos no contexto da aplicação.
5. Defina as classes, declare as interfaces, estabeleça as heranças, os relacionamentos e defina as variáveis de instancia. Identifique classes existentes na aplicação que o padrão irá afetar e modifique-as de forma correta.
6. Defina nomes para as operações de forma específica à aplicação. Utilize as responsabilidades e colaborações associadas a cada operação como um guia.
7. Implemente as operações garantindo as responsabilidades e colaborações do padrão

Estes passos são apenas um guia pra começar. Com o tempo, cada pessoa desenvolve sua própria maneira de trabalhar com padrões de projeto.

A maioria das pessoas utiliza padrões de projeto quando percebe um problema com seu projeto - alguma coisa que deveria ser fácil, mas não é - ou com sua implementação - como desempenho. Ao examinar um código, é necessário verificar quais são seus problemas e quais são seus comprometimentos, o que gostaria de realizar que agora esta sendo difícil. A partir daí, é necessário procurar uma referência de padrão de que corresponda aos problemas que se deseja resolver.

Padrões de projeto podem parecer abstratos à primeira vista, porém eles se tornam mais concretos à medida que são utilizados. Uma vez que o vocabulário dos padrões de projeto se torna conhecido, a interação se torna mais precisa e rápida com outras pessoas que utilizam este vocabulário. Por exemplo, é melhor dizer "esta é uma instância do padrão Visitor" do que "este é um código que varre a estrutura e realiza chamadas de retorno, sendo que alguns métodos devem estar presentes, para serem chamados em uma ordem particular e de uma determinada forma".

Padrões de projeto podem aumentar ou diminuir a capacidade de compreensão de um projeto ou de uma implementação. Eles podem diminuir a capacidade de compreensão ao complicar o projeto e/ou diminuir o desempenho enquanto que podem aumentar a capacidade de compreensão ao melhorar a modularidade, separando melhor os conceitos, e simplificando a descrição.

É preciso saber quando e como utilizar padrões de projeto de forma que um padrão de projeto só seja utilizado quando a flexibilidade e os benefícios que ele oferece forem realmente necessários.

4.3.1.3. Padrões de Projeto de Segurança

Como dito anteriormente, atualmente, existe uma grande lacuna entre as soluções teóricas e o que é de fato implementado na área de segurança. Neste contexto, a abordagem de padrões de projeto pode e deve ser aplicada à segurança como forma de preencher essa lacuna promovendo várias vantagens, entre elas:

- Novatos podem atuar como especialistas;
- Especialistas em segurança podem identificar, nomear e discutir problemas e soluções de modo mais eficiente;
- Problemas podem ser resolvidos de uma forma estruturada;
- Dependências de componentes podem ser identificadas e consideradas de forma apropriada;

Padrões de segurança devem ser utilizados quando há um problema específico em um contexto específico e quando se deseja desenvolver uma arquitetura para resolver este problema.

Yoder e Barcalow [YODER, BARCALOW 1997] foram uns dos primeiros a adaptar a abordagem de padrões de projeto a segurança da informação onde apresentaram os seguintes padrões:

- **Ponto de Acesso único**: prover um módulo de segurança e uma forma de autenticação;
- **Ponto de verificação**: organizar pontos de verificação de segurança e suas repercussões;

- **Papéis**: organizar usuários com privilégios de segurança similares;
- **Sessão**: localizar informação global em um ambiente multi-usuário;
- **Visão completa com erros**: prover uma visão completa aos usuários, exibindo exceções quando necessário;
- **Visão limitada**: permitir que os usuários visualizem somente aquilo a que eles têm acesso;

Atualmente, há vários trabalhos relacionados ao assunto de padrões de projeto de segurança. Dentre eles, podemos citar um guia técnico de padrões de segurança [Blakley et al 2004] publicado pelo Opengroup [OpenGroup 2007] que contém a definição de alguns padrões de segurança que foram separados em duas categorias.

A primeira categoria contém padrões de segurança que facilitam a construção de sistemas que estão sempre disponíveis, ou seja, que provêm acesso ininterrupto aos serviços e recursos que eles oferecem aos usuários. Como exemplo, podemos citar os padrões Sistema com Ponto de Restauração (*Checkpointed System***),** que visam estruturar um sistema de modo que seu estado possa ser recuperado e restaurado a um estado válido caso haja falha em algum componente. Outro exemplo, é o padrão Sistema Tolerante a falha (*Comparator-Checked Fault-Tolerant System***)** que visa estruturar um sistema de forma que uma falha independente em um componente seja detectada rapidamente e que não cause falha no sistema inteiro.

A segunda categoria contém padrões de segurança que facilitam a construção de sistemas que protegem recursos contra uso não autorizado, divulgação ou modificação. Como exemplo, podemos citar os padrões Sistema Protegido (*Protected System*), que visa estruturar um sistema de modo que todo acesso feito aos recursos seja mediado por um guardião que reforce uma política da segurança. Outro exemplo é o Proxy Seguro (*Secure Proxy*) que define o relacionamento entre dois guardiões de duas instâncias de Sistema Protegido no caso onde uma instância está totalmente contida dentro da outra.

Outro trabalho na área foi feito por [Fernandez and Pan 2001], onde são discutidos três padrões que correspondem a modelos de segurança: Autorização, Controle de Acesso Baseado em Papéis e Segurança Multi-nível. O padrão Autorização representa o modelo de matriz de acesso e visa descrever autorizações por entidades computacionais ativas (sujeitos) a recursos passivos (objetos). O padrão Controle de Acesso Baseado em Papéis visa descrever a atribuição de permissões a usuários de acordo com seus papéis em uma instituição. Já o padrão Segurança Multi-nível visa ajudar na decisão de acesso em um ambiente com classificações de segurança, ou seja, ambientes onde informações e documentos possuem níveis de segurança, como por exemplo, secreto, confidencial, etc.

Darrel M. Kienzle and Matthew C. Elder construíram um repositório de padrões de segurança com vinte e seis padrões e três mini-padroes. O foco desses padrões está na aplicação de segurança na web. Os padrões estão disponíveis em [KIENZLE, ELDER 2002]. O relatório final do projeto contendo resumo de todos os padrões esta disponível em [KIENZLE et al 2002].

Vários livros foram publicados, entre eles, podemos citar [Schumacher 2003], [Schumacher et al 2005] e [Steel et al 2005]. Este último foi escrito por um grupo da SUN [SUN 2007] e oferece um conjunto de padrões de segurança para aplicações J2EE,

serviços Web e gerência de identidade. Já em [Schumacher et al 2005], encontra-se a descrição de vários padrões de segurança separados por tipos, entre esses tipos, podemos citar: Padrões de identificação e autorização, Padrões de modelo de controle de acesso e Padrões de projeto de segurança para aplicações internet.

4.3.2. Segurança com MDA

MDA (*Model Driven Architecture*) é uma abordagem de desenvolvimento de sistemas que se baseia na idéia da separação da especificação das funcionalidades de um sistema dos detalhes de sua implementação [OMG 2003]. Os três objetivos principais da abordagem MDA são portabilidade, interoperabilidade e reutilização através da separação arquitetural de interesses. Neste sentido, MDA define mecanismos para: (i) especificar um sistema independentemente da plataforma que o suporta; (ii) especificar plataformas; (iii) escolher uma plataforma para um sistema; e (iv) transformar a especificação do sistema em código uma plataforma específica. Com a separação de interesses, esta abordagem ajuda a focar nos aspectos de negócio da solução ao invés dos aspectos técnicos.

A idéia de "orientação a modelos" está fundamentada na utilização de modelos para direcionar o entendimento, projeto, construção, instalação e manutenção de sistemas. Apesar de esta idéia ser antiga, a MDA tem se mostrado bastante promissora devido ao diferencial em que os modelos são transformados em código de forma automática e configurável. Desta forma, uma possível mudança de plataforma implica alteração da especificação das transformações, enquanto que a especificação da solução se mantém intacta.

A utilização de segurança com MDA permite que segurança esteja integrada na modelagem de sistemas à medida que modelos de projeto são combinados com modelos de segurança e técnicas de geração automática de código são utilizadas para automatizar a construção de sistemas a partir desses modelos auxiliando a criação de arcabouços de segurança flexíveis e garantindo que os requisitos de segurança sejam levados em consideração durante todas as fases do processo de desenvolvimento do sistema. Com isso, as falhas de segurança podem ser identificadas mais rapidamente no processo de desenvolvimento e a implementação é mantida consistente com a política de segurança modelada, além de poder ser migrada para novas plataformas.

4.3.2.1. Model Driven Architecture

O objetivo da engenharia de software é oferecer uma estrutura de processos, métodos e ferramentas para o desenvolvimento de software. Melhorar a produtividade, a manutenibilidade, a integração e a qualidade são alguns dos objetivos a serem alcançados em um ambiente de desenvolvimento de software. Desde o surgimento da engenharia de software, muitos métodos, processos e ferramentas foram propostos para alcançar esses objetivos. A MDA (*Model Driven Archictecture*) é um exemplo. Ela é uma abordagem de desenvolvimento de software definido pela OMG (*Object Management Group*) que se baseia na idéia da separação da especificação das funcionalidades de um sistema dos detalhes de implementação.

Os três objetivos principais da MDA são prover portabilidade, interoperabilidade e reutilização através da separação arquitetural de interesses. Neste sentido, MDA define mecanismos para especificar um sistema independentemente da plataforma que o

suporta, especificar plataformas, escolher uma plataforma para um sistema e transformar a especificação do sistema em uma plataforma específica. Com a separação de interesses, esta abordagem ajuda a focar nos aspectos de negócio da solução ao invés dos aspectos técnicos.

Atualmente, a UML (*Unified Modelling Language*) [UML 2007] é a notação mais utilizada na modelagem de sistemas orientados a objetos. A UML é uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos de sistemas orientados a objetos. Ela é basicamente composta de elementos e diagramas. Os elementos compõem o modelo do sistema e os diagramas são uma forma de visualizar apenas um subconjunto destes elementos. Através dos diagramas, podemos ter diferentes visões de um mesmo modelo.

A abordagem MDA recomenda a criação de modelos em três níveis de abstração na especificação de sistemas:

- Modelo Independente de Computação (*Computation Independent Model* - CIM): é a representação do sistema do ponto de vista independente de computação. Esse modelo foca nos requisitos do sistema, onde os detalhes de estrutura e processamento estão ocultos ou ainda indeterminados. Esse modelo é também conhecido como modelo de domínio. Serve, principalmente, como meio de comunicação entre os especialistas de domínio e requisitos e os especialistas no projeto e construção do sistema.
- Modelo Independente de Plataforma (*Platform Independent Model*) (PIM): é um modelo de alto nível de abstração que representa as funcionalidades de um sistema, com foco nas regras de negócio, utilizando-se de uma especificação que não dependa de plataforma. O PIM é transformado em um ou mais modelos PSM.
- Modelo Específico de Plataforma (*Platform Specific Model*) (PSM): é a complementação do Modelo Independente de Plataforma com detalhes tecnológicos específicos da plataforma de implementação do sistema.

As transformações de modelos consistem em converter um modelo em outro através de mapeamentos. A Figura 1.9. exemplifica essas transformações e os conceitos centrais da MDA. Um PIM é transformado em um ou mais PSM através de regras de transformação e o PSM por sua vez é transformado em código de uma determinada linguagem de programação.

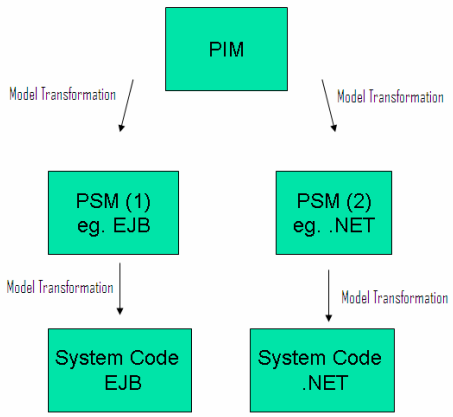


Figura 1.9. Transformações em MDA

Para que o processo da MDA funcione, é necessário que os seguintes requisitos sejam preenchidos:

- Construção de modelos em alto nível que contenham informações detalhadas sobre o domínio em questão.
- Linguagens bem definidas e padronizadas para escrever modelos em alto nível.
- Definições de como o PIM é transformado para um PSM específico, de modo a tal transformação poder ser automaticamente executada. Essas definições estarão em domínio público, padronizadas, mas poderão ser *home-made*, ou seja, feitas pela própria aplicação.
- Uma linguagem para escrever essas definições. Esta linguagem deve ser usada pelas ferramentas de transformação, além disso, deve ser uma linguagem com algum grau de formalismo.
- Ferramentas que executam as definições das transformações. Preferencialmente estas ferramentas devem oferecer a flexibilidade para mudar os passos da transformação de acordo com as necessidades específicas de cada projeto.
- Ferramentas que executam a transformação de PSM para código.

Os principais benefícios da MDA são:

1. Produtividade

Na MDA, o foco maior é dado à construção do modelo PIM. Com esse enfoque a produtividade torna-se maior, pois os desenvolvedores não precisam mais gastar tempo com o projeto específico de acordo com a plataforma escolhida. Esses detalhes técnicos específicos de plataforma são acrescentados ao PIM pela transformação PIM $\Rightarrow$ PSM. Além disso, a maior parte do código também é gerado automaticamente pela transformação PSM $\Rightarrow$ Código. Claro que o projeto e a especificação dessas transformações específicas são tarefas custosas e complexas, portanto influenciam no cronograma do Projeto. Mas uma vez essa transformação definida, ela poderá ser aplicada diretamente aos demais projetos, resultando assim num ganho de produtividade.

2. Portabilidade

A portabilidade se encontra no modelo PIM. O modelo PIM pode ser transformado em vários PSMs de diversas plataformas, ou seja, toda a informação especificada no PIM é completamente portável.

3. Interoperabilidade

A interoperabilidade a que a MDA se refere é a comunicação entre os PSMs.

4. Manutenção e Documentação

Na MDA, com a ajuda das ferramentas MDA, alterações feitas no PSM podem ser refletidas no seu PIM de origem, mantendo PIM, PSM e documentação consistentes até o término do projeto, o que não ocorre entre os modelos durante um processo tradicional de desenvolvimento de software.

4.3.2.2. Análise dos Trabalhos Existentes

Algumas abordagens já utilizam MDA para desenvolver arcabouços de segurança.

Em [BASIN, DOSER 2005], é apresentada uma abordagem MDA para engenharia de segurança, chamada segurança dirigida a modelo (*Model Driven Security*). Esta abordagem utiliza uma linguagem de modelagem baseada na UML chamada SecureUML [BASIN, DOSER 2002], que integra controle de acesso baseado em papéis (*role-based access control - RBAC*) no processo de desenvolvimento de software dirigido a modelo. A informação de segurança integrada nos modelos UML é utilizada para gerar infra-estruturas de controle de acesso.

Segurança dirigida a modelo provê métodos e ferramentas para integrar segurança no processo de desenvolvimento. A idéia chave é o uso de abstração, construindo modelos visuais que integram o projeto do sistema com o projeto de segurança e utiliza técnicas de geração automática de código para automatizar a construção de sistemas a partir desses projetos.

O modelo do projeto é combinado com o modelo de segurança, criando um novo tipo de modelo, chamado de modelo de projeto e segurança (*security design model*) . Neste modelo, as políticas de segurança se referem aos elementos do modelo do sistema, como componentes, objetos de negócio, métodos, atributos, etc.

A Figura 1.10. representa a abordagem de segurança dirigida a modelo, onde o modelo do projeto é combinado com o modelo de segurança e automaticamente transformado na arquitetura do sistema. O modelo do projeto é representado com um digrama de classes que é enriquecido com novos elementos de modelagem representando papéis e permissões. Esta combinação pode ser utilizada para definir uma política de controle de acesso de uma aplicação.

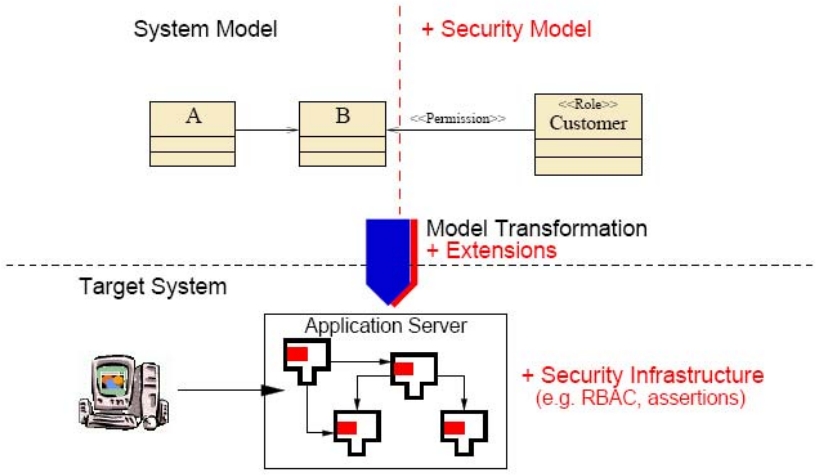


Figura 1.10. Segurança dirigida a modelo

Em [BASIN, DOSER 2005] foi construída uma ferramenta que suporta a modelagem da política de controle de acesso e que gera infra-estruturas de segurança que são compatíveis com diferentes padrões para sistemas baseados em componentes como J2EE/EJB e .NET. A utilização desta ferramenta promove algumas vantagens, como por exemplo, a simplificação da especificação da política de segurança,

identificação das falhas de segurança durante o processo de desenvolvimento, a implementação pode ser mantida de forma consistente com a política de segurança modelada e implementações podem ser migradas para novas plataformas mudando as regras de transformações utilizadas.

A Figura 1.11. exemplifica a modelagem de um sistema de acordo com a linguagem SecureUML através de um diagrama de classes. O exemplo dado é uma aplicação bancária simplificada. As entidades no exemplo são: *Account* (conta), que contém os atributos *owner* (dono) e balance (*saldo*) assim como métodos *withdraw* (retirar) e *desposit* (depositar). As contas, seus atributos e seus métodos representam os recursos que necessitam proteção. Além disso, há três papéis, que formaliza tipos diferentes de usuários: *Customer* (clientes), *Employee* (empregados) e *Manager* (gerentes). Neste contexto, a política de segurança será:

(P1) Cada empregado pode ler informação associada com todas as contas assim como criar novas contas. Além disso, ele pode alterar o dono de uma conta.

(P2) Em adição às permissões dos empregados, cada gerente pode deletar contas

(P3) Cada cliente pode ler todas as informações associadas a suas próprias contas

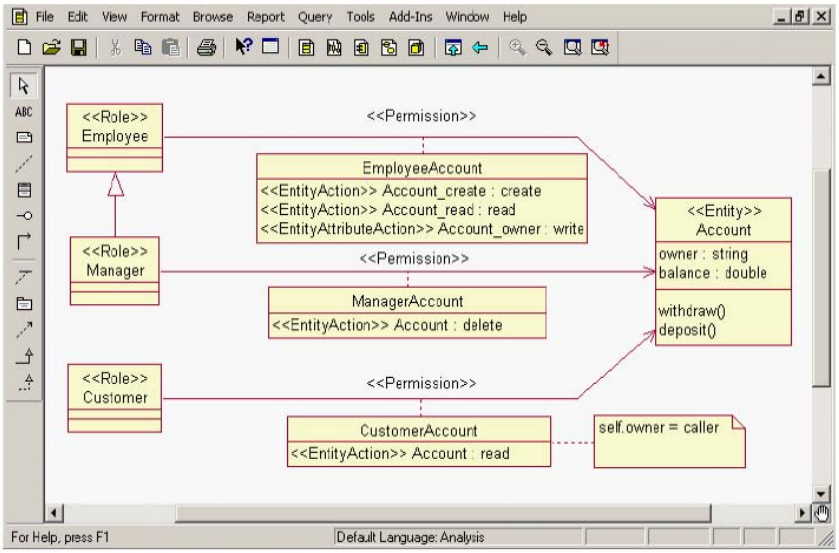


Figura 1.11. Modelando um cenário de um banco

Classes são utilizadas para definir a entidade *Account* (Conta) com seus atributos e métodos e para definir os papéis *Customer* (Cliente), *Employee* (Empregado) e *Manager* (Gerente). *Manager* (Gerente) é um sub papel de *Employee* (Empregado), o que significa que ele herda todas as permissões de *Employee* (empregado). As classes do meio conectam os papéis aos recursos de forma a modelar a política de controle de acesso. A permissão de um *Employee* (Empregado) para acessar um *Account* (Conta) é definida como uma associação com o estereótipo <<Permission>> entre as duas classes. Para restringir essa permissão, a associação é aplicada à classe *EmployeeAccount* (ContaEmpregado). Um atributo da classe de associação especifica o tipo de acesso ao recurso protegido.

No exemplo, (P1) é modelado da seguinte forma: a classe *EmployeeAccount* (ContaEmpregado) contém três atributos. Dois deles contêm o estereótipo *EntityAction* e possuem os tipos *create* (criar) e *read* (ler), respectivamente. Isto formaliza que um empregado possui o direito de criar novas contas e ler. O terceiro atributo na classe de associação *EmployeeAccount* (ContaEmpregado) expressa que um empregado pode alterar o valor do atributo owner (dono) de uma conta.

(P2) é modelado como uma herança entre papéis e definindo uma permissão adicional com o tipo de ação *delete* (deletar) entre *Manager* (gerente) e *Account* (Conta).

Para modelar (P3), é necessário formalizar que cada cliente pode ler contas, mas somente as contas que ele é dono. A permissão de ler contas é feita da mesma forma que (P1). Para expressar a restrição de dono, é necessário modelar uma *constraint*, que nesse caso, é especificada utilizando OCL (*Object Constraint Language*), que faz parte da UML. Neste exemplo, *CustomerAccount* (contaCliente) é restringida através da *constraint* self.owner = caller.

Como na primeira abordagem apresentada [BASIN, DOSER 2005], em [FINK, KOCH, PAULS 2004] é apresentada uma abordagem MDA para desenvolver políticas de controle de acesso em sistemas distribuídos. Os modelos são expressos como modelos MOF [MOF 2002] enriquecidos por perfis UML. MOF é um padrão de meta linguagem definido pela OMG a partir do qual onde outras linguagens de modelagem podem ser especificadas. A vantagem do MOF é fazer com que a definição de meta-modelos seja independente do domínio da aplicação e prover um conjunto de conceitos conciso e único para a definição de meta-modelos. Além disso, múltiplos meta-modelos podem ser gerenciados pelo MOF e relações entre eles podem ser utilizadas como base para uma transformação entre modelos.

Em [FINK, KOCH, PAULS 2004], um modelo de controle de acesso é especificado por um meta-modelo independente de domínio e de plataforma. Este meta-modelo pode ser instanciado em um domínio de aplicação. Além disso, o meta-modelo pode ser refinado para um meta-modelo especifico de plataforma (CORBA, J2EE, SOAP). A combinação de ambos resulta em um modelo específico de plataforma contendo uma aplicação de controle de acesso com a implementação especifica de alguma plataforma.

O modelo de controle de acesso utilizado é o VBAC (*View-Based Acces Control*) [Brose 2001], que é uma extensão do RBAC para sistemas distribuídos. VBAC é um modelo de controle de acesso feito para suportar o projeto e gerenciamento de políticas de controle de acesso em sistemas orientados a objeto. A principal característica do VBAC é ter uma visão dos direitos de acesso, que são as permissões ou negações para operações de objetos distribuídos. Visões são atribuídas aos principals, isto é, a sujeitos individuais ou papéis, e um principal possui acesso a uma operação de um objeto se ele possui uma visão do objeto com a permissão para chamar a operação. O principal não possui acesso se a operação está explicitamente negada em outra visão deste objeto que está disponível ao perfil ou se nenhuma permissão for achada.

O processo de desenvolvimento é feito da seguinte forma: o analista cria o modelo independente de plataforma VBAC (VBAC PIM). Após a fase de especificação, é necessário escolher a tecnologia que será utilizada para a implementação. Os modelos independentes de plataforma são mapeados para os modelos específicos de plataforma.

Para mapear os modelos relacionados ao controle de acesso, o VBAC-PIM é compilado para o VBAC-PSM (Modelo específico de plataforma VBAC). Por ultimo, o VBAC-PSM é utilizado para gerar a infraestrutura. Esses passos podem ser visualizados na Figura 1.12.

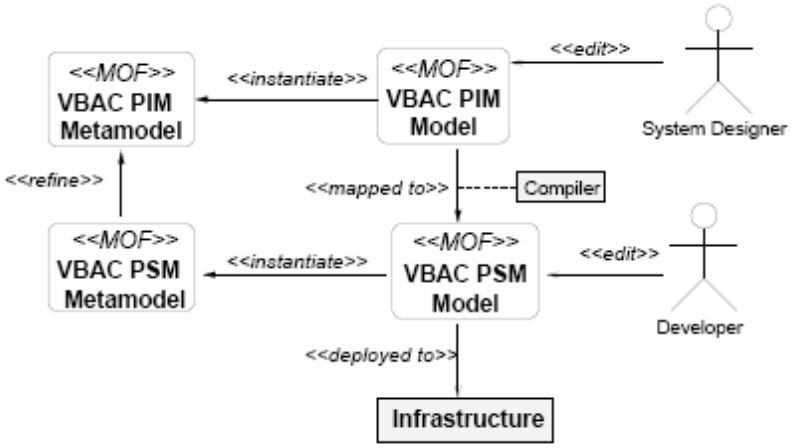


Figura 1.12. Modelos MDA-VBAC

Em [JIN 2006], é apresentada um arcabouço que utiliza a abordagem MDA juntamente com perfis UML para construir aplicações RBAC e especificações de segurança para sistemas distribuídos. Além disso, é mostrado um estudo de caso exemplificando como os perfis UML apresentados podem ser utilizados nas fases iniciais do sistema gerando automaticamente as especificações de segurança do sistema no formato XACML (*eXtensible Access Control Markup Language*).

XACML [GODIK, MOSES 2003] foi desenvolvida pelo OASIS [OASIS 2007] e descreve um formato para a definição de políticas de acesso em XML. O padrão XACML define linguagens de marcação que permitem especificar políticas de segurança, requisições e respostas para decisões de controle de acesso, permitindo a organizações utilizarem essas políticas para controlar acesso a conteúdos e informações protegidas.

O funcionamento proposto por essa especificação baseia-se em duas figuras principais: PEP (*Policy Enforcement Point*) e PDP (*Policy Decision Point*). O PEP é a entidade responsável por formular a solicitação de acesso. O PEP, dessa forma, envia a requisição ao PDP, que é responsável por avaliar a requisição. O PDP localiza as políticas aplicáveis e, baseado na avaliação das mesmas, informa a decisão de acesso, autorizando ou não a requisição.

Em fevereiro de 2005, a OASIS aprovou o padrão RBAC XACML [RBAC XACML 2004] que define um perfil para o uso do XACML junto com os requisitos do RBAC. Este padrão especifica quatro tipos de políticas para construir uma solução RBAC:

- Permission <PolicySet> ou PPS : PPS é uma coleção de permissões associadas a um papel.
- Role <PolicySet> ou RPS: RPS conecta um papel aos seus PPS correspondentes que contêm as permissões atuais associadas ao papel.

- Role Assignment <Policy> ou <PolicySet>: Este tipo é opcional. É utilizado para responder a questão se um sujeito tem permissão para fazer parte de um papel.
- HasPrivilegeOfRole <Policy>: Este tipo é opcional. É utilizado para responder a questão se um sujeito possui privilégios associados ao papel.

O processo de desenvolvimento é feito da seguinte forma: o desenvolvedor cria o modelo independe de plataforma (RBAC XACML PIM) baseado nos requisitos de controle de acesso da aplicação. Depois, o RBAC XACML PIM é transformado no modelo específico de plataforma (RBAC XACML PSM) que é utilizado para gerar os arquivos da infra-estrutura de segurança. Após a geração automática, o desenvolvedor implementa as partes que faltam, compila e testa o sistema. Neste trabalho, foi utilizado o pacote SunXACML, que foi projetado e desenvolvido pela empresa *Sun Microsystems*, é uma API que implementa a especificação XACML. Essa biblioteca é constituída por um conjunto de classes Java que interpretam a linguagem XACML. O Sun XACML facilita o desenvolvimento de PDPs e PEPs, fornecendo rotinas úteis para a utilização destas funções.

Em [JIN 2006], uma ferramenta de modelagem RBAC XACML foi projetada e implementada como um componente acoplável do Eclipse [ECLIPSE 2007] e utiliza os arcabouços EMF [EMF 2007] e GEF [GEF 2007]. Esta ferramenta foi elaborada para demonstrar a abordagem MDA proposta. Ela permite que o usuário crie modelos visuais para aplicações RBAC desde as fases iniciais do sistema e utilize técnicas de geração de código para automatizar a construção de sistemas a partir desses modelos.

Esta ferramenta também provê: (1) um editor visual que permite o usuário visualizar e editar modelos RBAC XCAML graficamente, (2) transforma e gera automaticamente arquivos de segurança no formato XACML, (3) cria um modelo EMF para usuário, papel e recursos nos modelos RBAC XACML e (4) valida automaticamente o modelo para evitar erros no projeto.

Todas as abordagens descritas até o momento, permitem a definição de políticas de controle de acesso utilizando perfis UML [UML 2007] (estereótipos, valores etiquetados) e OCL [OCL 2003] para especificar restrições, onde esses modelos são modelos de permissões. Ou seja, o PIM é modelado com requisitos de segurança. Porém, o fator humano pode interferir na robustez de segurança em qualquer ponto onde intervenção manual é requerida. Isto é, a partir do momento onde a segurança do sistema é ativada somente através da modelagem feita manualmente, ela se torna suscetível a erros humanos, prejudicando a qualidade da implementação dos requisitos de segurança. Ou seja, o PIM sendo marcado com requisitos de segurança pode provocar uma sobrecarga de atribuições ao desenvolvedor, expondo à falhas de segurança e poluindo o modelo de negócio e consequentemente, dificultando sua compreensão.

Nesse contexto, outra abordagem que utiliza MDA para desenvolver arcabouços de segurança é [FIORIO 2007]. Nessa abordagem foi criado um arcabouço de segurança que é dividido em duas partes, uma responsável pelas regras de segurança, chamada de arcabouço de regras de segurança, e outro responsável pela administração de segurança específica de cada aplicação, chamada de arcabouço administrativo de segurança.

O arcabouço de regras de segurança é responsável pela autenticação e autorização dos artefatos da aplicação. Ele simplifica e flexibiliza a modelagem dos requisitos de segurança seguindo o padrão arquitetural MVC [MVC 2007], provendo proteção por níveis (camadas) de vista, controle e dados. A segurança é aplicada implicitamente no sistema através de transformação entre modelos e codificação automática. O desenvolvedor continua modelando seu sistema normalmente, sem se preocupar com a segurança e mesmo assim, a segurança será gerada e o sistema estará protegido.

Já o arcabouço administrativo de segurança representa o modelo específico de segurança da aplicação, que depende da política de controle de acesso da organização. O arcabouço criado estende o modelo de controle de acesso RBAC com contextos, porém o padrão deixa em aberto a interpretação de usuário, papel, autorização e contexto para ser especificado em modelos mais detalhados. Porém, visando facilitar o desenvolvimento da aplicação, um arcabouço administrativo de segurança genérico é gerado automaticamente, de forma que só é necessário que a aplicação tenha que criar seu próprio modelo caso este não esteja de acordo com a política de segurança da organização.

O processo de desenvolvimento do arcabouço de segurança criado pode ser visto na Figura 1.13. O arquiteto de negócio desenvolve a arquitetura MDA de negócio através de extensões dos perfis UML e criando dessa forma, um engenho de transformação convencional. O arcabouço criado é transparente ao desenvolvedor de negócio, pois este continua modelando o PIM de negócio como de costume e utilizando o engenho de transformação convencional criado pelo arquiteto de negócio para realizar as transformações de modelo de PIM para PSM e de PSM para código. Os desenvolvedores não precisam tomar nenhuma atitude em relação à segurança. Eles apenas precisam indicar no PIM as entidades do RBAC com contextos, ou seja, usuário, papel, autorização e contexto.

Por sua vez, o arquiteto de segurança, desenvolve arquitetura de segurança criando um engenho de transformação de segurança seguro e um gerenciável. O arcabouço de segurança é gerado através do engenho de transformação de segurança seguro. Já o gerenciável é responsável pela geração padrão da aplicação de segurança do sistema. Esta aplicação é uma aplicação administrativa de segurança. Ela é responsável pelo cadastro de perfis, usuários, permissões, etc. Porém, cada sistema possui sua própria política de segurança, de forma que a aplicação padrão possa não se encaixar no modelo de segurança do sistema. Nesse caso, o engenheiro de segurança deve desenvolver a aplicação de segurança específica do sistema e fazer a ligação dessa aplicação com o arcabouço de segurança gerado.

A implementação de segurança é gerada automaticamente através do modelo. O fato de ser gerado automaticamente provê uma liberdade ao arquiteto de segurança à medida que ele pode alterar o que é gerado, a todo o momento, sem influenciar ou prejudicar o desenvolvimento do sistema. Em caso de troca de plataforma, o arquiteto apenas precisa implementar a mesma solução para outra plataforma, ou seja, criar novas regras de transformação específicas da plataforma escolhida. Além disso, o fato de ser gerado automaticamente e sem intervenção humana garante a qualidade do código e conseqüentemente do sistema.

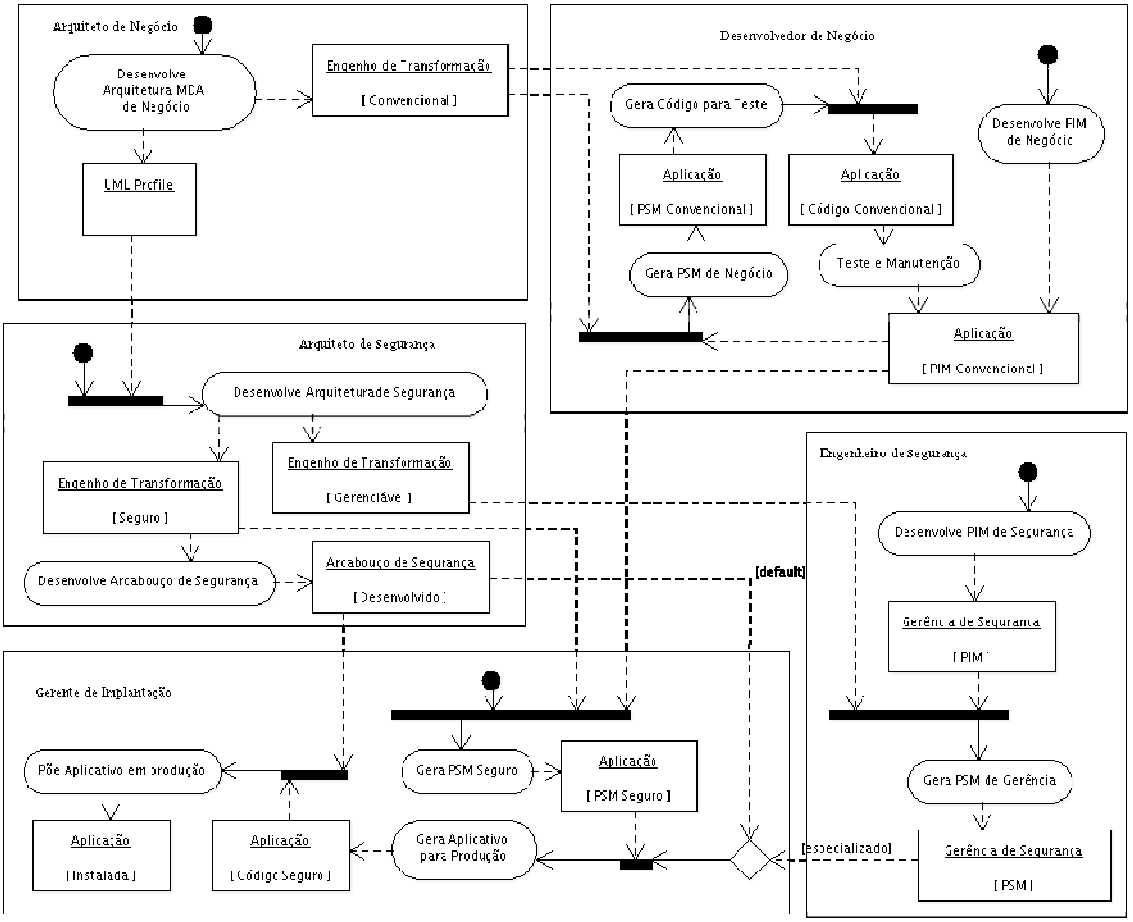


Figura 1.13. Projeto do arcabouço

Após a implementação da ligação entre o arcabouço de segurança e a aplicação de segurança é necessário que haja um gerente de segurança que faça a manipulação dos artefatos de segurança, como o cadastro de usuários, perfis e permissões. Se esses cadastros não forem feitos, o sistema irá barrar qualquer acesso, pois o usuário não conseguira se autenticar e com isso não terá acesso a nenhum artefato. A segurança é dinâmica, isto é, ela é manipulada em cima de permissões do usuário, e essas permissões podem ser criadas ou removidas dinamicamente. Por exemplo, se as permissões são agrupadas por perfis, é possível criar ou remover perfis e associar permissões aos perfis sem que seja necessário alterar o modelo de negócio. Dessa forma, o modelo não se torna poluído, continuando a ter apenas o negócio propriamente dito.

O gerente de implantação é a pessoa responsável pela ativação de segurança. Ou seja, é ele que, através do PIM modelado pelo desenvolvedor de negócio, da aplicação padrão ou específica de segurança do sistema e do arcabouço de segurança, gera o PSM com segurança e em seguida, o código contendo todos os artefatos de segurança gerados que é colocado em produção. A segurança pode ser excluída ou incluída (*switch on-off*) na geração do sistema de forma configurável sem interferir no trabalho dos desenvolvedores, facilitando o desenvolvimento e a configuração do sistema. Essa exclusão ou inclusão da segurança pode ser feita por níveis, onde é possível escolher em que níveis a segurança será ativada.

Nesse contexto, podemos perceber que a grande mudança ocorre quando a segurança está ativada, pois nesse momento, na transformação do PIM para o PSM, os novos requisitos de segurança requerem que um novo PSM seja gerado incluindo os artefatos de segurança. Consequentemente, a transformação de PSM para código também é alterada devido ao fato da criação de toda infra-estrutura de segurança do sistema. A Figura 1.14. exemplifica as transformações em alto nível de MDA com as extensões de segurança.

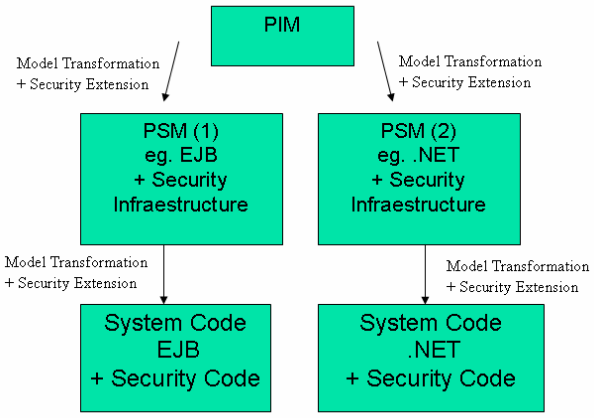


Figura 1.14. Transformações em MDA com extensões de segurança

O arcabouço foi utilizado e testado em um sistema militar brasileiro que possui uma política de segurança elevada, permitindo inúmeras vantagens em sua utilização pudessem ser percebidas.

Ele contribui na integração de segurança durante o desenvolvimento do sistema simplificando o desenvolvimento de sistema, aumentando a produtividade e a qualidade das especificações de segurança e com isso, aumentando consideravelmente a qualidade e a manutenibilidade dos sistemas construídos. O desacoplamento entre a implementação do negócio propriamente dito e a solução de segurança, facilita a evolução da solução fazendo com que ela não se torne obsoleta, principalmente pelo fato da segurança poder ser gerada automaticamente.

4.4. Conclusão

Os Modelos de Segurança constituem uma importante ferramenta para a definição de políticas de segurança para qualquer ambiente computacional. Ao longo dos anos, diversos modelos de segurança foram propostos, baseando-se nas mais variadas premissas e visando atender a diferentes aspectos de segurança.

Neste capítulo, fizemos uma revisão dos trabalhos mais recentes na área de modelos de segurança computacional, destacando modelos considerados clássicos na área e que ainda hoje são amplamente utilizados nos mais variados ambientes. Vimos a essência destes modelos, analisando criticamente a sua validade e adequação, bem como alguns aspectos de sua implementação.

Podemos perceber que os modelos baseados em papéis possuem algumas vantagens sobre os outros modelos, dentre as quais podemos destacar a separação de

tarefas, o princípio do mínimo privilégio e a possibilidade de incorporar, de alguma forma, a hierarquia natural das organizações ao controle de acesso. Devido às suas características os sistemas baseados em papéis representam uma importante tendência na área de controle de acesso em sistemas computacionais.

Apesar de grande parte dos sistemas de informação depender de requisitos de segurança de alta qualidade, constantemente há noticias sobre vulnerabilidades e falhas de segurança. Isto se deve a vários motivos, mas principalmente pelo fato deste requisito ser raramente considerado nos estágios iniciais do desenvolvimento de software e ser delegado a segundo plano ao longo do mesmo. Outro fato que contribui para esse problema é que os arcabouços de segurança existentes não oferecem um modelo arquitetural suficientemente flexível para acompanhar as necessidades do negócio em desenvolvimento ou limitam o escopo da arquitetura de que o negócio necessita. Devido à diversidade de requisitos de negócio de cada aplicação, arquitetos de software e analistas necessitam cada vez mais criar seus próprios arcabouços e manter suas integridades e coerência em relação às necessidades do negócio as quais eles atendem.

Vimos que são necessárias novas formas de desenvolver software seguro, baseadas não somente na aplicação das teorias existentes mas também na adoção de um processo de desenvolvimento que considere os requisitos de segurança como parte integral do projeto de construção de software. Além disso, o aumento da complexidade dos sistemas aliado à dinamicidade inerente às regras de negócio das aplicações atuais faz com que o processo de desenvolvimento de segurança necessite cada vez mais de arcabouços flexíveis e melhor gerência de requisitos e mudanças.

Neste contexto, apresentamos técnicas de integração de soluções de segurança nas fases de análise e projeto de sistemas de software, tais como a utilização de padrões de projeto de segurança e de uma abordagem orientada a modelos como forma de auxiliar arquitetos e projetistas a construir sistemas seguro.

Presentemente, existe uma grande lacuna entre as soluções teóricas e o que é de fato implementado na área de segurança. Portanto, a abordagem de padrões de projeto pode e deve ser aplicada à segurança como forma de preencher essa lacuna promovendo várias vantagens, entre elas: novatos podem atuar como especialistas, especialistas em segurança podem identificar, nomear e discutir problemas e soluções de modo mais eficiente, problemas podem ser resolvidos de uma forma estruturada, dependências de componentes podem ser identificadas e consideradas de forma apropriada.

Atualmente, há vários trabalhos relacionados ao assunto de padrões de projeto de segurança. Dentre eles, podemos citar um guia técnico de padrões de segurança [Blakley et al 2004] publicado pelo Opengroup [OpenGroup 2007]. Já em [Fernandez and Pan 2001] são discutidos três padrões que correspondem a modelos de segurança: Autorização, Controle de Acesso Baseado em Papéis e Segurança Multi-nível; em [KIENZLE, ELDER 2002] encontramos um repositório de padrões de segurança com vinte e seis padrões e três mini-padrões. O foco desses padrões está na aplicação de segurança na web. Os padrões estão disponíveis em [KIENZLE, ELDER 2002]. O relatório final do projeto contendo resumo de todos os padrões esta disponível em [KIENZLE et al 2002]. Além desses trabalhos, vários livros foram publicados, dentre eles podemos citar Security Engineering With Patterns: Origins, Theoretical Models, and New Applications [Schumacher 2003], Security Patterns: Integrating Security and

Systems Engineering [Schumacher et al 2005] e Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management [Steel et al 2005].

Vimos que para obter os benefícios da utilização de padrões de projeto, é necessário saber quando e como utilizá-los a partir de suas definições e que uma abordagem orientada a modelo (MDA) pode auxiliar o processo de desenvolvimento e criação de arcabouços de segurança. Com isso, podemos obter uma implementação controlada dos padrões de projeto e desta forma, garantir aderência às diretrizes arquiteturais da organização e às questões de segurança.

A utilização de segurança com MDA permite que segurança esteja integrada na modelagem de sistemas à medida que modelos de projeto são combinados com modelos de segurança e técnicas de geração automática de código são utilizadas para automatizar a construção de sistemas a partir desses modelos, dessa forma auxiliando a criação de arcabouços de segurança flexíveis e garantindo que os requisitos de segurança sejam levados em consideração durante todas as fases do processo de desenvolvimento do sistema. Com isso, as falhas de segurança podem ser identificadas mais rapidamente no processo de desenvolvimento e a implementação é mantida consistente com a política de segurança modelada, além de poder ser migrada para novas plataformas.

Além disso, a utilização de MDA contribui na integração de segurança durante a construção do sistema simplificando o seu desenvolvimento, aumentando a produtividade e a qualidade das especificações de segurança e com isso, aumentando consideravelmente a qualidade e a manutenibilidade dos sistemas construídos além de facilitar a evolução da solução de segurança.

Vimos que algumas abordagens já integram a segurança a MDA. Grande parte dessas abordagens permitem a definição de políticas de controle de acesso através da marcação do modelo utilizando perfis UML [UML 2007] (estereótipos, valores etiquetados) e OCL [OCL 2003] para especificar restrições. Ou seja, o PIM de negócio é marcado com requisitos de segurança.

Referências

AHN, G.-J.; SANDHU, R. (1999) “The RSL99 Language for Role-Based Separation of Duty Constraints”, In Proceedings of 4th ACM Workshop on Role-Based Access Control, Fairfax, VA, Oct, p. 43-54.

ALEXANDER, C. (1978) “A Pattern Language”, Oxford Press, Oxford, R. Unido.

ALEXANDER, C. (1979) “A Timeless Way of Building”, Oxford Press, Oxford, R. Unido.

AMOROSO, E. G. (1994) “Fundamentals of Computer Security Technology”. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994, ISBN: 0-13-108929-3.

BASIN, D. AND DOSER, J. (2002) “SecureUML: A UML-Based Modeling Language for Model-Driven Security”. 5th International Conference on the Unified Modeling Language, Lecture Notes in Computer Science 2460.

BASIN, D.; DOSER, J. (2005) “Model Driven Security: from UML Models to Access Control Infrastructures. In 5th International School on Foundations of Security Analysis and Design”, FOSAD.

BELL, D. E.; LA PADULA, L. J. (1973) “Secure Computer Systems : Mathematical Foundations”, MTR-2547 Vol. I, The MITRE Corporation, Bedford, Massachusetts, Mar.

BIBA, K. J. (1977) “Integrity Considerations for Secure Computer Systems”, Technical Report ESD-TR-76-372, USAF Electronic Systems Division, Hanscom Air Force Base, Bedford, Massachusetts, Apr.

BISHOP, M. (2003) “Computer Security: Art and Science”, Addison Wesley.

BLAKLEY,B.;HEALTH,C. (2004) “Security Design Patterns”, Technical Guide, 2004, Doc. No. G031, ISBN: 1-931624-27-5.

CLARK, D.; WILSON, D. (1987) “A comparasion of commercial and military computer security policies”, Proceedings of the IEEE Computer Society Simposium of Research in Security and Privacy, Los Alamitos, Calif., p. 184-194.

CHRISTOPHER STEEL, RAMESH NAGAPPAN, RAY LAI. (2005) “Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management,” Prentice Hall.

DENNING, D. (1976) “A Lattice Model of Secure Information Flow”, Communication of ACM, Vol. 19, Nº. 5, May.

DENNING, D. (1982) “Criptography and data security”, Addison-Wesley.

ECLIPSE (2007) “Eclipse”, disponível em http://www.eclipse.org/, acesso em julho de 2007.

EMF (2007) “Eclipse Modeling Framework”, disponível em http://www.eclipse.org/emf/, acesso em julho de 2007.

GEF (2007) “Graphical Editing Framework”, disponível em http://www.eclipse.org/gef/, acesso em julho de 2007.

FERNANDEZ, E. (2000) "Metadata and authorization Patterns", Report TR-CSE-00-16, Dept. of Computer Science and Eng., Florida Atlantic University, May.

FERNANDEZ, E.;PAM,R. (2001) “A Pattern Language for Security Models”, Dept. of Computer Science & Engineering, Florida Atlantic University, 2001.

FERRAIOLO, D.F.; BARKLEY, J.F.; KUHN R. (1999) “A Role Based Access Control Model and Reference Implementation within a Corporate Intranet”, Proc.of. National Institute of Standars and Technology, Vol. 2, nº. 1, February, p. 34-64.

FINK, T., KOCH, M., PAULS, K. (2004) “An MDA approach to Access Control Specifications Using MOF and UML Profiles”, In Proceedings 1st International Workshop on Views On Designing Complex Architectures.

FIORIO, M.; EMMANOEL, C. O; PIRES, P. F. (2007) “Um arcabouço de segurança baseado em transformações de modelos em MDA”, Relatório técnico, Núcleo de Computação Eletrônica, UFRJ.

GAMMA, E.; HELM, R.; JOHNSON, H.; VLISSIDES, J.; WESLEY, A. (1995) "Design Patterns: Elements of Reusable Object-Oriented Software", ISBN: 0-201-63361-2.

GLENN, F. (1999) "RBAC in UNIX Administration", Proceedings of 4th ACM Workshop on Role-Based Access Control, Fairfax, VA, Oct., p. 95-101.

GODIK, S.;MOSES T. (2003) "eXtensible Access Control Markup Language (XACML) Version 1.0", OASIS Standard.

GOGUEN, J.A;. MESAJUER, J. (1982) "Security Policies And Security Models", Proceedings of IEEE symposium on Reseach in Security and Privacy.

ISO/IEC 9126-1. (2001) "Software Engineering — Product Quality — Part 1: Quality Model", International Organization for Standardization, ISO, Geneva, Switzerland.

JIN, X. 2006 "Applying Model Driven Architecture approach to Model Role Based Access Control System", Master of Science in System Science, University of Ottawa, Ottawa, Ontario, Canada.

KIENZLE, D. M. ; ELDER, M. C. (2002) "Final Technical Report: Security Patterns for Web Application Development", DARPA Contract F30602-01-C-0164, disponível em http://www.modsecurity.org/archive/securitypatterns/dmdj_final_report.pdf, acesso em junho de 2007.

KIENZLE, D. M.; ELDER, M. C.; TYREE, D. S.; EDWARDS-HEWITT, J. (2002) "Security Patterns Repository Version 1.0", disponível em http://www.modsecurity.org/archive/securitypatterns/dmdj_repository.pdf, acesso em junho de 2007.

LAMPSON, B.W. (1971) "Protection", Proc. 5th Princeton Conf. on Information Sciences and Systems, Princeton, p. 437.

LANDWEHR, C. E. (1981) "Formal models for computer security", Computing surveys.

LANDWEHR, C. E. (1983) "Best available technologies for computer security", IEEE Computer Vol. 16, No. 7, Jul, p. 86-100.

LANDWEHR, C.E.; HEITMEYER, C.L.; MCLEAN, J. (1984) "A security Model for Military Message Systems", ACM Transactions on Computer Systems, Vol. 2, No 3, Aug, p. 198-222.

LANDWEHR, C. E. (2001) "*Computer security*", IJIS.

LARMAN, C. (2004) "Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process", ISBN: 0131489062.

MACKENZIE, D.E.; POTTINGER, G. (1997) "Mathematics, Technology, and Trust: Formal Verification", Computer Security, and the U.S. Military, IEEE Annals of the History of Computing, Vol. 19, No 3.

MCLEAN, J. (1990) "The Specification and Modeling of Computer Security", IEEE Computer, Vol. 23, No 1.

MOF (2002) “Meta-Object Facility“, disponível em http://www.omg.org/cgi-bin/doc?formal/2002-04-03, acesso em julho de 2007.

MOTTA G. H. M. B.; FURUIE S. S. (2002a) “Um Modelo de Autorização Contextual para Controle de Acesso Baseado em Papéis”, WSeg.

MOTTA G. H. M. B.; FURUIE S. S. (2002b) “MACA: Uma Ferramenta de Autorização e Controle de Acesso para o Prontuário Eletrônico de Pacientes”, VIII CBIS.

OASIS (2007) “OASIS: Avancing E-Business Standards Since 1993”. Disponível em: http://www.oasis-open.org, acesso em junho de 2007.

OMG (2003) “MDA Guide V1.0.1”. Disponível em http://www.omg.org/cgi-bin/doc?omg/03-06-01, acesso em julho de 2007.

OPENGROUP (2007) http://www.opengroup.org/, acesso em junho de 2007.

OSBORN, S.; SANDHU, R.; MUNAWER, Q. (2000) “Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies”, ACM Transactions on Information and System Security, Vol. 3, no. 2.

RABAC XACML (2004). “XACML Profile for Role Based Access Control (RBAC)”, Organization for the Advancement of Structured Information Standards, disponível em http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf, acesso em junho de 2007.

SANDHU, R.; COYNE, E.J.; FEINSTEIN, H.L.; YOUMAN, C.E. (1996) “Role-Based Access Control Models”, IEEE Computer, Vol. 29, Nº. 2, Feb, p. 38-47.

SANDHU, R. (1998) “Role-Based Access Control”, In Advances in Computers, Volume 46. Academic Press.

SANDHU, R.; FERRAIOLO, D.; KUHN, D.R. (2000) “The NIST Model for Role-Based Access Control: Towards A Unified Standard”, Proc. of 5th ACM Workshop on Role- Based Acess Control, Berlin, Germany.

SANDHU, R.; PARK, J. (2003) “Usage Control: A vision for Next Generation Access Control”, In The Second International Workshop Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS), St. Petersburg, Russia, Sep.

SCHUMACHER, M. (2003) “Security Engineering With Patterns: Origins, Theoretical Models, and New Applications”, Springer Berlin, Heidelber.

SCHUMACHER, M.; FERNANDEZ-BUGLIONI, E.; HYBERTSON, D.; BUSCHMANN, F.; SOMMERLAD, P. (2005) “Security patterns. Integrating security and systems engineering”, John Wiley & Sons.

SUN XACML (2007) “Sun’s XACML Implementation Programmer’s Guide”, disponível em: http://sunxacml.sourceforge.net/guide.html, acesso em junho de 2007.

SUN (2007) “Sun Microsystems”, disponível em http://www.sun.com/, acesso em julho de 2007.

TSUKOMO, A et al. (1997) “Qualidade de Software: Visões de Produto e Processo de Software”, II ERI – SBC, Piracicaba, São Paulo, Brasil.

UML (2007) “Unified Modeling Language”, disponível em http://www.uml.org/, acesso em julho de 2007.

WESTPHALL, C.M. (2000) “Um esquema de autorização para a segurança em sistemas distribuídos de larga escala”, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina, Tese de doutorado, Florianópolis – Santa Catarina.

YODER, J.; BARCALOW, J. (1997) “Architectural Patterns for Enabling Application Security”, Pattern Languages of Programs, Monticello, IL.