
# XIV SIMPÓSIO BRASILSEIRO EM SEGURANÇA DE INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS

## ▶ MINICURSOS

SBC- SOCIEDADE BRASILEIRA DE COMPUTAÇÃO


SBSeg 2014 - Belo Horizonte, MG

XIV Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais

de 3 a 6 de Novembro de 2014 – Belo Horizonte, MG

# MINICURSOS

**Editora**
Sociedade Brasileira de Computação – SBC

**Organizadores**
Jeroen van de Graaf, UFMG
José Marcos Nogueira, UFMG
Leonardo Barbosa Oliveira, UFMG

**Realização**
Universidade Federal de Minas Gerais – UFMG

**Promoção**
Sociedade Brasileira de Computação – SBC

Edição: Vitor Mendes Paisante

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (14. : 3-6 nov. 2014 : Belo Horizonte)
Minicursos / XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2014) [recurso eletrônico] / organização: Jeroen van de Graaf ; José Marcos Nogueira ; Leonardo Barbosa Oliveira. Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2014.
viii 194 p. ; PDF ; 8.44MB

Inclui bibliografia
ISBN 978-65-87003-99-3 (e-book)

1. Ciência da Computação. 2. Segurança da informação. 3. Sistemas computacionais. I. Van de Graaf, Jerome. II. Nogueira, José Marcos. III. Oliveira, Leandro Barbosa. IV. Universidade Federal de Minas Gerais. V. Sociedade Brasileira de Computação. VI. Título.

CDU 004(063)

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

Biblioteca Digital da SBC – SBC OpenLib

**Índice para catálogo sistemático:**

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

# Sociedade Brasileira de Computação – SBC



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

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

# Comitês do SBSeg 2014

## Comitê de Organização

### Coordenação Geral

- Jeroen van de Graaf (UFMG)
- José Marcos Nogueira (UFMG)
- Leonardo B. Oliveira (UFMG)

### Coordenação do Comitê de Programa

- Diego F. Aranha (UNICAMP)
- Marinho P. Barcellos (UFRGS)

### Coordenadores do CTD

- Aldri Santos (UFPR)
- Anderson do Nascimento (UNB)

### Coordenador do FSC

- Pericles Luz

### Coordenador de Minicursos

- André dos Santos (UECE)

### Coordenador de Palestras e Tutoriais

- Michele Nogueira (UFPR)

### Coordenadores do WFC

- Cinthia Freitas (PUCPR)
- William Robson Schwartz (UFMG)

### Coordenador do WGID

- Marco Aurélio Amaral Henriques (UNICAMP)

### Coordenadores do WTE

- Diego F. Aranha (UNB)
- Jeroen van de Graaf (UFMG)

### Coordenador do WTICG

- Eduardo Souto (UFAM)

### Publicidade

- Mário S. Alvim (UFMG)

## Comitê de Programa

- Adriano Cansian (UNESP)
- Alberto Egon Schaeffer Filho (UFRGS)
- Aldri dos Santos (UFPR)
- Altair Santin (PUC-PR)
- Anderson Nascimento (UnB)
- André dos Santos (UECE)
- André Gregio (CTI)
- Antonio Augusto de Aragão Rocha (UFF)
- Arlindo L. Marcon Jr. (IFPR)
- Carla Merkle Westphall (UFSC)
- Carlos Maziero (UTFPR)
- Carlos Westphall (UFSC)
- Célio Albuquerque (UFF)
- Diego de Freitas Aranha (UNICAMP)
- Djamel Fawzi Hadj Sadok (UFPE)
- Eduardo Feitosa (UFAM)
- Eduardo Souto (UFAM)
- Emerson Ribeiro de Mello (IFSC)
- Eulanda Miranda dos Santos (UFAM)
- Fernando Magno Quintao (UFMG)
- Hao Chi Wong (Intel)
- Jean Martina (UFSC)
- Jeroen van de Graaf (UFMG)
- Joaquim Celestino Júnior (UECE)
- Joni da Silva Fraga (UFSC)
- Julio López (UNICAMP)
- Lau Cheuk Lung (UFSC)
- Leonardo B. Oliveira (UFMG)
- Lisandro Zambenedetti Granville (UFRGS)
- Luciano Paschoal Gaspary (UFRGS)
- Luiz Carlos Albini (UFPR)
- Luiz Fernando Rust da Costa Carmo (UFRJ)
- Marcos Simplicio (Poli-USP)
- Marinho P. Barcellos (UFRGS)
- Mário Alvim (UFMG)
- Michele Nogueira (UFPR)
- Michelle Wangham (UNIVALI)
- Paulo André da Silva Gonçalves (UFPE)
- Paulo Lício de Geus (UNICAMP)
- Paulo S. L. M. Barreto (Poli-USP)
- Pedro Velloso (UFF)
- Raul Ceretta Nunes (UFSM)

- Raul Weber (UFRGS)
- Ricardo Custódio (UFSC)
- Ricardo Dahab (UNICAMP)
- Roberto Gallo (KRYPTUS)
- Routo Terada (USP)
- Ruy José Guerra Barretto de Queiroz (UFPE)
- Sergio de Oliveira (UFSJ)

# CESeg

## Coordenadores

- Anderson C. A. Nascimento (UnB, coordenador)
- Aldri L. Dos Santos (UFPR, vice)

## Comitê consultivo

- Anderson Clayton Alves Nascimento (UnB)
- Aldri Luiz dos Santos (UFPR)
- Jeroen van de Graaf (UFMG)
- Ricardo Dahab (UNICAMP)
- Altair Santin (PUC-PR)
- Eduardo Souto (UFAM)
- Raul Ceretta Nunes (UFSM)
- Michele Nogueira (UFPR)
- Diego Aranha (UNICAMP)

# Colaboradores

- Artur Luis (UFMG)
- Júlia Terra (UFMG)
- Vitor Paisante (UFMG)

# Mensagem do Coordenador de Minicursos

O Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) é um evento científico promovido anualmente pela Sociedade Brasileira de Computação (SBC), o qual representa o principal fórum no país para a apresentação de pesquisas e atividades relevantes ligadas à segurança da informação e de sistemas, integrando a comunidade brasileira de pesquisadores e profissionais atuantes nessa área. Entre as principais atividades técnicas do SBSeg encontram-se os minicursos, que têm como objetivo a atualização em temas normalmente não cobertos nas grades curriculares e que despertem grande interesse entre acadêmicos e profissionais. Nesta edição do SBSeg 2014, de um total de 10 submissões de propostas de minicursos foram selecionados quatro minicursos, representando assim uma taxa de aceitação de 40%. O Comitê de Programa foi composto por 11 pesquisadores para a elaboração das revisões, sendo cada proposta avaliada por, pelo menos, 3 desses especialistas.

Este livro reúne então 4 capítulos produzidos pelos autores das propostas de minicursos aceitas. O Capítulo 1 mostra o estado da arte em sistemas de armazenamento nas nuvens que sejam tolerantes a faltas e intrusões; O Capítulo 2, discute as técnicas de device fingerprint que são aquelas empregadas para identificar (ou re-identificar) um usuário ou um dispositivo através de um conjunto de atributos (tamanho da tela do dispositivo, versões de softwares instalados, entre muitos outros) e outras características observáveis durante processo de comunicação; O Capítulo 3 apresenta métodos e ferramentas de detecção de botnets; Por fim, o Capítulo 4 analisa os desafios de segurança e as principais contramedidas descritas em trabalhos acadêmicos sobre as inovações das redes veiculares.

Meus sinceros agradecimentos aos membros da Comissão de Avaliação dos Minicursos do SBSeg 2014 por sua colaboração voluntária e dedicação, lembrando que o sucesso dos minicursos deve ser creditado ao trabalho conjunto desta equipe. Também gostaria de agradecer aos autores que submeteram suas propostas de minicursos e que nos motivam a realizar anualmente este evento de interesse, visibilidade e sucesso crescentes. Finalmente, não poderia de deixar de expressar também os meus sinceros agradecimentos aos coordenadores gerais do SBSeg 2014, Jeroen van de Graaf (UFMG), José Marcos Nogueira (UFMG) e Leonardo B. Oliveira (UFMG) por confiarem a mim a tarefa de conduzir os minicursos e pela disponibilidade e orientações providas ao longo do processo.

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

Belo Horizonte, novembro de 2014.

André Luiz Moura dos Santos (Universidade Estadual do Ceará)

Coordenador dos Minicursos do SBSeg 2014

# Sumário


# SBSeg 2014 — Belo Horizonte, MG

XIV Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais — SBSeg 2014

# Capítulo 1

# Tolerância a Faltas e Intrusões para Sistemas de Armazenamento de Dados em Nuvens Computacionais

Hylson Vescovi Netto, Lau Cheuk Lung, Rick Lopes de Souza

***Abstract***

*Distributed storage is an area already explored in the literature. The increasing availability of storage providers in the Internet - the clouds - creates opportunities for research on problems that arise with this global infrastructure, that has high geographical distribution and on-demand costs. Systems that works in this environment have to maintain data consistency, run over partitioned networks and achieve good performance. Systems that need to be reliable have to tolerate fault and intrusions. This minicourse presents concepts and techniques that are used in commercial cloud providers, also in scientific works that consider data storage in clouds.*

***Resumo***

*Armazenamento distribuído é uma área bastante explorada na literatura. A crescente disponibilidade de provedores de armazenamento na Internet - as nuvens - criou oportunidades de pesquisa para resolver desafios que surgiram com essa infraestrutura global, que possui alta distribuição geográfica e custos sob demanda. Sistemas que operam nesse ambiente precisam manter a consistência dos dados, trabalhar sob particionamento de rede e alcançar desempenho satisfatório. Sistemas que requerem alta confiabilidade precisam tolerar faltas e intrusões. Este minicurso apresenta conceitos básicos e técnicas que são utilizadas por provedores de nuvens comerciais, bem como por trabalhos científicos que apresentam soluções para o armazenamento de dados em nuvens.*

## 1.1. Introdução

Armazenamento distribuído de dados é um assunto bastante explorado pela literatura. Desde registradores compartilhados [2] a sistemas de arquivo em rede, protocolos tem

sido desenvolvidos para adicionar capacidades de tolerar faltas, tornando assim os sistemas mais confiáveis. Trabalhos recentes atuam sob o novo cenário criado pela disponibilização de serviços de armazenamento em nuvens, onde existem benefícios como a utilização sob demanda (custos sob demanda) e a elasticidade de recursos. As nuvens de armazenamento podem ser categorizadas em passivas ou ativas. As nuvens passivas não tem capacidade de executar código, assemelhando-se em muito aos registradores compartilhados: possuem operações básicas de leitura e escrita. Como exemplo de nuvens passivas temos a Amazon S3[1], a Microsoft Azure Storage[2], o RackSpace Cloud Block Storage[3] e a Google Cloud Storage[4]. As nuvens ativas oferecem capacidade de processamento, onde é possível, por exemplo, realizar comunicação entre nuvens e analisar ou validar comandos antes de sua execução. Exemplos de nuvens que disponibilizam recursos de processamento são a Google App Engine[5] e a Amazon EC2[6]. A disponibilidade de armazenamento em nuvens possibilitou uma nova fase de pesquisa em armazenamento distribuído: protocolos como o DepSky [5] e o SCFS [6] tornam possível o armazenamento de dados nas nuvens de maneira confiável.

Os provedores de armazenamento de dados em nuvens disponibilizam várias formas de utilizar os serviços. A aplicação cliente pode utilizar bancos de dados relacionais, por meio de comandos SQL; pode especificar um nome (uma chave) e um valor, em um banco de dados NoSQL; ou pode mapear dados como um sistema de arquivos remoto. Cada tipo de dado possui características e custos diferentes. Estas formas de utilização de serviço referem-se ao uso de uma nuvem em modo SaaS (*Software as a Service*), por meio de API's. Uma API (*Application Programming Interface*) é uma maneira padronizada que permite programas acessarem serviços. É possível também utilizar nuvens como IaaS - *Infrastructure as a Service*, de modo que seja possível instalar o software desejado, como um gerenciador de banco de dados, e assim interagir com a nuvem por meio de protocolos mais específicos. Entretanto, o SaaS é mais apropriado no caso de utilização em grande escala, oferecendo benefícios como elasticidade e custos sob demanda. A elasticidade se refere ao fato de o próprio provedor de nuvem controlar a criação ou remoção de recursos em função da demanda da aplicação.

A diversidade de características existente entre os provedores de nuvens públicas traz consigo um benefício no que se refere à capacidade de tolerar faltas [3]. Diferentes provedores de nuvens são administrados por diferentes pessoas, com diferentes estratégias, em ambientes de rede variados. Assim, a utilização de diferentes provedores de nuvens pode aumentar a tolerância a faltas. Há trabalhos que tem por objetivo realizar o armazenamento de dados em mais de um provedor de nuvem, visando alcançar benefícios como disponibilidade total. No artigo de Bessani et al. [5], experimentos apontam que o uso de múltiplos provedores de nuvens são a única forma de garantir 100% de disponibilidade de acesso a dados no ambiente da Internet. Essa disponibilidade total inclui tolerar faltas de acesso de provedores, ainda que em poucos momentos. Mesmo os raros

[1] http://aws.amazon.com/s3/
[2] http://azure.microsoft.com/en-us/services/storage/
[3] http://www.rackspace.com/cloud/block-storage/
[4] https://cloud.google.com/products/cloud-storage/
[5] https://developers.google.com/appengine/
[6] http://aws.amazon.com/ec2/

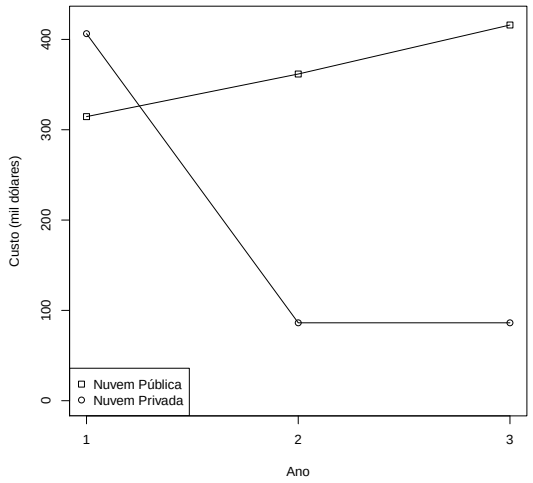
 

instantes de inacessibilidade de serviços tornam-se significativos quando o serviço é escalado em grande proporção, em função do acesso de muitos clientes. Por isso tornam-se importantes protocolos que garantam acesso contínuo aos dados.

Apesar da grande disponibilidade de provedores de serviços em nuvens públicas da atualidade, ainda existe alguma resistência em se utilizar exclusivamente nuvens públicas para, por exemplo, armazenar dados sigilosos. Há uma grande tendência em manter informações como metadados de arquivos ou controles de acesso em servidores internos às instituições; nesse caso, são apropriadas as nuvens privadas, que também possuem características como elasticidade, mas funcionam de forma dedicada a uma organização, não sendo compartilhada. Nuvens híbridas consideram a utilização simultânea de nuvens públicas e privadas. Existem provedores que já sugerem essa composição de nuvens híbridas[7], enquanto questões de confidencialidade e privacidade ainda estão sendo fonte de problemas ainda nos dias atuais - o recente caso do vazamento de fotos íntimas de celebridades[8] é mais um exemplo de consequências do simples envio de dados para as nuvens. Além de questões de segurança, o custo para criar e manter infraestruturas computacionais também pode ser um fator relativo na decisão de utilizar sistemas e dados em nuvens públicas ou privadas. Um estudo comparativo [18] apontou a relação entre a utilização de um mesmo sistema em nuvem pública e privada. Conforme mostra a figura 1.1, nuvens privadas podem fornecer um melhor benefício ao longo do tempo, apesar do maior custo inicial.

| Ano | Nuvem Pública ($) | Nuvem Privada ($) |
|---|---|---|
| 1 | 314.525 | 406.395 |
| 2 | 361.703 | 86.248 |
| 3 | 415.959 | 86.248 |

(a)




(b)

Figura 1.1: Valores gastos em nuvens, para uma mesma aplicação. Nuvens privadas podem apresentar melhor benefício após determinado período [18].

Uma das grandes ameaças à computação em nuvem é o controle que os provedores deste tipo de tecnologia detém sobre as comunicações e dados dos usuários com as empresas contratantes. Recentemente os Estados Unidos, por meio da agência *NSA*, espionaram milhões de ligações, comunicações e dados armazenados nos mais variados serviços de nuvem. Conclui-se então que esse tipo de mecanismo dá muitos poderes às

[7]http://www.emc.com/microsites/cio/articles/goulden/index.htm?pid=hp-goulden-article-180714

[8]http://www.latimes.com/business/la-fi-celebrity-hack-20140901-story.html

empresas de telecomunicações e provedores de nuvem que monitoram as atividades dos seus usuários. As empresas Google, Microsoft, Apple, Facebook, entre outras [19], são exemplos de empresas que tem parceria com o governo americano e que fornecem dados para espionagem. A seguir, serão elencados diversos aspectos referentes à segurança na computação em nuvens.

**Localização dos dados:** normalmente, os servidores dos provedores de nuvem estão localizados em outros países, como os Estados Unidos, no qual possuem rígidos controles de dados para controles internos e fazem com que os dados das corporações de outras empresas fiquem expostos a interesses destes países [20]. O usuário final normalmente não sabe onde estão armazenados seus dados e, com isso, terá dificuldades de buscar por seus direitos caso seus dados sejam expostos devido às diferentes leis vigentes em cada país.

**Segurança da rede:** na computação em nuvem os dados são obtidos das estações locais dos usuários, processados e armazenados na nuvem. Todos os dados que trafegam entre o usuário e a nuvem precisam ser cifrados com o objetivo de proteger os dados com conteúdos sensíveis. Para que isso seja feito, deve-se utilizar mecanismos de cifragem como o *Secure Socket Layer* (SSL). Estes provedores de serviço de nuvem devem estar protegidos contra ataques advindos da rede, como por exemplo, ataque do homem no meio, *IP spoofing* e captura de pacotes.

**Segregação de Dados:** uma das principais características da computação em nuvem são os múltiplos clientes que acessam a mesma instância do software na nuvem. Como resultado deste uso compartilhado, dados de diversos usuários serão armazenados no mesmo servidor. Com isso, ataques de invasão ao espaço de outros usuários tornam-se possíveis pelo não-isolamento seguro dos dados. Estes tipos de ataques podem ser realizados inserindo códigos maliciosos no provedor de serviços. Se o provedor executá-los sem uma verificação, existe um alto potencial de sucesso no ataque. O provedor de serviços deve prover mecanismos de separação segura entre os dados dos usuários.

**Autenticação e Autorização:** grande parte das empresas que utilizam a computação em nuvem armazenam as informações e credenciais de seus funcionários nesses provedores de serviços. Com isto, os dados ficam fora da proteção da empresa, tornando assim os dados dos funcionários vulneráveis a possíveis ataques. Uma das alternativas seria a utilização de parte destes serviços para os servidores internos da empresa. Desta forma, alguns dados sensíveis e autenticações poderiam estar protegidos física e logicamente.

**Confidencialidade dos Dados:** Quando indivíduos, empresas ou governos utilizam a computação em nuvem para armazenar seus dados, questões sobre a privacidade e confidencialidade são discutidas. Pela natureza distribuída da nuvem, os dados dessas organizações são compartilhadas em diversos servidores que são de propriedade externa e que são operados por terceiros. Existem diversos serviços de nuvem que armazenam dados sensíveis, como sistemas governamentais, de saúde, judiciários e sistemas pessoais. Algumas das implicações relacionadas a privacidade são:

- A preocupação com o sigilo dos dados não se restringe a dados pessoais, mas também de empresas e órgãos governamentais;

- Os direitos sobre a privacidade e confidencialidade dos dados muda conforme os tipos e categorias de dados que o provedor de conteúdo fornece para a nuvem;
- O armazenamento de dados pessoais e empresariais feito em nuvens pode gerar consequências negativas para a imagem com relação à privacidade dos dados;
- A localização dos servidores tem influência direta sobre os efeitos da privacidade e obrigações legais para aqueles provedores que fornecem serviços e armazenam dados;
- As informações armazenadas na nuvem podem estar localizadas em mais de um local, podendo assim estar sob jurisdições diferentes;
- Algumas leis podem obrigar os provedores de nuvem a fornecer dados de seus usuários alegando investigações sobre crimes ou outros motivos;

**Falhas:** servidores dos provedores de nuvem podem falhar por diversas razões e parar de oferecer seus serviços. Um sistema que preza pela segurança deve ser tolerante a faltas, mantendo o funcionamento mesmo que em menor capacidade. Essa ameaça é contínua nos provedores de nuvem, haja visto que o usuário não detém controle sobre o funcionamento total do serviço contratado. Por esse motivo, deve-se utilizar uma arquitetura diferenciada, que possa manter o funcionamento do sistema em casos de falha. Uma das alternativas que pode-se adotar é a utilização de múltiplos provedores de nuvem, garantindo assim uma maior heterogeneidade no fornecimento dos serviços e minimizando o impacto de possíveis falhas dos sistemas.

Ao final deste minicurso, espera-se que o leitor possa compreender os conceitos e técnicas que permitem a utilização de sistemas de armazenamento em nuvens, tanto em artigos científicos, quanto em descrições de produtos comerciais. A seguir temos um pequeno exemplo de texto que envolve os conceitos abordados neste minicurso[9]. O trecho transcrito refere-se ao funcionamento do Facebook[10], e foi apresentado por Harry Li em 2012, no evento PODC[11]:

> Facebook tem mais de um milhão de usuários, e usa *cache* distribuído em memória capaz de processar um bilhão de operações por segundo. Sua prioridade máxima é garantir uma experiência rápida, confiável e consistente ao usuário. Eles usam uma arquitetura geo-distribuída de regiões mestre e escravo, onde cada região tem seu próprio conteúdo web, *cache* e *clusters* de banco de dados. Uma região escravo atende apenas pedidos de leitura de dados e usa MySQL com replicação para sincronizar seus bancos de dados com o primário, reduzindo uma latência de acesso entre países de 70ms para 2ms. Embora o Facebook garanta apenas consistência eventual, eles usam um pequeno truque para garantir que um usuário possa ler suas próprias escritas: se o usuário recentemente atualizou dados, os consecutivos pedidos deste usuário são redirecionados para a região primária durante algum tempo.

[9] Caso o leitor não entenda o texto, este trecho pode ser lido novamente após o minicurso, sob a expectativa de sua compreensão.

[10] http://www.facebook.com

[11] http://www.podc.org/podc2012-report/

## 1.2. Replicação e Tolerância a faltas

Tolerância a faltas é uma área de pesquisa que tem por objetivo tornar um sistema capaz de continuar funcionando a despeito de problemas que possam ocorrer no mesmo, de forma transparente ao usuário. Normalmente é estabelecido um limiar $f$ que significa o número de faltas que podem ser toleradas no sistema; os recursos necessários para manter esse funcionamento transparente do sistema geralmente são definidos em função desse parâmetro $f$. Consequentemente, quanto mais faltas deseja-se que o sistema tolere, mais recursos são necessários.

Dentre as possíveis faltas que podem ocorrer em um sistema, destaca-se a falta denominada *crash*, também chamada falta de parada, onde um elemento do sistema, por algum motivo, interrompe definitivamente a interação com o usuário. Uma questão conhecida e bastante relevante em sistemas assíncronos é a dificuldade em diferenciar se essa falta de interação deve-se a uma falha total no elemento ou a uma eventual lentidão na rede. Em sistemas síncronos essa detecção é baseada em restrições temporais que orientam a verificação sobre o funcionamento do elemento: após um certo tempo sem interagir (*timeout*), considera-se que o elemento não funciona mais - sofreu um *crash*. Para tolerar este tipo de falta, redundância é a técnica mais utilizada: a falha de um elemento não compromete o sistema porque existem outros elementos que podem realizar as atividades necessárias.

Quando os componentes do sistema distribuído apresentam desvios de funcionamento além da simples interrupção de interação, é necessário utilizar técnicas mais elaboradas para garantir o funcionamento do sistema: é preciso tolerar faltas arbitrárias, ou bizantinas. Essas faltas podem ocorrer por quaisquer motivos, e sua detecção geralmente envolve critérios de maioria: os elementos são replicados, sendo necessário que mais da metade dos elementos apresente respostas iguais para que o resultado seja confiável. Alguns exemplos de faltas bizantinas são: modificações no sistema por invasores, substituição de respostas por valores antigos, interrupção de funcionamento de elementos (*crash*) e acessos indevidos por operadores internos do sistema. A partir de agora, será utilizado o termo BFT (*Byzantine Fault Tolerance*) para referenciar a expressão "tolerância a faltas bizantinas".

Replicação é a maneira mais prática de obter redundância em sistemas. A criação de réplicas implica na existência de mais de um provedor de serviço, trazendo vantagens para o cliente: *i*) o desempenho do sistema poderá ser melhor devido à utilização de réplicas que estiverem mais próximas ao cliente, resultando assim em um tempo de resposta menor; *ii*) a disponibilidade do serviço replicado é maior, pois a existência do serviço em diversos locais permite um maior conjunto de opções ao cliente, sobre qual réplica escolher para utilizar o serviço; *iii*) em sistemas replicados, geralmente tolera-se a perda ou o mau funcionamento de uma ou mais réplicas, agregando assim a tolerância a faltas como uma característica do sistema. Naturalmente, existem custos decorrentes da replicação de serviços ou recursos. Porém, dados os benefícios apresentados, considera-se a replicação como uma das maneiras mais efetivas para o provimento de tolerância a faltas em sistemas computacionais. O armazenamento de dados em provedores de nuvens também utiliza-se da replicação para aumentar a durabilidade do dado, bem como a disponibilidade, o desempenho no tempo de resposta, entre outras características.

Dentre as diversas maneiras de realizar a replicação, duas são mais conhecidas e utilizadas: a replicação passiva e a replicação ativa. A replicação passiva consiste em utilizar uma estratégia do tipo primário-secundário [10], de maneira que o cliente interage apenas com uma réplica principal, e esta réplica atualiza as outras réplicas secundárias. O termo ”passivo” refere-se ao fato de o cliente interagir apenas com uma réplica principal, e o sistema atualizar as demais réplicas passivamente, no sentido de não exigir a participação ativa do cliente para tal operação (o cliente não precisa solicitar que cada réplica secundária seja atualizada). A atualização entre a réplica primária e as secundárias pode ocorrer em tempo síncrono ou assíncrono, dependendo dos requisitos do sistema. Caso a atualização das réplicas secundárias seja síncrona, o cliente deverá aguardar até que as réplicas secundárias sejam atualizadas. Se a atualização das réplicas secundárias for assíncrona, o tempo de resposta do sistema será igual ao tempo de resposta da réplica primária, que após enviar a resposta ao cliente, poderá iniciar a atualização das demais réplicas. A figura 1.2a ilustra um sistema com replicação passiva: clientes acessam uma réplica primária, que atualiza as réplicas secundárias. Na replicação ativa (figura 1.2b), o cliente interage diretamente com todas as réplicas (ou um subconjunto delas). Pode-se aguardar pela resposta de todas as réplicas, ou de um subconjunto de respostas, dependendo do modelo do sistema. As setas possuem tracejado diferenciado para explicitar a comunicação de cada cliente com as réplicas.

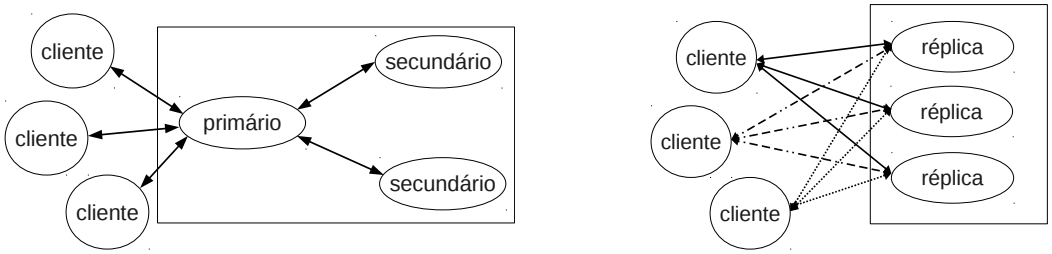



Figura 1.2: Replicação (a) passiva e (b) ativa [13].

Considerando o aspecto de tolerância a faltas, a replicação passiva é mais suscetível a falhas do que a replicação ativa, pois o cliente interage apenas com a réplica primária. Quando esta réplica primária sofre faltas de parada, o cliente inicia o contato com uma réplica secundária, que será considerada pelo cliente a nova réplica primária. Em geral esta verificação de falta de parada é realizada pelo cliente com a utilização de um temporizador (*timeout*). Dessa maneira, consideramos que na replicação passiva sob condições síncronas de tempo, tolera-se faltas de parada utilizando $f+1$ réplicas, onde $f$ é o número máximo de faltas toleradas. Considerando, por exemplo, $f = 1$, temos duas réplicas, uma sendo primária e outra secundária. Quando a primária deixa de responder ao cliente, após um período de tempo específico (*timeout*), considera-se que ocorreu uma falta no sistema, sendo essa falta tolerada devido à presença da réplica secundária, que assume a função de responder ao cliente. Na replicação passiva não é possível tolerar faltas bizantinas, pois a réplica primária poderia, por exemplo, realizar o armazenamento primário incorretamente, mas responder ao cliente que a operação ocorreu com sucesso. Ao mesmo tempo, essa réplica primária poderia não encaminhar a solicitação à réplica secundária, deixando

a réplica secundária desatualizada; quando a réplica primária manifestar comportamento incorreto, o cliente acionaria a réplica secundária, que estaria desatualizada e não poderia responder corretamente. Por isso no modelo de replicação passiva apenas faltas de parada são toleradas, e apenas em condição de tempo síncrono.

Na replicação ativa, todas as réplicas são solicitadas a executar o pedido do cliente. Dessa maneira, é possível que o cliente consiga executar a operação se ao menos uma réplica funcionar corretamente. Esse fato traduz-se na definição de que na replicação ativa as faltas de parada são toleradas com a existência de $f + 1$ réplicas, sendo $f$ o número máximo de faltas toleradas, independentemente do modelo de tempo (síncrono ou assíncrono).

Para uma tolerância a faltas mais robusta - a tolerância a faltas bizantinas - a quantidade necessária de réplicas varia em função do modelo de tempo considerado. Para sistemas síncronos, é possível tolerar faltas bizantinas dispondo de $2f + 1$ réplicas. Essa garantia existe porque caso uma réplica não responda no tempo máximo especificado (*timeout*), aquela réplica será considerada incorreta. Se todas as réplicas responderem no tempo máximo, mas alguma réplica apresentar valor diferente das demais, é possível aplicar um critério de maioria para verificar qual a réplica que está apresentando valor incorreto. Isso deve-se ao fato de que a partir de $2f + 1$ réplicas é possível obter $f + 1$ respostas iguais provenientes de réplicas diferentes. Por exemplo, considerando três réplicas no sistema ($f = 1$), é possível obter duas respostas iguais, eliminando a necessidade de a terceira resposta estar correta.

No caso de replicação ativa e sistema assíncrono (ou parcialmente síncrono), serão necessárias $3f + 1$ réplicas para que o sistema tolere faltas bizantinas. Isso deve-se ao fato de que, da mesma maneira que no modelo síncrono, deve haver uma maioria de respostas corretas ($f + 1$) para garantir a correção da resposta. Subtraindo estas respostas corretas ($f + 1$) do total de réplicas ($3f + 1$), restam $2f$ réplicas, das quais $f$ podem ter comportamento incorreto e $f$ podem apresentar atraso de rede (alta latência de resposta). Independente da origem do problema, o progresso do protocolo deve ocorrer. Por isso, a tolerância a faltas bizantinas inclui como possibilidade o mau funcionamento de réplicas, bem como o mau funcionamento da rede, e apesar destes problemas, fornece a resposta correta ao cliente, respeitando o limite de faltas toleradas no sistema.

**Replicação de máquinas de estado:** os protocolos que permitem tolerar faltas bizantinas geralmente consideram dois tipos de estratégias para realizar a replicação: os quóruns [26] e a replicação de máquinas de estado (RME) [32]. Na RME, um cliente envia a solicitação às réplicas e ocorre uma comunicação entre essas réplicas com o objetivo de estabelecer uma ordem de execução das solicitações. Considerando que vários clientes estarão efetuando requisições ao sistema, é necessário que as réplicas estabeleçam uma ordem na execução dos pedidos, de tal maneira que todas as réplicas executem os pedidos na mesma ordem, mantendo assim o estado de todas as réplicas da mesma maneira. O protocolo PBFT [12] é um dos mais conhecidos na realização de tolerância a faltas bizantinas utilizando RME, e seus passos de comunicação estão ilustrados na figura 1.3. O cliente envia uma solicitação *R* a todas as réplicas (etapa *I*); uma das réplicas é definida previamente como líder, cuja função é ordenar a solicitação *R* em relação às anteriores e reenviar esta solicitação às demais réplicas, informando uma ordem de execução *i* de-

terminada para aquela solicitação *R* (etapa *II*, denominada *pré-prepare*). No próximo passo, as réplicas que receberam a mensagem do líder trocam informações entre si, com o objetivo de certificar-se de que as outras réplicas também foram orientadas pelo líder a executar a solicitação *R* seguindo a ordem de execução *i* (etapa *III*, denominada *prepare*). Neste momento, pode-se verificar que uma réplica apresentou comportamento incorreto, ou o líder atuou de forma incorreta (por exemplo, enviando o mesmo pedido com ordens diferentes para réplicas diferentes). No caso de comportamento incorreto do líder, as réplicas que percebem esse fato solicitam uma troca de visão, operação que tem por objetivo eleger uma outra réplica para exercer o papel de líder. Quando ocorre essa mudança de visão, o protocolo precisa ser reiniciado, e nesse contexto a etapa *IV*, denominada *commit*, tem por objetivo organizar a execução de pedidos definidos antes e depois de uma possível troca de visão.

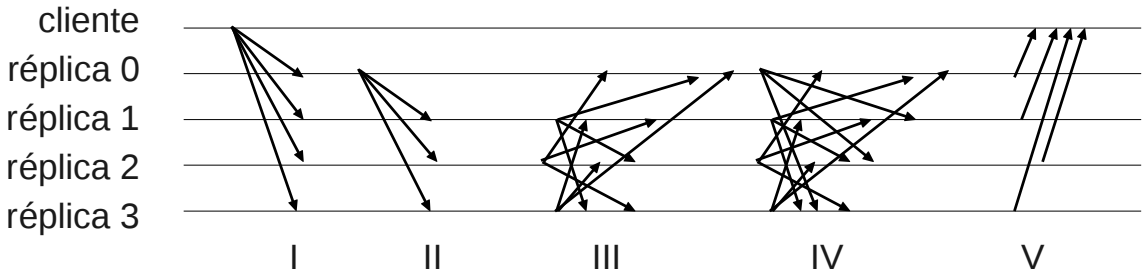



Figura 1.3: O protocolo PBFT [11].

**Quóruns:** os protocolos RME podem ser utilizados para realizar armazenamento de dados, entretanto essa técnica requer que as réplicas sejam entidades ativas, no sentido de comunicar-se com outras réplicas, verificar se as respostas das outras réplicas configuram maioria de respostas iguais, etc. A abordagem de quóruns [26] é capaz de trabalhar com réplicas passivas, ou seja, réplicas que não possuem capacidade de processamento, mas apenas realizem operações simples de ler e escrever dados. Inicialmente, será considerado o caso de faltas de parada, tal que $n = 2f + 1$ réplicas são suficientes para tolerar este tipo de falta; o sistema considerado será assíncrono. De maneira geral, a utilização de quóruns em sistemas de armazenamento de dados tolerantes a falta de parada ocorre da seguinte maneira: o cliente solicita a $2f + 1$ réplicas que armazenem um dado; quando $f + 1$ réplicas confirma a execução do armazenamento, o cliente obtém a confirmação da operação. Quando uma leitura de dados é solicitada por um cliente, novamente $2f + 1$ réplicas são questionadas sobre o valor do dado, e quando $f + 1$ réplicas respondem, o cliente pode obter o valor do dado. Considere que o sistema não suporta concorrência de escrita, ou seja, apenas um cliente por vez realizará a escrita de um dado. Além disso, os dados são versionados, ou seja, cada dado escrito possui um número de versão incremental, cujo maior valor define a versão mais recente do dado. Dessa maneira, a tabela 1.1 ilustra o desenvolvimento de solicitações de escrita de dados em um objeto. Considere a escrita de dados contendo os parâmetros (*valor*, *versão*), onde *valor* é o valor a ser escrito e *versão* é a versão do *valor*. Inicialmente, todas as réplicas possuem um valor nulo ($\perp$) no objeto.

| operação | $R_A$ | $R_B$ | $R_C$ |
|---|---|---|---|
| | $\perp$ | $\perp$ | $\perp$ |
| c1, write | (X,1) | (X,1) | |
| c2, write | | (Y,2) | (Y,2) |

Tabela 1.1: Operações de escrita em um sistema de armazenamento usando quóruns.

O primeiro cliente escreveu o valor $X$, sob um número de versão 1, e as réplicas $R_A$ e $R_B$ executaram esse pedido; a réplica $R_C$ ainda não executou o pedido por motivo de atraso de rede. Um segundo cliente escreveu o valor $Y$, sob o número de versão 2, sendo atendido pelas réplicas $R_B$ e $R_C$. Um terceiro cliente que deseja ler o valor atual do objeto armazenado poderá obter dois conjuntos de respostas provenientes das réplicas: $(X,1),(Y,2)$, das réplicas $R_A,R_B$ e $R_A,R_C$, ou o valor $(Y,2)$ das réplicas $R_B,R_C$. Para definir a resposta mais recente, observa-se o maior valor no número de versão. No primeiro caso, o valor $Y$ será decidido como o valor mais recente, pois possui o maior número de versão. No segundo caso, como os números de versão são iguais, o critério de maior valor não tem efeito e o resultado retornado também será o valor $Y$. A propriedade de interseção dos quóruns garante o correto funcionamento apesar de possíveis falhas de parada, como as falhas consideradas no exemplo. Um dos maiores desafios na detecção de falhas de parada em sistemas assíncronos é diferenciar réplicas lentas de réplica que sofreram uma parada total. Por isso, réplicas lentas são consideradas faltosas em protocolos tolerantes a faltas que trabalham em redes assíncronas.

A interseção dos quóruns garante que sempre uma das réplicas vai possuir o valor de versão mais recente. Entretanto, se a réplica que possuir o valor mais recente for uma réplica maliciosa, a resposta retornada ao cliente poderá ser incorreta. Dessa maneira, para tolerar faltas bizantinas em sistemas que utilizam quóruns é necessário adicionar mais uma quantidade de $f$ réplicas no sistema, garantindo então que mesmo que uma réplica apresente valores incorretos, ainda haverá outra réplica que possuirá o valor correto mais recente. Uma réplica maliciosa pode apresentar valores incorretos de duas maneiras: com versões antigas de dados, ou com versões futuras de dados. No primeiro caso, as réplicas adicionais inseridas carregam a versão mais recente do dado (a quantidade de réplicas aumenta de $2f+1$ para $3f+1$; o quórum de gravação aumenta de $f+1$ para $2f+1$), tornando portanto sem efeito o valor antigo de versão apresentado pela réplica incorreta. Para evitar que réplicas maliciosas apresentem valores de versões futuras de dados, utiliza-se a estratégia de fazer com que os escritores de dados compartilhem uma chave criptográfica, tornando assim impossível para a réplica forjar um número futuro de versão de dado. Uma outra maneira de não permitir valores maiores incorretos é utilizar a técnica de *non-skipping timestamps* [4], onde os valores obtidos são ordenados e considera-se como correto e maior valor o $(f+1)$-ésimo valor.

**Paxos:** por fim, é importante apresentar o Paxos [24], que pode ser considerado uma família de protocolos, dada a grande quantidade de implementações que seguem sua definição. O Paxos é um algoritmo que tem por objetivo resolver o problema de consenso, que consiste em, dado um conjunto de participantes, decidir por um valor, a partir de propostas de um ou mais participantes. O Paxos, em sua forma mais básica, baseia-se na lógica de que participantes devem propor valores até que um valor seja aceito por uma

maioria de participantes. Os participantes deste processo incorporam papéis denominados: cliente, proponente, eleitor e aprendiz. A figura 1.4 ilustra as etapas de comunicação entre os participantes. O cliente faz uma solicitação ao sistema (etapa *I*); o proponente recebe a solicitação do cliente, atribui um valor à solicitação e encaminha aos eleitores a proposta que sugere o valor atribuído (etapa *II*). Pode-se considerar um sistema onde qualquer participante pode atuar como proponente, ou pode-se realizar uma eleição para definir um líder de tal forma que apenas o líder será proponente. A eleição também é uma espécie de consenso, e o Paxos pode ser utilizado para tal fim. Os eleitores recebem propostas do proponente, verificam se irão aceitar ou não a proposta, e respondem ao proponente caso possam aceitá-la (etapa *III*). O proponente, se conseguir obter respostas dos eleitores de tal forma que seja obtida uma maioria de votos apontando o mesmo valor, informa aos eleitores que foi possível decidir por um valor (etapa *IV*). Os eleitores, ao receberem a informação de que foi possível decidir por um valor, verificam se aquela decisão ainda é válida para eles; em caso positivo, retornam uma confirmação ao proponente informando que o valor foi aceito por eles, e repassam o valor decidido aos aprendizes (etapa *V*). Os aprendizes, após receberem um valor definido (e aceito) pelos eleitores, realizam a tarefa solicitada e então retornam o resultado ao cliente (etapa *VI*).

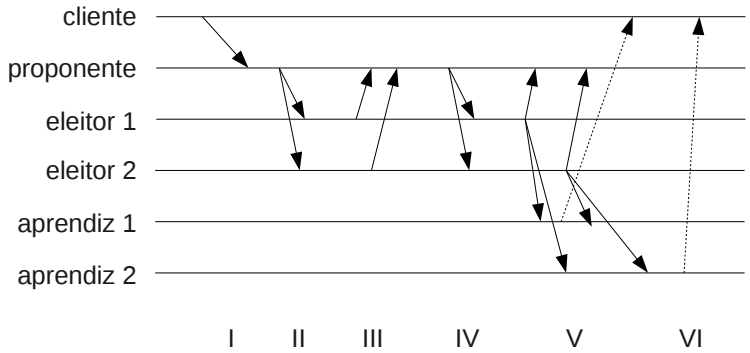



Figura 1.4: Etapas do algoritmo Paxos básico [24].

Existem alguns detalhes que fazem do Paxos um protocolo altamente flexível e versátil. O eleitor, quando recebe uma proposta de número *n* (etapa *II*), verifica se *n* é maior do que o maior número de proposta recebido até então. Caso esta verificação se confirme, o eleitor retorna ao proponente uma mensagem que representa uma ”promessa” de que ignorará futuras propostas com números menores do que *n* (etapa *III*). Informações sobre ”promessas” anteriormente feitas ou propostas anteriormente aceitas também são enviadas ao proponente, juntamente com a ”resposta-promessa”. Quando o proponente recebe uma quantidade de respostas dos eleitores de tal forma que essa quantidade configura uma maioria de respostas, o proponente pode definir o número de sua proposta; geralmente essa decisão usa o critério do maior valor. Após decidir o número da proposta, o proponente envia aos eleitores uma mensagem (etapa *IV*) informando que a proposta foi aceita por uma maioria de eleitores, e informa também qual foi o valor definido para a proposta. Quando o eleitor recebe uma mensagem de proposta aceita, ele deverá aceitar essa decisão somente se não prometeu a ninguém que iria aceitar somente propostas com valor maior que *n*. Caso esta situação se confirme, o eleitor registra o valor definido e envia aos aprendizes a solicitação e o valor aceito (etapa *V*). Essa mensagem é também enviada ao proponente.

O algoritmo Paxos é bastante utilizado em provedores de nuvens. A nuvem Microsoft Azure menciona que ”Um algoritmo de consenso, semelhante ao Paxos, é utilizado para manter as réplicas consistentes.”[12]. É possível identificar características do Paxos na descrição detalhada de como a nuvem Azure proporciona tolerância a faltas em seus bancos de dados:

> O banco de dados SQL da Microsoft Azure mantém múltiplas cópias de todos os bancos de dados em diferentes nós localizados em subsistemas físicos independentes, como diferentes servidores e roteadores. A todo tempo, os bancos de dados são mantidos em três réplicas: uma primária e duas secundárias. O mecanismo de replicação usa um esquema de confirmação baseado em quóruns onde dados são escritos na réplica primária e em uma réplica secundária antes que a transação possa ser confirmada. Se a réplica primária falhar, a falha é detectada e o sistema aciona a réplica secundária. Em caso de perda física da réplica, o sistema cria uma nova réplica automaticamente. Portanto, existem sempre ao menos duas réplicas de banco de dados que possuem consistência transacional.

## 1.3. Segurança: confidencialidade e fragmentação

Esta seção contém técnicas utilizadas para prover confidencialidade aos dados. É apresentada uma alternativa ao gerenciamento de chaves tradicional: o compartilhamento secreto (*Secret Sharing* - SS), onde é possível distribuir uma chave entre participantes, ao invés de manter a custódia da chave sob um elemento único ou centralizado. Será também abordada a técnica de criptografia baseada em identidade (*Identity Based Encryption* - IBE), que possibilita a construção de uma chave pública a partir de um conjunto de caracteres conhecido, como um nome ou um e-mail.

Esta seção inclui também a fragmentação de dados, uma operação que auxilia na proteção do dado com a estratégia de fragmentação, replicação e espalhamento (*Fragmentation, Replication and Scattering* - FRS), bem como auxilia na economia de espaço de armazenamento por meio da técnica de *erasure codes*, amplamente utilizada por provedores de nuvens comerciais. O particionamento de dados será também apresentado como estratégia para permitir um melhor desempenho de aplicações, bem como melhorar a capacidade de tolerância a faltas dos sistemas. Por fim, serão apresentados alguns tipos de consistência e suas características em relação aos dados considerados.

### 1.3.1. Compartilhamento secreto

O compartilhamento secreto [33] (*secret sharing - SS*) é uma técnica que permite codificar um segredo e distribuir as partes (também chamadas compartilhamentos) entre participantes, de maneira que seja necessária uma quantidade mínima de compartilhamentos para que o segredo possa ser restaurado. Geralmente especifica-se um esquema do tipo $(m,k)$, onde $m$ é o número total de partes e $k$ é o número mínimo de partes necessárias para reconstruir o segredo.

[12] http://azure.microsoft.com/blog/2012/07/30/fault-tolerance-in-windows-azure-sql-database/

Em sistemas de armazenamento, é comum a utilização da técnica de SS para gerenciar uma chave criptográfica por meio da distribuição da chave em forma de compartilhamentos. Assim, para decifrar um conteúdo armazenado, é necessário reunir uma quantidade mínima de provedores que possuam compartilhamentos, possibilitando a restauração da chave simétrica que poderá decifrar o conteúdo ofuscado.

Como exemplo, seguem comandos para criar compartilhamentos secretos no Linux Debian 7. O programa selecionado foi a biblioteca *libgfshare-bin*. Considere o arquivo *senha.txt*, listado abaixo, contendo o texto *senha secreta*. Os compartilhamentos serão criados no esquema $(3,2)$: dentre três compartilhamentos, quaisquer dois deles são suficientes para restaurar o segredo. Seguem os comandos:

```
$ ls -l
-rw-r--r-- 1 friend friend 14 Sep  2 09:07 senha.txt
$ gfsplit -n 2 -m 3 senha.txt
$ ls -l
-rw-r--r-- 1 friend friend 14 Sep  2 09:07 senha.txt
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.014
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.104
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.233
```

Os compartilhamentos possuem o mesmo tamanho do segredo[13], porém seu conteúdo não tem qualquer característica ou proximidade com o segredo. Para restaurar o conteúdo, bastam quaisquer dois compartilhamentos. Após a restauração, pode-se averiguar que não há diferenças entre o arquivo restaurado e o arquivo original.

```
$ gfcombine -o senhaRestaurada.txt senha.txt.014 senha.txt.104
$ ls -l
-rw-r--r-- 1 friend friend 14 Sep  2 10:42 senhaRestaurada.txt
-rw-r--r-- 1 friend friend 14 Sep  2 09:07 senha.txt
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.014
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.104
-rw-r--r-- 1 friend friend 14 Sep  2 10:40 senha.txt.233
$ diff senha.txt senhaRestaurada.txt
$
```

É importante mencionar que os compartilhamentos são restaurados a despeito de corrupções que possam ter sido realizadas nos mesmos. Assim, para garantir a integridade, é necessário proteger o compartilhamento, por exemplo, com assinatura digital. Outra maneira é dispor de uma quantidade suficiente de compartilhamentos de tal forma que seja possível restaurar corretamente o segredo mesmo que uma quantidade de compartilhamentos esteja corrompida. No sistema de armazenamento Belisarius [27], a garantia de integridade dos compartilhamentos é realizada exigindo-se que o sistema contenha $2f+2$ compartilhamentos, de forma que seja sempre possível verificar a correção de $f+2$ compartilhamentos. A figura 1.5a mostra o caso onde os três compartilhamentos gerados estão corretos, e a combinação entre quaisquer dois compartilhamentos irá gerar o segredo correto. Já no caso da figura 1.5b, existe um compartilhamento incorreto, fazendo

[13] Apesar de o texto ser formado por 13 caracteres, é possível ver na listagem o tamanho 14. O fato é que existe o caractere de término de linha, ou *carriage return*, de código igual a $0A$, no Linux; no Windows, o término de linha é representado por dois caracteres adicionais: *line feed* e *carriage return*, respectivamente, $0D$ e $0A$, ou \n e \r.

com que seja possível restaurar o segredo de três maneiras diferentes. Neste caso, não é possível identificar qual é o compartilhamento incorreto. Dispondo de $2f + 2$ compartilhamentos, conforme mostra a figura 1.5c, é possível obter $f + 2$ alinhamentos (conjuntos de $f + 1$) de compartilhamentos que levarão ao mesmo segredo; assim, é possível identificar qual é o compartilhamento incorreto.

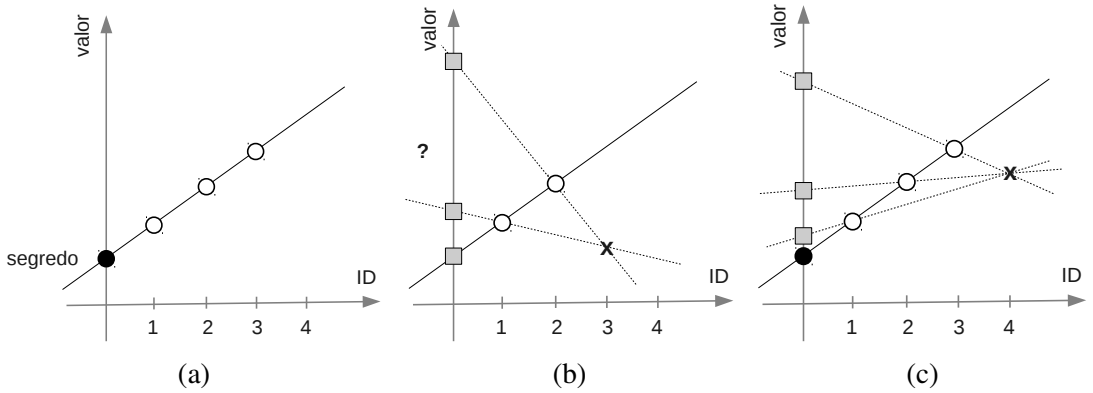



Figura 1.5: (a) compartilhamentos corretos; (b) um compartilhamento incorreto, não é possível saber qual é o segredo correto; (c) é possível identificar qual o compartilhamento incorreto [27].

### 1.3.2. Criptografia Baseada em Identidade

A criptografia baseada em identidade (Identity Based Cryptography - IBE) foi proposta por Shamir [34] com o objetivo de se utilizar conjuntos de caracteres, denominados identidades, como chaves públicas. Desta forma, não haveria a necessidade de se calcular aleatoriamente um número e vinculá-lo a uma pessoa por meio de um certificado digital. Isto significa que pode-se obter a chave pública de um indivíduo sem consultar nenhum repositório ou verificar certificados, basta ter o conhecimento do identificador relacionado. Um exemplo seria a utilização de um e-mail para a chave pública de um indivíduo.

Entretanto, a proposta de Shamir concebia somente a assinatura baseada em identidade e não havia como utilizá-la nas aplicações reais para o ciframento baseado em identidade. Em 2001 Boneh e Franklin [8] propuseram o primeiro esquema viável matematicamente de IBE, fazendo com que houvesse uma renovação no interesse dos pesquisadores e da indústria por novas propostas de uso do IBE. A proposta de Boneh baseia-se no uso do emparelhamento bilinear e possui as mesmas características da proposta de Shamir.

A IBE baseia-se no uso de uma Autoridade de Confiança (AC) para que esta emita as chaves privadas referentes às identidades utilizadas como chaves públicas. Dessa forma, um usuário que tem um identificador *ID* deve se comunicar com o sua AC e se autenticar. Caso essa autenticação seja feita de maneira correta, a AC envia por meio de um canal seguro a chave privada *CPriv* relacionada com o identificador *ID*. A IBE pode ser descrita com base nos seguintes passos, segundo Boneh:

- **Inicialização do Sistema:** Tem como entrada um parâmetro de segurança e retorna um conjunto de parâmetros públicos e uma chave mestra. Entre os parâmetros

públicos, deve-se conter a descrição do espaço finito que será utilizado para uma mensagem e uma descrição do espaço finito de um texto cifrado. Os parâmetros públicos serão publicamente divulgados enquanto a chave mestra será guardada e conhecida apenas pela AC.

- **Extração da Chave Privada:** Tem como entrada o conjunto de parâmetros públicos, a chave mestra e um identificador arbitrário *ID* e retorna uma chave privada. O identificador *ID* é um conjunto de caracteres arbitrário que se utiliza como chave pública.
- **Cifragem:** Tem como entrada os parâmetros públicos, o identificador *ID* e uma mensagem. O método retorna um texto cifrado.
- **Decifragem:** Recebe-se como entrada um texto cifrado e uma chave privada. O método retorna o texto em claro.

### 1.3.3. Fragmentação de dados

Considerando os sistemas de armazenamento de dados, a replicação integral de um dado implica em custos proporcionais à quantidade de réplicas utilizadas. Além disso, problemas comuns como uma pequena corrupção em parte do dado tornam a simples replicação custosa, visto que essa corrupção pode comprometer todo o restante da informação. Dessa maneira, foram criadas técnicas de fragmentação de dados, com o objetivo de permitir um isolamento de partes do dado e consequentemente o tratamento diferenciado de cada parte, ao invés de tratar o dado como um todo. Nesta seção duas estratégias serão apresentadas: a fragmentação com replicação e espalhamento, e a fragmentação com redundância.

#### 1.3.3.1. Fragmentação, replicação e espalhamento

Um dos primeiros trabalhos que sugeriram a fragmentação de dados foi o artigo de Fraga [17], que estabeleceu o termo ”tolerância a intrusão”. A seguir, Deswarte [15] utilizou a técnica em um sistema juntamente com estratégias para o gerenciamento de acesso, consolidando a técnica FRS (*fragmentation, replication and scattering*). Os dados considerados pelo FRS são arquivos. De uma maneira geral, o processo ocorre conforme a figura 1.6. O arquivo original é convertido em fragmentos, cada fragmento é replicado e a seguir os fragmentos são espalhados pelas réplicas. As questões de quantidades de fragmentos e réplicas tem sido investigadas deste então, no sentido de definir quais os valores ideais para cada nível de tolerância desejado.

Em detalhe, conforme mostra a figura 1.7a, um arquivo é fragmentado em páginas (no artigo esta operação é denominada particionamento); eventualmente é necessário adicionar um complemento ao arquivo (*padding*) (etapa *I*), para que as páginas tenham tamanhos iguais (etapa *II*). A seguir, conforme ilustra a figura 1.7b, cada página é cifrada com uma chave simétrica (etapa *III*) e então a página cifrada é fragmentada (etapa *IV*). A distribuição dos fragmentos deve ocorrer em ordem aleatória para diminuir a chance de observadores na rede (ataques *eavesdropping*) identificarem a ordem dos fragmentos durante o envio.

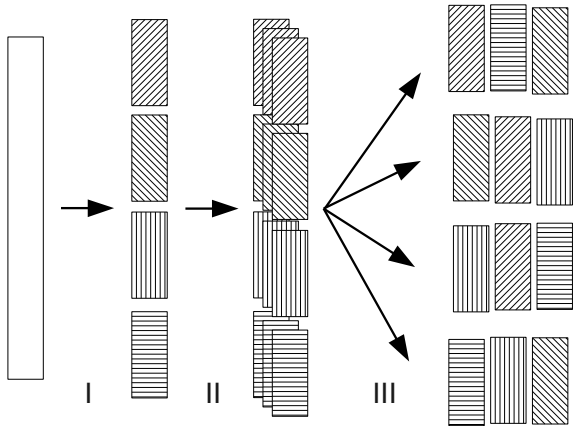



Figura 1.6: Visão geral do processo FRS [15].

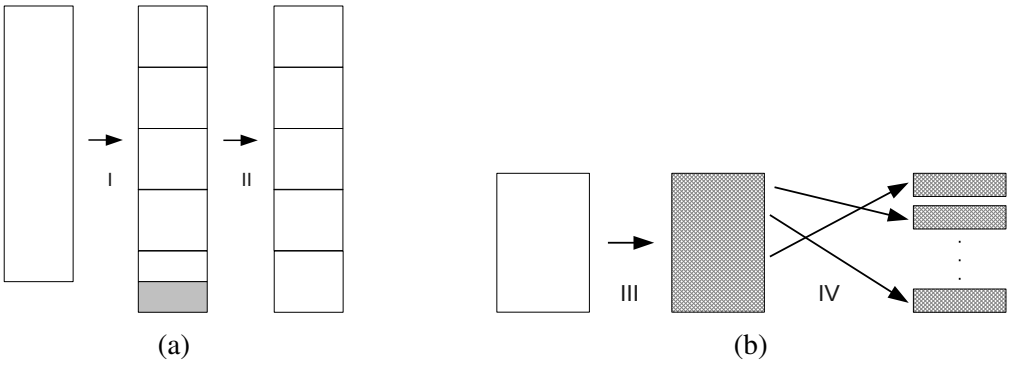



Figura 1.7: (a) Preenchimento complementar para igualar o tamanho da última página; (b) Cifragem e fragmentação da página [15].

A técnica FRS possui como premissa a replicação de fragmentos para garantir a redundância do dado. É possível fazer várias construções de acordo com o número de replicação dos fragmentos e o número de servidores de armazenamento disponíveis. Por exemplo, considere a fragmentação de um dado em três fragmentos, conforme ilustração na figura 1.8. O dado é particionado (1); a seguir, é inserido um complemento para igualar o tamanho da última página às demais páginas(2). Cada página então é cifrada (3), fragmentada (4) e replicada (5). Os fragmentos podem ser agrupados dois a dois e enviados a três provedores diferentes (6). O conjunto de fragmentos obtidos de quaisquer dois provedores é suficiente para restaurar o dado.

A técnica FRS garante confidencialidade ao dado; entretanto, possui um fator de ocupação de espaço igual ao dobro do espaço requerido pelo dado. Esta quantidade equivale a uma replicação integral. A próxima técnica (fragmentação com redundância) realiza operação de fragmentação semelhante à FRS, ocupando bem menos espaço, porém sem garantir confidencialidade ao dado.

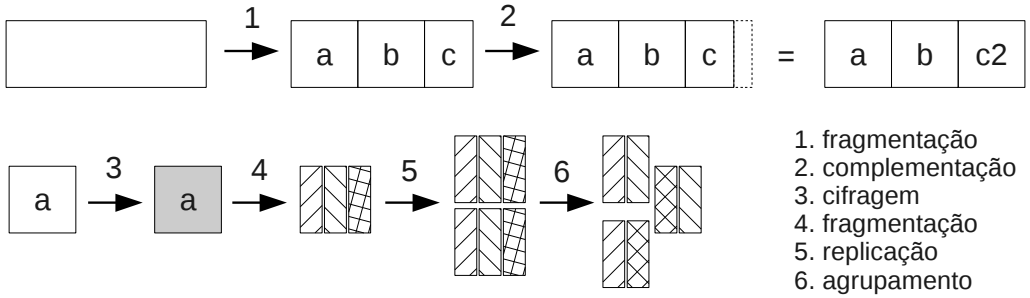



Figura 1.8: Exemplo de aplicação da técnica FRS: replicação em três provedores.

#### 1.3.3.2. Fragmentação com redundância

Os códigos de correção de erro (CRC)[14], foram criados para tolerar pequenas corrupções em dados. De forma geral, algumas informações eram agregadas ao dado original, tornando possível a recuperação de perdas de dados. Esta técnica era inicialmente utilizada para corrigir perdas em sinais de comunicação; posteriormente, foi adaptada para ser utilizada em armazenamento de dados, permitindo a correção de dados em repouso.

A correção de erros foi utilizada para prover redundância a dados, em uma técnica denominada RAID [29] (*Redundant Array of Inexpansive Disks*), que tinha por objetivo permitir a realização de armazenamento confiável utilizando dispositivos econômicos de baixa confiabilidade. Os primeiros discos rígidos não eram dispositivos confiáveis; logo, o RAID foi criado para permitir o uso de vários discos, de modo que a substituição de discos pudesse ser feita de maneira a não interromper o funcionamento do sistema nem implicar em perda de dados.

A partir da fragmentação de dados realizada pelo RAID, surgiu a ideia de distribuir fragmentos em nós de uma rede de computadores. A técnica de dispersão de dados [30], também chamada de fragmentação com redundância, foi criada para prover correção de erros e redundância a dados em dispositivos distribuídos por uma rede de computadores. O procedimento consiste em dividir o dado em $k$ fragmentos e criar $m$ fragmentos adicionais de tal forma que qualquer conjunto de $k$ fragmentos possibilite a reconstrução do dado. A configuração dos parâmetros geralmente é definida por $(k,m)$, sendo que $k$ é o número de fragmentos originais e $m$ é o número de fragmentos adicionais ou redundantes.

A utilização de fragmentação com redundância adiciona tolerância a faltas ao armazenamento de dados. Entretanto, é importante notar que esta técnica de distribuição de dados apenas tem sentido quando são utilizados no mínimo três provedores de armazenamento. A figura 1.9 ilustra o processo de codificação dos fragmentos e de restauração do dado, para o caso de uma codificação $(2,1)$. Inicialmente, o dado é dividido em dois fragmentos (I); a seguir, um fragmento redundante é criado (II), a partir do conteúdo dos outros dois fragmentos. Os três fragmentos são então distribuídos em diferentes provedores de armazenamento (III). Para restaurar o dado, dois fragmentos quaisquer são obtidos dos provedores (IV), sobre os quais é realizado um processamento para então recuperar o dado original (V).

[14]Richard W. Hamming, 1950. Error detecting and error correcting codes.

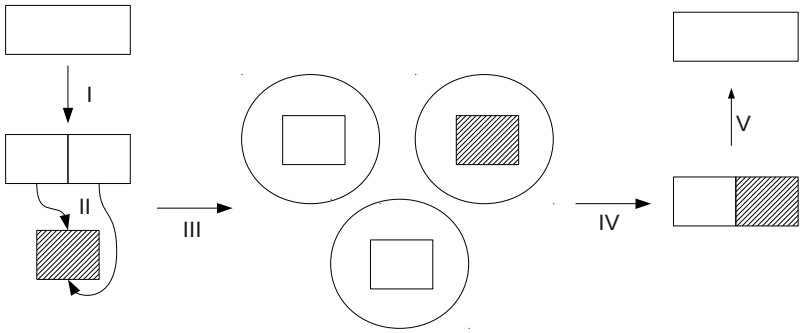



Figura 1.9: Codificação e restauração de dado em uma operação de *erasure code* (2,1).

Para fins de demonstração, seguem exemplos da fragmentação com redundância utilizando a ferramenta Jerasure[15], versão 1.2. O sistema operacional utilizado foi o Debian 7.0. Considerando o diretório onde o arquivo *.tar* foi salvo, seguem de forma resumida os comandos para a instalação deste software:

```
tar -xvf Jerasure-1.2A.tar
cd Jerasure-1.2A/Examples
make
```

A partir de então, os comandos *encoder* e *decoder* realizam as tarefas de fragmentar e restaurar o dado, respectivamente. Considere um diretório contendo os seguintes arquivos:

```
$ ls -lh
total 40M
-rwxr-xr-x 1 friend friend  69K Sep  1 10:21 decoder
-rw-r--r-- 1 friend friend  13K Sep  1 10:24 documento.pdf
-rwxr-xr-x 1 friend friend  69K Sep  1 10:21 encoder
-rw-r--r-- 1 friend friend 3.4M Jun 16 20:44 slides.pdf
-rw-r--r-- 1 friend friend  37M Sep  1 10:23 video.mp4
```

Para realizar a criação dos fragmentos, será utilizado o comando *encoder*. Os parâmetros 2 e 1 são relativos à quantidade de fragmentos originais e à quantidade de fragmentos redundantes, respectivamente. Os demais parâmetros são específicos do Jerasure; para verificar detalhes, consulte a documentação disponível no relatório técnico[16] da biblioteca Jerasure. Após a codificação, é criado o diretório *Coding*, que contém os fragmentos e metadados sobre a operação:

```
$ ./encoder slides.pdf 2 1 reed_sol_van 8 1024 1000
$ ls -lh Coding/
total 5.1M
-rw-r--r-- 1 friend friend 1.7M Sep  1 10:40 slides_k1.pdf
-rw-r--r-- 1 friend friend 1.7M Sep  1 10:40 slides_k2.pdf
-rw-r--r-- 1 friend friend 1.7M Sep  1 10:40 slides_m1.pdf
-rw-r--r-- 1 friend friend   54 Sep  1 10:40 slides_meta.txt
```

[15]Disponível em http://web.eecs.utk.edu/ plank/plank/www/software.html.

[16]http://web.eecs.utk.edu/ plank/plank/papers/CS-08-627.pdf

Note que os fragmentos possuem tamanho igual à metade do arquivo original. Isto deve-se ao fato de que o tamanho dos fragmentos é estabelecido em função do número de fragmentos originais especificado (no exemplo, $k = 2$). O tamanho dos fragmentos, portanto, é igual ao tamanho do arquivo original divido pelo parâmetro $k$. Os metadados contém informações necessárias durante a restauração do dado:

```
$ cat Coding/slides_meta.txt
slides.pdf
3534834
2 1 8 1024 65536
reed_sol_van
0
54
```

Um dos fragmentos será excluído para que o funcionamento da ferramenta seja aferido. Para restaurar o arquivo original, é utilizado o comando *decoder*. O arquivo original será restaurado dentro da pasta *Coding*:

```
$ rm Coding/slides_k2.pdf
$ ./decoder slides.pdf
$ ls -lh Coding/
total 6.8M
-rw-r--r-- 1 friend friend 3.4M Sep  1 10:44 slides_decoded.pdf
-rw-r--r-- 1 friend friend 1.7M Sep  1 10:40 slides_k1.pdf
-rw-r--r-- 1 friend friend 1.7M Sep  1 10:40 slides_m1.pdf
-rw-r--r-- 1 friend friend   54 Sep  1 10:40 slides_meta.txt
```

Pode-se verificar se o arquivo original está igual ao arquivo restaurado; no caso do comando abaixo, nenhuma saída foi mostrada, indicando que os arquivos são idênticos. Adicionalmente, podem ser obtidos resumos dos arquivos para outra verificação de integridade.

```
$ diff slides.pdf Coding/slides_decoded.pdf
$ md5sum slides.pdf
020dd875eacfed1f3a8e9da2fefb8efe  slides.pdf
$ md5sum Coding/slides_decoded.pdf
020dd875eacfed1f3a8e9da2fefb8efe  Coding/slides_decoded.pdf
```

### 1.3.4. Particionamento de dados

A replicação de dados pode ser aprimorada considerando-se características específicas das aplicações. Por exemplo, em caso de dados replicados em regiões geográficas distantes, nem sempre é necessário que os dados estejam presentes em todas as réplicas, isto é, nas réplicas que estão próximas e distantes (em termos de latência de acesso) da aplicação. Por exemplo, em um sistema de cadastro de clientes, o registro de dados dos clientes pode estar armazenado em provedores próximos às regiões nas quais os referidos clientes residem. Isto significa, por exemplo, armazenar clientes de nacionalidade Brasileira em um provedor do Brasil, e clientes de nacionalidade Japonesa em provedores do Japão. Naturalmente, os dados dos provedores podem ser replicados em outros provedores, mas a replicação pode ser feita de maneira assíncrona, como comentado na seção 1.2. Separar os dados desta maneira significa *particionar* os dados. No exemplo do cadastro de clientes, a separação de registros consiste em um particionamento horizontal de dados. No

caso deste exemplo, o critério de particionamento foi a localização geográfica do dado. O particionamento horizontal pode feito de diversas outras maneiras [31]: *round-robin*, *hashing* e faixa de valores.

Com o objetivo de distribuir as informações de forma mais homogênea, o particionamento horizontal pode seguir uma ordem denominada *round-robin*: requisições sequentes armazenam dados em partições sequentes. Considerando um sistema de requisições ordenadas (sequência linearizável) de armazenamento de dados, a primeira requisição de dados armazenará o dado na partição 0, a segunda requisição armazenará o dado na partição 1, e assim por diante. O pedido $R_i$ armazenará o dado na partição $P_{(i \ mod \ n)}$, sendo que $n$ é o número total de partições. O particionamento horizontal pode também ser definido de uma maneira fixa, com faixas de valores. Um exemplo simples consiste na regra de que clientes cujo primeiro nome inicia pela letra que está na faixa entre A e L serão armazenados em um determinado provedor (partição); e clientes de nome iniciando pela letra M a Z serão armazenados em outra partição. Ainda relativo ao particionamento horizontal, outra forma bastante utilizada de selecionar a partição na qual o dado será armazenado é através de *hashing*. Neste caso, é gerado um *hash* do dado, e de acordo com este *hash* é decidido em qual partição o dado será armazenado. Pode-se considerar que o particionamento por *hashing* é uma espécie de particionamento de faixa de valores, sendo que o valor considerado é o *hash* do dado, ao invés de algum outro campo específico.

Outra maneira de atenuar o esforço de replicação é a separação de campos específicos em um conjunto de dados. Por exemplo, o cadastro de clientes considerado poderia ter um campo chamado *CPF* que faria referência ao cadastro de pessoa física, informação vital para qualquer cidadão brasileiro. Os números de CPF do cadastro de clientes deveriam ser armazenados em provedores brasileiros, para que o acesso por brasileiros seja beneficiado em termos de latência. Este tipo de separação de dados é considerado um particionamento vertical. Campos específicos podem ser armazenados em partições diferentes.

### 1.3.5. Consistência

A consistência dos dados armazenados em um local $x$ refere-se a visão que os leitores têm do valor de $x$ quando ocorrem leituras e escritas de dados em $x$ por diversas réplicas ou processos. Lamport [22] define o nível de consistência sequencial; Tanenbaum [35] apresenta essa definição de forma pragmática: considerando que operações são executadas em diferentes réplicas, "...qualquer sequência de operações de leitura e escrita é válida, mas todas as réplicas executam a mesma sequência de leituras e escritas". A figura 1.10 demonstra exemplos com e sem consistência sequencial.

Na figura 1.10a, a réplica $P1$ escreve o valor $a$ em $x$; em seguida, $P2$ escreve o valor $b$ em $x$. As réplicas $P3$ e $P4$ lêem o valor $b$, e em um segundo momento, o valor $a$ é lido por essas réplicas. Pode-se inferir que a escrita de $b$ foi efetivada antes da escrita de $a$. Este fato não configura problema, visto que a sequência de operações observada pelas réplica $P3$ e $P4$ foi a mesma. Essa igualdade de sequência de operações entre réplicas configura a consistência sequencial. A sequência observada pelas réplicas $P3$ e $P4$ será: $W(x)b > R(x)b > Wx(a) > R(x)a$. Na figura 1.10b, as réplicas $P3$ e $P4$ observam os valores $a$ e $b$ em ordens diferentes, não possuindo portanto uma consistência sequencial.

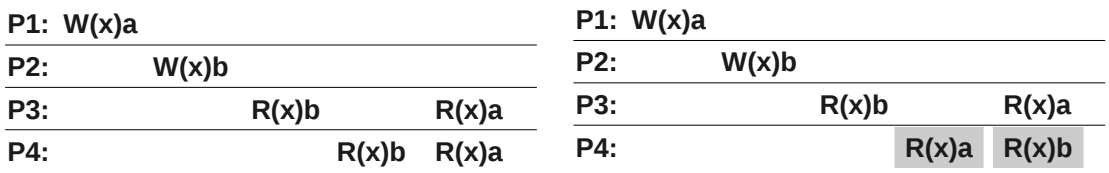



Figura 1.10: (a) Consistência sequencial; (b) Consistência não-sequencial.

A consistência causal [22] trata de operações que possuem uma relação de causalidade potencial. Tanenbaum [35] enuncia este nível de consistência de forma objetiva: "Se uma operação $b$ é causada ou influenciada por uma operação anterior $a$, então a causalidade requer que todos os executores de operações executem $a$ antes de $b$". Uma observação fundamental neste tipo de consistência é que operações de escrita concorrentes não podem estabelecer relação causal com leituras consecutivas. A figura 1.11 apresenta situações onde a causalidade é obedecida e violada, respectivamente. Na figura 1.11a, as réplicas $P1$ e $P2$ realizam escrita sobre objeto $x$. As escritas são consideradas concorrentes, pois não há relação causal entre as mesmas: nenhuma das escritas utiliza o valor anterior do objeto para a composição do novo valor que está sendo escrito. Sendo assim, como a ordem das escritas não importa, na consistência causal, a ordem das leituras também não importa. As duas sequências observadas pelas réplicas $P3$ e $P4$, respectivamente: $W(x)b > R(x)b > W(x)a > R(x)a$; e $W(x)a > R(x)a > W(x)b > R(x)b$ obedecem à relação causal entre as operações de leitura e escrita realizadas.

A figura 1.11b mostra a inserção de uma operação de leitura antes de uma escrita, na mesma réplica, o que torna a operação de escrita $W(x)b$ dependente da operação de leitura $R(x)a$; essa, por sua vez, é dependente da operação $W(x)a$. Sendo assim, a única sequência admissível que mantém a causalidade de operações é: $W(x)a > R(x)a > W(x)b > R(x)b$. Esta sequência está sendo observada pela réplica $P4$, mas a réplica $P3$ observa uma sequência que viola a causalidade entre as operações de escrita dos valores $b$ e $a$. A causalidade entre estas operações de escrita existe porque a escrita de $b$ depende potencialmente do valor de $a$, devido ao fato de que $b$ pode resultar de um processamento envolvendo o valor de $a$.

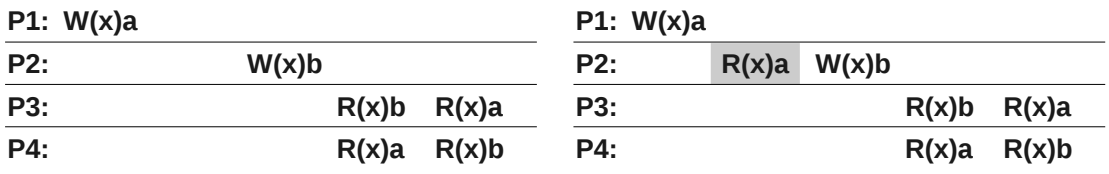



Figura 1.11: (a) Consistência causal; (b) Consistência não-causal.

Considerando a situação de concorrência entre escritas, Lamport [23] definiu três comportamentos possíveis quando leituras de dados ocorrem com escritas concorrentes:

- Na semântica segura, é possível afirmar que uma leitura não-concorrente com escritas retornará o último valor gravado; no caso de leitura simultânea com escritas,

apenas é possível afirmar que será retornado um valor válido para o local $x$. Por exemplo, se é possível armazenar em $x$ valores de 0 a 9, então é possível afirmar que o valor retornado durante uma leitura concorrente com uma ou mais escritas, na semântica segura, será igual a um valor que estará localizado entre 0 e 9.

- A semântica regular inclui as características da semântica segura, e adicionalmente define que leituras realizadas de forma concorrente com uma ou mais escritas apenas possuem duas possibilidades de valor de retorno: ou irão retornar o valor anterior ao momento em que começaram as escritas, ou irão retornar o valor de uma das escritas que está sendo realizada. É possível que diferentes leitores obtenham diferentes valores.
- Na semântica atômica estão presentes as características da semântica regular, com uma restrição adicional: as escritas concorrentes, sob o ponto de vista dos leitores, tornam-se sequenciais. Como exemplo, considere que o sistema possui um valor $a$ armazenado no local $x$. A partir do momento em que o sistema entregar para um leitor $L$ o valor $b$, dado que $b$ é um valor mais recente do que $a$, nenhum outro leitor poderá receber mais o valor $a$; todos os leitores subsequentes a $L$ receberão o valor $b$ ou outro valor mais atual do que $b$.

A necessidade de desempenho e a alta distribuição geográfica de sistemas proporcionada pelas redes de longa distância, como a Internet, motivaram a criação da semântica eventual. Vogels [36] apresenta alguns níveis níveis de consistência observados por aplicações, caracterizando por fim a consistência eventual:

- consistência forte: possui definição equivalente à semântica forte, de Lamport [23].
- consistência fraca: o sistema não garante que, após um escritor $A$ concluir a operação, leitores subsequentes observarão o valor escrito por $A$. Diversas condições precisam ser observadas para que os leitores tenham garantia de que observarão um valor atual. Há um conceito de janela de inconsistência que se refere a um intervalo de tempo entre o momento da última atualização e o momento no qual é possível garantir que o valor retornado para um leitor é um valor atual.
- consistência eventual: é uma consistência fraca cuja janela de inconsistência pode ser definida por características do sistema, como latência do meio de comunicação, carga do sistema, número de réplicas, etc.

O nível de consistência utilizado pelas aplicações depende dos requisitos da confiabilidade que a informação precisa ter, em relação ao tempo de propagação da informação, ou a atualidade da informação. O provedor de armazenamento em nuvem Amazon S3 utiliza consistência de dados eventual para seus dados[17], sendo este nível de consistência suficiente para o Dropbox armazenar seus dados [16] nessa nuvem de armazenamento.

#### 1.3.5.1. Teorema CAP

Dadas as diversas possibilidades de configurações que sistemas podem apresentar em relação a consistência, disponibilidade (em termos de latência) e particionamento (no sentido de desconexão de redes), foi criado o teorema CAP [9] (*Consistency, Availability and Partition*), que estabelece relações entre estes itens. Afirma-se que em sistemas

[17] http://aws.amazon.com/s3/faqs/

que compartilham dados em rede, não é possível atingir simultaneamente as máximas unidades destas grandezas. Em geral, aplicações priorizam duas características, em detrimento da terceira. Várias interpretações podem retratar a relação entre as três variáveis. Particionamento pode ser considerado como um limite de tempo de comunicação: se uma consistência não for alcançada em um determinado tempo, pode-se dizer que houve uma partição no sistema. Partições podem ser detectadas com *timeouts*; na existência de partições, o sistema pode limitar operações - por exemplo, não permitir que escritas sejam realizadas no sistema. Neste caso, a consistência está sendo priorizada. Outra possibilidade é armazenar as operações que deverão ser efetivadas quando uma posterior etapa de recuperação for executada, na ocasião do desaparecimento da partição. Assim, a disponibilidade é priorizada, permitindo que a escrita sempre possa ocorrer, permitindo no entanto que leitores nem sempre obtenham a versão mais recente do dado.

## 1.4. Estado da arte

Nesta seção serão comentados alguns dos trabalhos mais relevantes que abordam armazenamento de dados em provedores de nuvens.

### 1.4.1. SPANStore

SPANStore [37] é um sistema que permite armazenar um dado em múltiplos provedores de serviços em nuvem tendo como objetivo reduzir os custos associados à gravação e leitura do dado. A figura 1.12 apresenta a arquitetura do SPANStore. Nos *data centers* estão localizados os serviços de armazenamento e as aplicações. Cada *data center* contém uma máquina virtual (VM) contendo o SPANStore, uma VM contendo a aplicação e um serviço de armazenamento.

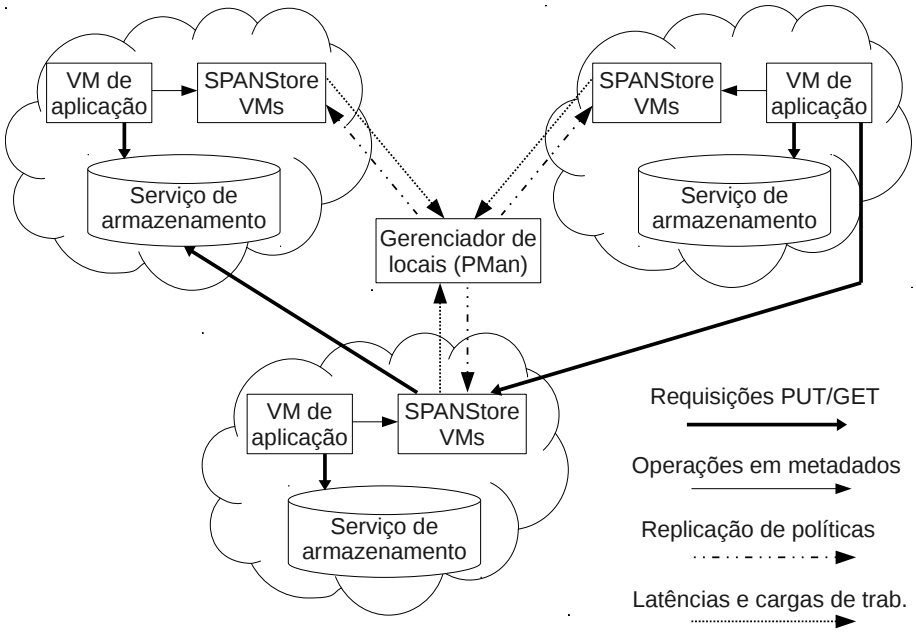



Figura 1.12: Visão geral do SPANStore [37]

A otimização de custos é alcançada com a geração de uma recomendação para a aplicação sobre quais são os provedores que devem ser utilizados para realizar o ar-

mazenamento. Para produzir a recomendação são necessários dois conjuntos de informações: informações mantidas pelo SPANStore e informações de necessidades específicas da aplicação. O SPANStore precisa ser informado com os valores praticados pelos provedores de serviço; na prática, são tabelas de preços. Esta informação pode ser obtida diretamente dos sites de provedores. Para se ter uma noção dos valores tarifados, a tabela 1.2[18] mostra informações sobre as cobranças realizadas para transferir dados a partir de instâncias de máquinas virtuais da Amazon EC2. As informações de custo utilizadas são: preço por byte armazenado, preço por PUT e GET e hora de utilização de máquina virtual (VM). Além disso, para cada par de *data centers*, obtém-se o preço ($/GB) por transferência de dados (também denominado custo da rede) que é determinado pelo preço de envio de dados a partir do *data center* de origem.

| **Destino** | | **Valor ($/GB** |
|---|---|---|
| EC2 mesma zona | endereço IP privado | 0 |
| | IP público | 0.01 |
| EC2 em outra região AWS | | 0.02 |
| Para a Internet | Primeiro GB/mês | 0 |
| | Até 10TB/mês | 0.12 |
| | Próximos 40TB/mês | 0.09 |
| | Próximos 100TB/mês | 0.07 |
| | Próximos 350TB/mês | 0.05 |
| | Próximos TB/mês | consultar |

Tabela 1.2: Valores de transferência de dados a partir de instância Amazon EC2

Ainda relativo ao conjunto das informações mantidas pelo SPANStore, existe uma tabela interna que contém as latências de comunicação entre os *data centers* e a quantidade de acessos (*workload*) dos objetos armazenados em cada *data center*. Estas informações são coletadas por um componente do sistema nomeado *Placement Manager* (PMan), denominado Gerenciador de Locais na figura 1.12. Este componente periodicamente solicita[19] a cada *data center* as latências de comunicação entre o *data center* e os outros *data centers* com os quais ele possui comunicação, e também a frequência de acesso dos objetos na VM por diferentes clientes. Essas informações são atualizadas a cada hora. Especificamente, as informações de latência são as seguintes: medidas de *PUT*, *GET* e *pings* entre VMs e serviços de armazenamento, ou entre VMs e outras VMs em outros *data centers*. Foram utilizados *data centers* nos provedores Amazon, Microsoft Azure e Google Cloud.

O outro conjunto de informações que o SPANStore utiliza para gerar a recomendação para a aplicação é específico de cada cliente, sendo por este fornecido, e consiste nas seguintes definições: nível de consistência desejado (eventual ou forte), número máximo de falhas a tolerar, latência máxima desejada e uma fração de *requests* que devem possuir latência menor que a latência máxima especificada. A confecção da melhor recomendação pode ser considerada como um problema de otimização, e o SPANStore

[18] Valores anotados em 22/08/2014, para um *data center* da Irlanda.

[19] A solicitação é feita à VM do SPANStore existente no *data center.*

utiliza um algoritmo que utiliza técnicas de programação inteira (também denominada programação linear) para resolver esta questão.

Quando a consistência definida pela aplicação é eventual, é possível replicar os dados de maneiras diversas, conforme mostra a figura 1.13. Os itens $A..F$ são *data centers*, o cliente está localizado no *data center A* e a operação considerada é a escrita de dados. Na figura 1.13a, a confirmação da escrita é feita apenas no *data center A*, e a atualização do dado é enviada do *data center A* para todos os outros *data centers* em segundo plano (protocolos de *gossip*). Este cenário garante menor latência, pois o *data center* mais próximo é atualizado e a resposta é retornada ao cliente, enquanto as atualizações são enviadas aos outros *data centers*. Em termos de tolerância a faltas, confirma-se que uma réplica armazenou o dado, e as outras $f$ réplicas são atualizadas assincronamente. Não é garantido que as outras $f$ réplicas terão sucesso na operação, entretanto isto é uma premissa intrínseca da semântica eventual e da consideração de que $f+1$ réplicas armazenarão o dado.

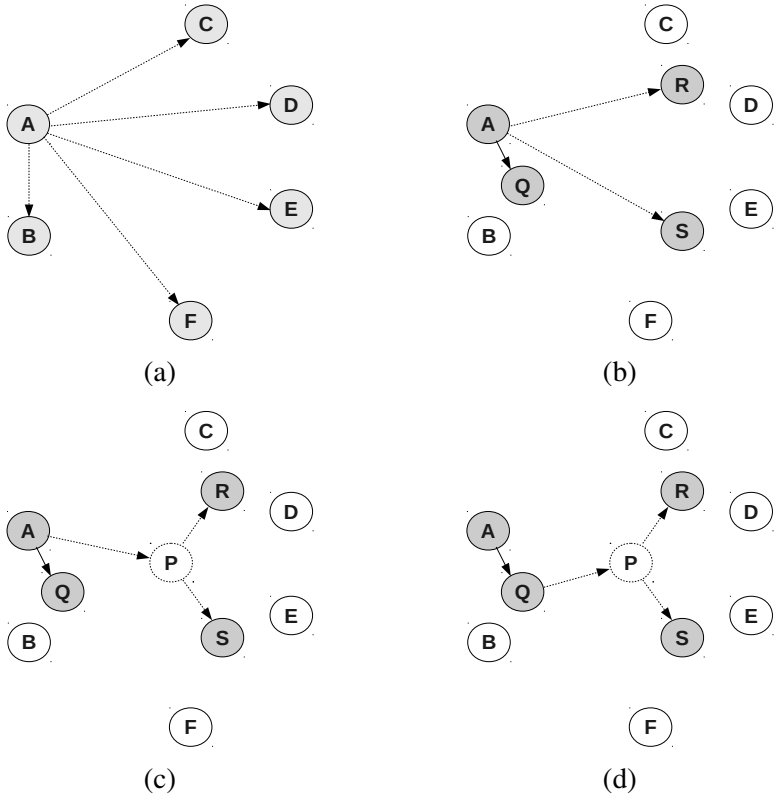



Figura 1.13: Replicação e consistência no SPANStore: (a) replicação total minimiza latência; (b) replicação parcial reduz custos; (c) delegação de escrita reduz custos de rede; (d) delegações múltiplas podem reduzir ainda mais os custos.

A figura 1.13b ilustra uma replicação parcial: nem todos os *data centers* são solicitados a armazenar o dado. Os *data centers* menos custosos são escolhidos para realizar o armazenamento de dados. O *data center* mais próximo ao cliente (neste caso, o $A$) pode sequer ser escolhido, devido às métricas de custo (será mais barato armazenar em $Q$ do que em $A$, e a latência de $Q$ está no limite definido pela aplicação). Na figura 1.13c ocorre

um encaminhamento (*relay*) de pedido de escrita: ao invés do cliente (localizado em *A*) solicitar o armazenamento diretamente aos *data centers* *R* e *S*, o pedido é encaminhado ao nó *P* para que o mesmo faça o armazenamento nos vizinhos *R* e *S*. Uma outra situação ilustrada na figura 1.13d pode ocorrer se, de acordo com os custos calculados, o nó *Q* além de armazenar o dado também possuir um custo de rede melhor do que o custo de rede entre os pontos *A* e *P*. Nesse caso, o nó *Q*, após realizar o armazenamento, encaminha o pedido ao nó *P*; o nó *P* não armazena o dado, mas apenas encaminha o pedido aos nós *R* e *S*.

As figuras 1.14, 1.15 e 1.16 ilustram[20] alguns cenários com uma série de detalhes que permitirão compreender melhor como o SPANStore toma algumas decisões. Os custos apresentados são denominados custo da rede. A figura 1.14 demonstra o armazenamento de dados utilizado quando a consistência exigida pela aplicação é forte. Neste caso, não pode ser utilizada delegação de pedidos, pois é necessário ter a confirmação de cada *data center* sobre o sucesso da operação de armazenamento. O custo de rede total neste caso será igual a $0.75/GB, correspondente ao envio do pedido para os três provedores.

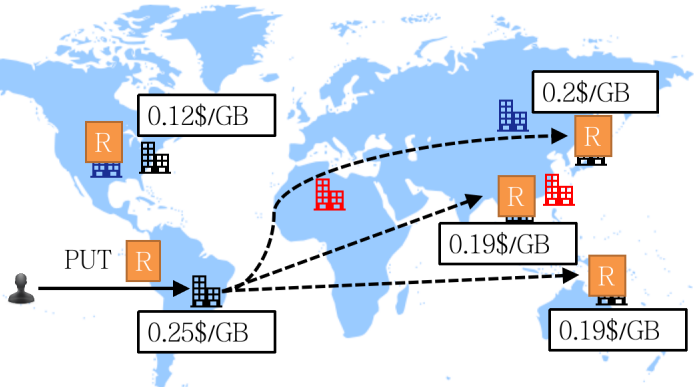



Figura 1.14: Escrita de dados, consistência forte [37].

Quando uma aplicação requer consistência eventual pode ser mais econômico delegar a transferência de dados a outro *data center* que possua menores taxas de envio de dados. Por exemplo, na figura 1.15 o pedido foi delegado ao provedor que possui custo de rede igual a $0.12/GB. Nesse caso, o valor do envio de dados para os provedores será igual a $0.25 + 3 * 0.12$, totalizando $0.61/GB, sendo este um valor menor do que $0.75/GB. Um inconveniente desta estratégia é que a transferência de responsabilidade de execução do pedido pode provocar atrasos que excederão as latências definidas no SLO (*Service Level Objective*), definido pela aplicação.

Para delegar a execução de um pedido e manter a latência dentro do SLO, pode-se solicitar a um provedor destinatário distante que realize duas tarefas: execute o pedido localmente e também solicite a outros *datacentes* que executem o pedido. Neste caso, um *data center* atua simultaneamente como provedor, quando executa o pedido, e cliente, quando solicita a outros *data centers* que executem o pedido. A figura 1.16 apresenta esse cenário; o custo de transferência será igual a $0.25 + 2 * 0.19$, totalizando $0.63/GB.

[20]Estas três figuras foram obtidas de slides dos autores do SPANStore, e estão disponíveis em http://sigops.org/sosp/sosp13/program.html

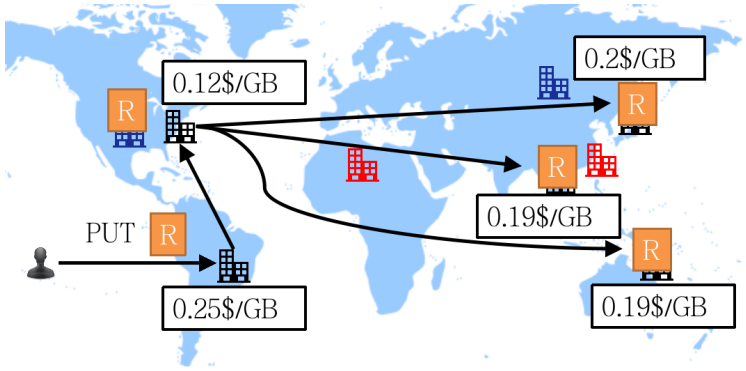



Figura 1.15: Escrita de dados, consistência fraca, *relay* da escrita [37].

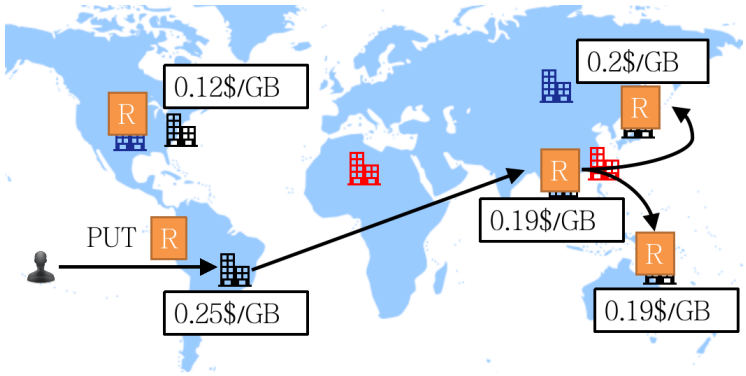



Figura 1.16: Solicitação de execução e delegação de pedido a um provedor [37].

O SPANStore apresenta considerações de custos em armazenamento nos provedores de nuvens. Ainda que exista resistência para realizar armazenamento de dados sensíveis (metadados, senhas, documentos confidenciais) em provedores de nuvens devido à falta de confiança dos clientes na garantia de confidencialidade dos seus dados, uma questão bastante relevante na utilização de serviços de armazenamento remoto é o custo. Ocorrem alterações nos valores praticados pelos provedores[21], o que pode reconfigurar o conjunto de provedores menos custosos, sob o ponto de vista do cliente. Existem ferramentas que permitem estimar gastos em provedores de nuvens[22], entretanto o diferencial do SPANStore é a consideração de itens dinâmicos como latências momentâneas e a disponibilidade de provedores no momento em que a recomendação é elaborada para a aplicação.

Sobre a tolerância a faltas, aponta-se o elemento PMan como um ponto fraco do sistema: é um elemento único (não-replicado) que, em caso de corrupção de dados (invasão maliciosa) pode apresentar respostas incorretas, afetando assim as recomendações geradas para as aplicações. Apesar de o PMan ser um elemento centralizado, ele não afeta significativamente o desempenho do sistema, pois a periodicidade com a qual o PMan é executado é grande: ele é executado a cada hora.

[21] http://techcrunch.com/2014/08/20/cloud-storage-is-eating-alive-traditional-storage/
[22] http://www.planforcloud.com

### 1.4.2. Augustus

O Augustus [28] é um sistema de armazenamento chave-valor tolerante a faltas bizantinas que proporciona escalabilidade através de particionamento e permite realizar operações complexas de uma maneira mais simples com a utilização de transações de curta duração (*minitransações*). As transações de curta duração surgiram em um sistema chamado Sinfonia [1], com o objetivo de abstrair o gerenciamento de concorrências e falhas no acesso a dados. No Augustus, elas recebem um identificador único, que é construído a partir de uma análise (*digest*) do conteúdo das operações na transação. Dessa forma, para garantir a unicidade, toda transação deve conter uma operação *nop* tendo como parâmetro um número aleatório. A vantagem das transações de curta duração utilizadas no Augustus é que elas diminuem a chance de haver conflitos de *locks* em dados, devido ao tamanho reduzido da transação. A figura 1.17 mostra uma visão geral do Augustus, onde diferentes clientes executam transações em uma ou mais partições.

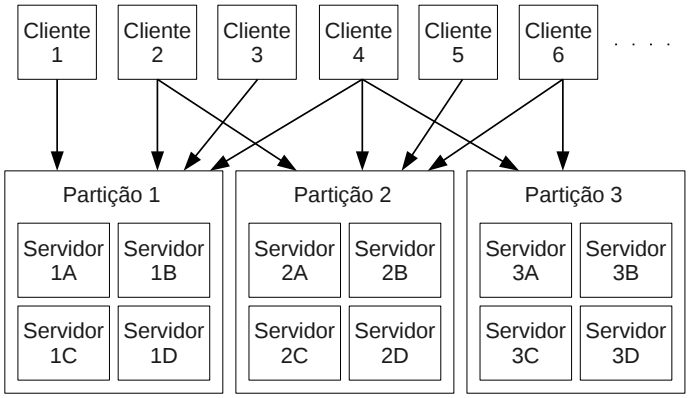



Figura 1.17: Clientes executam transações em partições [28].

As fases do protocolo estão ilustradas na figura 1.18. Inicialmente, o cliente envia a transação às partições envolvidas (passo *I*), via *multicast*. Na fase de execução (passo *II*) cada partição executa um protocolo de *atomic multicast* tolerante a faltas para ordenar as requisições; no caso do Augustus, é utilizado o PBFT [12].

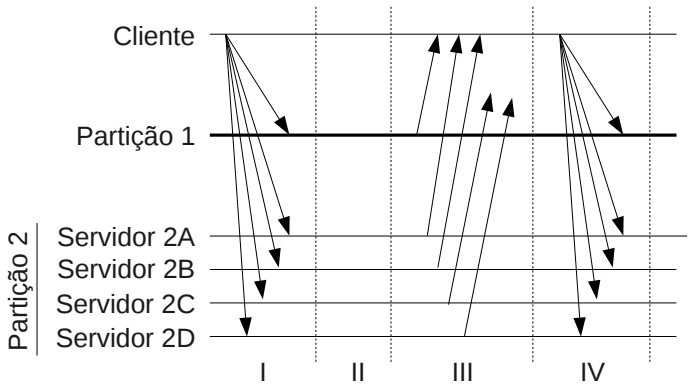



Figura 1.18: Etapas da execução global de uma transação [28].

Para executar a requisição, cada réplica na partição segue o seguinte fluxo de execução: *i*) tenta obter os *locks* dos recursos contidos na transação; *ii*) caso obtenha

sucesso, a transação é executada e assume um estado *pendente*; caso não obtenha sucesso, a transação passa ao estado *terminada* (figura 1.19); *iii*) se a transação for executada mas não obtiver sucesso (por exemplo, uma comparação que resultou em desigualdade), a mesma é abortada e passa para o estado *terminada*.

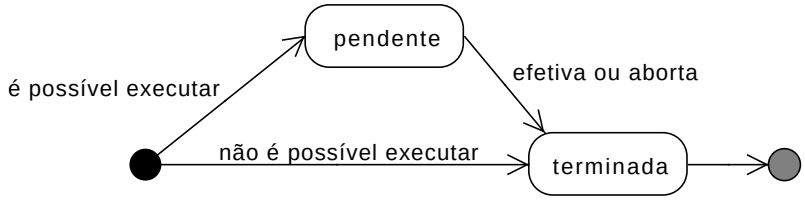



Figura 1.19: Estados possíveis a uma transação.

No passo *III*, são geradas respostas (votos) assinadas (chave privada) para o cliente: a réplica envia um *commit* ao cliente, caso tenha executado a transação com sucesso; caso contrário, envia um *abort*. Quando o cliente recebe $f_g + 1$ respostas iguais[23] das réplicas na partição $g$, e estas respostas confirmam a execução da transação, o cliente confecciona um certificado contendo um parecer *commit* referente ao voto da partição; caso contrário, é criado um certificado com o parecer *abort*. Quanto o cliente obtém certificados do tipo *commit* para todas as partições envolvidas na operação, considera-se que a transação foi efetivada globalmente (em todas as partições envolvidas). Na última etapa (passo *IV*), o cliente envia todo o conjunto de certificados que recebeu às partições envolvidas. Quando uma réplica recebe um certificado referente a uma transação que se encontra no estado "pendente", a transação é efetivada ou abortada, respectivamente, de acordo com o parecer do certificado recebido (*commit* ou *abort*); os *locks* são liberados e o estado da transação é atualizado para "terminada". Em réplicas que possuem transações já terminadas, o recebimento de certificados não tem efeito sobre a réplica.

Para ilustrar uma execução bem sucedida no Augustus, considere a figura 1.20 e a seguinte sequência de passos: 1) o cliente envia a solicitação a duas partições envolvidas; 2) cada partição executa o protocolo de ordenação de mensagens entre suas réplicas; 3) cada partição executa a transação solicitada; 4) as partições retornam ao cliente o estado "pendente" relativo ao pedido solicitado; 5) o cliente confecciona certificados das execuções bem sucedidas nas partições; 6) o cliente envia todos os certificados a todas as partições envolvidas; 7) as partições efetivam as transações e o protocolo termina.

Uma operação pode requerer mais de uma transação para realizar uma tarefa. A tabela 1.3 apresenta um algoritmo de *Compare-and-Swap* usando transações de curta duração no Augustus. São exibidos o código de duas transações e o código executado pelo cliente. Apesar de nas transações exemplificadas não constar a operação *nop* com um número aleatório, ela é necessária para aumentar a garantia de unicidade da transação. Inicialmente, o cliente tenta realizar a leitura do dado por meio da transação de leitura (linha 18). A falha na leitura significa que não foi possível obter o bloqueio necessário. Nesse caso, é possível repetir essa transação até que esse bloqueio seja alcançado. Após ler o valor, a aplicação modifica o dado (linha 20) e então submete a

[23]Note que cada partição possui $3f + 1$ réplicas, que são capazes de fornecer um quorum de $2f + 1$ respostas; este quorum deve possuir no mínimo $f + 1$ respostas iguais para caracterizar uma resposta correta.

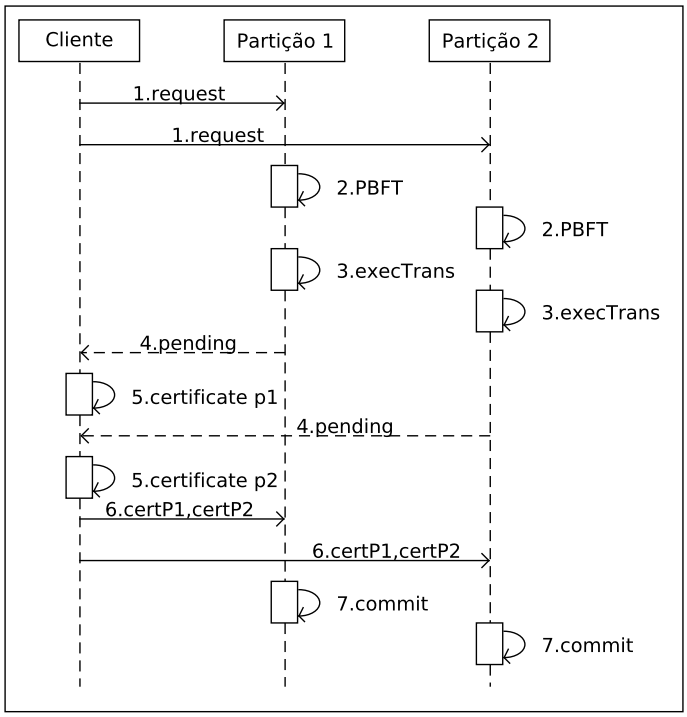



Figura 1.20: Execução bem sucedida de uma solicitação em duas partições.

transação de gravação ao Augustus (linha 21). O protocolo segue conforme o resultado da execução da transação. A gravação pode falhar quando não é possível obter o bloqueio para gravação do dado, ou quando o valor lido já foi modificado por outra aplicação. Nesse caso, será necessário executar novamente todo o código da aplicação.

Se uma transação for executada em ordens diferentes nas partições, ao menos uma das partições envolvidas enviará um *abort*, o que fará com que todas as outras partições também abortem a execução da transação. Se um cliente falhar durante a execução e deixar transações pendentes (não executarem o passo *IV* na figura 1.18), clientes corretos subsequentes atuam no sentido de solicitar o cancelamento das transações conflitantes, mediante a apresentação de certificados que mostram a situação de conflito nas outras partições. O cliente é responsável por definir o esquema de particionamento. Quando o esquema é *hashing*, quaisquer partições podem ser envolvidas. Para um esquema baseado em valor, uma análise nas operações da transação define as partições envolvidas.

O banco de dados Cassandra[24] adicionou o recurso de minitransações em sua versão 2.0 do software, permitindo, por exemplo, a não execução de um comando no caso de uma verificação falhar. O código[25] mostrado na tabela 1.4 representa uma inserção que apenas será executada se a verificação de pré-existência do dado for bem sucedida. Sem a verificação, o comando de inserção funcionaria como uma atualização, sobrescrevendo o valor antigo. Outro exemplo está na tabela 1.5: reiniciar uma senha. A senha apenas será alterada caso o campo *reset_token* possua o valor especificado.

[24]http://cassandra.apache.org/

[25]O código foi obtido no site http://www.datastax.com/dev/blog/lightweight-transactions-in-cassandra-2-0. Datastax é uma empresa que distribui comercialmente o Cassandra.

Tabela 1.3: Minitransação de alteração atômica de valor (compare-and-swap)

```
1:  function MINITRANSACTIONALREAD(key)
2:      if tryReadLock(key) then
3:          return read(key);
4:      end if
5:  end function

6:  function MINITRANSACTIONALWRITE(key, old, new)
7:      if tryWriteLock(key) then
8:          actual ← read(key)
9:          if actual == old then
10:             write(key, new);
11:             return true;
12:         end if
13:     end if
14:     return false;
15: end function
                                          ▷ Código da aplicação cliente
16: old ← null;
17: while old == null do
18:     old = miniTransactionalRead(x);
19: end while
20: new ← old + " modified";
21: success ← miniTransactionalWrite(x, old, new);
22: if success then
23:     state ← "pending"; vote ← "commit";
24: else
25:     state ← "terminated"; vote ← "abort";
26: end if
```

### 1.4.3. Depot

Depot[25] é um sistema de armazenamento chave-valor que alcança confiabilidade máxima usando a premissa de não confiar em nenhum nó. O Depot incorpora uma característica de sistemas *peer-to-peer*: a capacidade de comunicar-se com outros clientes; dessa maneira, uma aplicação pode permanecer operante mesmo se todas as réplicas do sistema estiverem comprometidas.

A arquitetura do Depot é mostrada na figura 1.21: um conjunto de servidores compõem um volume (uma partição). No exemplo da figura, cada partição contém uma faixa de chaves de um cliente (a estratégia de particionamento é definida pelo cliente). Qualquer servidor no volume pode ser contactado pelo cliente: os demais servidores recebem as atualizações de acordo com a forma de replicação considerada, que é transparente ao Depot. A figura 1.21 apresenta três exemplos de estratégias de replicação no interior das partições: *chain*, *mesh* e *star*. Caso todos os servidores fiquem indisponíveis, os clientes podem comunicar-se e compartilhar dados entre si. Clientes e servidores utilizam

Tabela 1.4: Exemplo de minitransação no Cassandra: inserção condicional.

```
INSERT INTO USERS (login, email, name, login_count) VALUES
('jbellis', 'jbellis@datastax.com', 'Jonathan Ellis', 1) IF NOT EXISTS
```

Tabela 1.5: Exemplo de minitransação no Cassandra: atualização condicional.

```
UPDATE users SET reset_token = null, password = 'newpassword'
WHERE login = 'jbellis' IF reset_token = 'some-generated-reset-token'
```

o mesmo protocolo, sendo assim denominados *nós* do sistema. Os dados replicados nos servidores também ficam armazenados nos clientes; existe um coletor de lixo que remove versões anteriores dos dados, segundo uma política de versionamento.

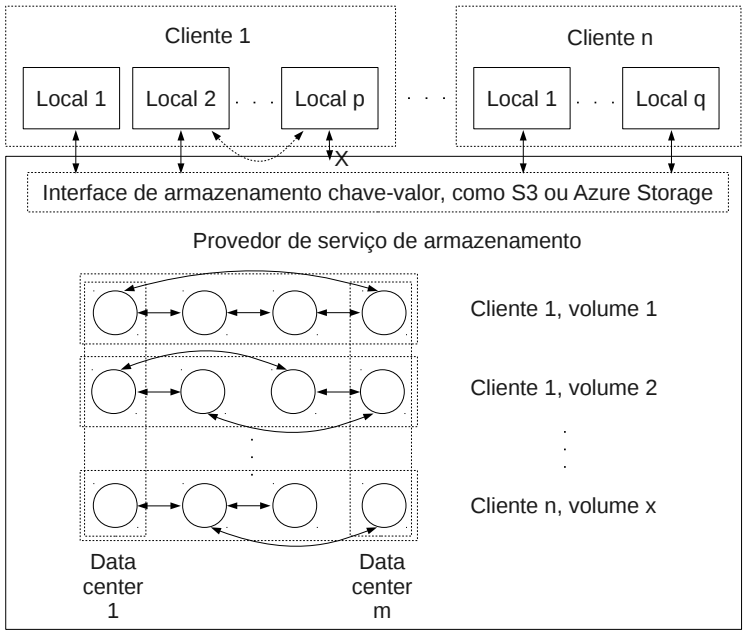



Figura 1.21: Arquitetura do Depot [25]

Para armazenar um dado no Depot, a primeira ação do cliente é armazenar o dado localmente. Em seguida, o cliente propaga a atualização para as réplicas segundo um protocolo padrão de troca de *logs* (*gossip*), em segundo plano de execução[26]. A leitura de um dado inicia-se com o cliente enviando para um servidor primário a chave $k$ e a última versão de $k$ que o cliente conhece (o cliente também participa da troca de *logs*, sendo eventualmente informado de atualizações de dados que conhece). Se o valor da última versão conhecida pelo servidor for o mesmo valor informado pelo cliente, o dado é retornado para o cliente. Toda chave possui associada um histórico de *hashes* dos valores anteriores. O cliente verifica se os *hashes* dos valores anteriores a $k$ correspondem aos que ele conhece; em caso positivo, o dado é entregue à aplicação. Caso contrário, o cliente envia ao servidor primário as informações que possui sobre o dado buscado. O servidor

[26]Uma otimização de implementação utiliza de forma otimista um servidor primário (geralmente o mais próximo), ao invés de iniciar uma troca de *logs* com um conjunto de réplicas.

deverá responder com as atualizações que o cliente não possui e também o conjunto de dados mais recente que o servidor possui. Tanto na escrita quanto na leitura de dados, caso o servidor primário não responda ao cliente, outro servidor será buscado; caso nenhum servidor responda, o cliente interage com outros clientes para realizar o armazenamento ou a busca de dados. Operações cliente-a-cliente são mais rápidas do que comunicações entre clientes e servidores, pois dispensam a troca de *logs*.

Duas escritas *A* e *B* são concorrentes logicamente se a escrita *A* não aparece no histórico da escrita *B*, e vice-versa. Se essas escritas atualizarem o mesmo objeto, elas podem ser conflitantes. Escritas são ordenadas por um relógio lógico (*timestamp*); em caso de escritas com o mesmo *timestamp*, o identificador do nó serve como critério de desempate. Diversas condições certificam a efetivação de uma escrita: assinatura digital, *timestamp* e *hash* do histórico; leituras sujas e escritas concorrentes são detectadas com estas informações. Quando servidores detectam que estão fora de sincronia, é iniciada uma troca de *logs*.

A corrupção de dados pode ser resolvida com a recuperação de versões anteriores do dado. A coleta de lixo é feita pelos nós quando estes recebem uma lista de candidatos a serem descartados (*Candidate Discard List* - CDL) assinada por todos os clientes. Uma ação de um nó faltoso para comprometer a consistência causal é realizar uma escrita divergente (com valores diferentes para nós diferentes), provocando assim um *fork*. A figura 1.22 ilustra esse cenário.

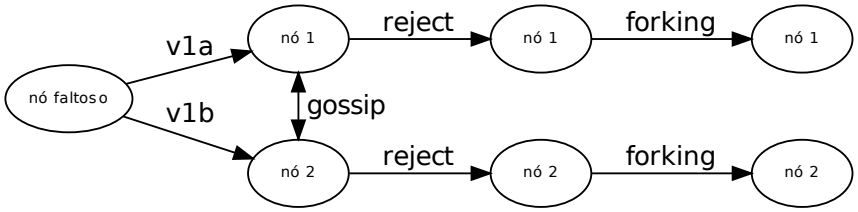



Figura 1.22: Operação *forking*: conversão de faltas em concorrência.

Para resolver essa inconsistência, cada nó considerará ambas as escritas como válidas, tornando-as concorrentes localmente; essa estratégia é denominada *forking*. No exemplo da figura 1.22, os nós trocarão informação sobre o dado a ser gravado (*gossip*); após a verificação de que os *hashes* dos históricos de valores (que incluem o valor atual) são diferentes, cada nó rejeitará a efetivação do novo valor. Após identificar que houve uma bifurcação no histórico de valores (*fork*), cada nó criará um novo dado contendo os dois valores propostos, realizando assim o *forking* que permite a coexistência de mais de um valor para uma mesma chave.

Quando uma leitura for solicitada em um nó que possui um conjunto de valores para uma determinada chave, o valor retornado será o conjunto das escritas concorrentemente realizadas. A aplicação é quem deve tomar a decisão de como proceder quanto à diversidade de valores, para realizar a convergência dos valores diferentes. Algumas estratégias sugeridas para as aplicações são: filtragem (exemplo: considerar o valor que provém do nó de maior identificador) ou substituição (exemplo: realizar uma computação dos valores retornados e utilizar este resultado). Nós faltosos são excluídos do sistema mediante provas de comportamento errôneo (*proofs of misbehavior* - POM). Essas provas

tem por base as escritas assinadas, a partir das quais é possível verificar comportamento malicioso. Seria possível, por exemplo, verificar que o nó que iniciou a operação na figura 1.22 é um nó faltoso. Altas latências em nós são tratadas como violações no SLA (*Service Level Agreement*).

O Depot tolera faltas arbitrárias por meio do enfraquecimento da semântica de dados: qualquer dado válido é considerado. A concorrência é tratada como um fato natural: ambos os valores propostos são incluídos no conjunto de valores da chave (*fork*), e a aplicação é responsável por realizar uma operação que converta os diversos valores em um único valor (*join*). Faltas arbitrárias são reduzidas a problemas de concorrência: por meio dos históricos de valores anteriores (*hashes*), é possível detectar comportamento malicioso, e assim excluir os nós bizantinos do sistema. Os servidores no interior das partições podem ser distribuídos geograficamente, o que favorece a tolerância a faltas catastróficas.

Todo nó assina as atualizações com sua chave privada. Cada cliente escritor compartilha sua chave pública para que o dado escrito possa ser lido. Por fim, confidencialidade e privacidade não são tratados no Depot, deixando a cargo da aplicação a tarefa de ofuscar os dados. Depot também não trata questões de controle de acesso dinâmico.

Foi desenvolvida uma versão do Depot denominada Teapot, que tem por objetivo armazenar dados em nuvens simples de armazenamento, como a Amazon S3, a Microsoft Azure Storage e a Google Cloud Storage. Porém, como o Depot requer que os provedores de nuvens executem código, são apresentadas no artigo algumas funcionalidades que a nuvem deveria executar para tornar possível a utilização destas nuvens de armazenamento na implementação do Teapot. Estas funcionalidades poderiam ser executadas por nuvens ativas. Por exemplo, códigos executados em máquinas virtuais na Amazon EC2 podem acessar dados armazenados na Amazon S3 sem custos, desde que a máquina virtual esteja na mesma região do provedor de dados[27]

### 1.4.4. DepSky

O DepSky [5] é um sistema de armazenamento chave-valor que utiliza diversos provedores de armazenamento em nuvem (deste ponto em diante denominados apenas como provedor, por fins de brevidade) para fornecer confiabilidade e confidencialidade ao dado armazenado. Os dados são armazenados e recuperados dos provedores utilizando-se protocolos de quóruns bizantinos [26]. O DepSky é implementado sob forma de biblioteca que deve ser utilizada junto ao cliente. Dois protocolos são responsáveis por realizar o armazenamento: o DepSky-A e o DepSkyCA.

O DepSky-A realiza a replicação integral do dado. Os passos realizados para a escrita de um dado estão representada na figura 1.23. O cliente inicialmente envia o dado a todos os provedores (etapa *I*); a quantidade de provedores é definida pela regra $n = 3f + 1$, onde $f$ é o número máximo de faltas que deseja-se tolerar no sistema, e $n$ é o total de provedores necessários para tornar possível a tolerância a $f$ faltas no sistema. Após a confirmação de $2f + 1$ respostas dos provedores (etapa *II*), relativas à gravação do dado, o cliente solicita aos provedores que armazenem os metadados correspondentes ao

[27] http://aws.amazon.com/s3/pricing/

dado que foi armazenado (etapa *III*). Após $2f+1$ confirmações de gravação do metadado pelos provedores (etapa *IV*), considera-se a operação de escrita concluída. Para ler os dados, o cliente que usa o DepSky-A solicita a todos os provedores os metadados relativos ao dado desejado. Após obter um quorum de $2f+1$ respostas, o cliente pode solicitar a qualquer provedor a leitura do dado. Nos protocolos do DepSky, a escrita de dados é feita antes da escrita de metadados para garantir que somente após a confirmação da gravação dos dados, os metadados estarão disponíveis para consulta, por clientes que desejam ler o dado.

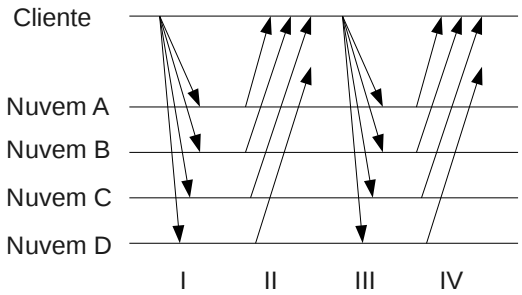



Figura 1.23: Quórum com dois *rounds*.

O protocolo DepSky-CA aplica fragmentação com redundância (seção 1.3.3.2) por meio da técnica de *erasure codes* [30] para otimizar o uso do armazenamento, reduzindo os custos de replicação. Além disso, o DepSky-CA garante a confidencialidade do dado por meio de cifragem dos dados, utilizando uma chave simétrica gerada aleatoriamente. Esta chave é distribuída entre os servidores utilizando-se a técnica de compartilhamento secreto [33] (seção 1.3.1). A fragmentação e o compartilhamento secreto são combinados seguindo uma estratégia já estabelecida [21], de forma que um bloco é composto por um fragmento do dado e um compartilhamento secreto da chave. Os blocos são os elementos distribuídos para serem armazenados nos provedores. A figura 1.24 ilustra o processo de escrita de dados nos provedores, utilizando-se o protocolo DepSky-CA. Inicialmente, o dado é cifrado (passo *I*) utilizando-se uma chave $K$ gerada aleatoriamente. A seguir (passo *II*), o dado cifrado é fragmentado ($F1$ e $F2$) e fragmentos redundantes são gerados ($F3$ e $F4$). A chave $K$ é codificada em compartilhamentos $S1..S4$ (passo *III*). São criados blocos de dados (passo *IV*): cada bloco agrega um fragmento e um compartilhamento. Os blocos $F1+S1$, $F2+S2$, $F3+S3$ e $F4+S4$ são enviados aos provedores. Após a escrita do dado, é realizada a escrita dos metadados, que contém o número de versão do dado e dados de verificação. A leitura de dados no DepSky-CA é realizada consultando-se blocos de dados de $m$ provedores, tal que $m$ é o número mínimo de fragmentos necessários para reconstruir o dado e a chave usada na cifragem.

O controle de concorrência de escrita no DepSky utiliza um mecanismo de baixa contenção: inicialmente, o escritor verifica se há um arquivo de bloqueio no formato *lock*-ID-T nos provedores, onde *ID* é o identificador de algum outro cliente que possa estar realizando a escrita, e $T$ é uma referência tal que o tempo local $t$ do cliente deve ser menor que $T+\delta$. O parâmetro $\delta$ é uma diferença estimada máxima entre os relógios dos outros escritores. Se um arquivo de bloqueio com este formato for encontrado, considera-se que existe outro cliente atuando como escritor; o cliente deverá aguardar por um tempo aleatório e tentar novamente realizar este procedimento. Se o escritor não

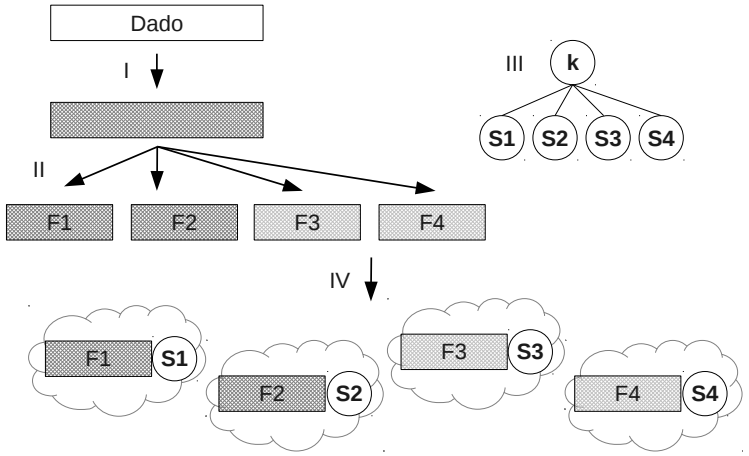



Figura 1.24: DepSky-CA: composição de técnicas na escrita de dados [5].

encontrar o arquivo segundo estas condições, ele poderá escrever o arquivo de bloqueio, definindo $T = t + tempodebloqueio$. O arquivo de bloqueio terá o formato *lock*-p-T. A seguir, o cliente busca novamente por algum arquivo de bloqueio com $t < T + \delta$. Se for encontrado algum arquivo (com exceção do arquivo escrito pelo próprio cliente), o cliente remove seus bloqueios e aguarda um tempo aleatório antes de tentar novamente a execução completa do procedimento de bloqueio.

O DepSky admite a existência de um mecanismo para compartilhamento de chaves assimétricas; no entanto, este é um dos principais desafios para garantir o sigilo de dados armazenados em provedores nuvem, quando se utiliza criptografia. O DepSky possui como ponto de vulnerabilidade a reutilização do controle de acesso fornecido pelos provedores de nuvem pública.

Foram realizados alguns testes com o Depsky, utilizando o código disponível em http://code.google.com/p/depsky/. Os testes foram feitos utilizando o armazenamento local, ao invés do armazenamento nos provedores de nuvens. O sistema operacional utilizado foi o Debian 7.0.

O arquivo obtido no site do DepSky foi o DepSky-v0.4.zip, que contém os seguintes arquivos e diretórios:

```
$ ls
bin  config  DepSky_Run.sh  dist  lib  README.txt  Run_LocalStorage.sh  src
```

Como será utilizado armazenamento local, o serviço de servidor será iniciado em um terminal específico para tal fim:

```
$ sh Run_LocalStorage.sh
```

Em um segundo terminal o cliente será ativado. Três parâmetros são necessários para executar o programa: identificador do cliente, protocolo e tipo de armazenamento. Para protocolo, utiliza-se 0 para o DepSkyA e 1 para o DepSky-CA. Para tipo de armazenamento, utiliza-se 0 para as nuvens e 1 para armazenamento local. Para acionar o cliente inicialmente com o protocolo DepSky-A, utiliza-se:

```
 $ sh DepSky_Run.sh 1 0 1
USAGE:  commands             function
       pick_du 'name' - change the container
       write 'data'   - write a new version in the selected container
       read           - read the last version of the selected container
       delete         - delete all the files in the selected container
       read_m 'num'   - read old versions written in this running. If 'num' = 0 read the last version

starting drivers...
All drivers started.
```

Seguem comandos[28] que demonstram o armazenamento e a recuperação de dados:

```
pick_du cliente
DataUnit 'cliente' selected!
write João da Silva
WRITING: João da Silva
I'm finished write -> 276 milis
read
I'm reading
I'm finished read -> 158 milis
READ RESULT = João da Silva
write Marina Lima
WRITING: Marina Lima
I'm finished write -> 241 milis
read
I'm reading
I'm finished read -> 139 milis
READ RESULT = Marina Lima
read_m 0
I'm reading
I'm finished read -> 126 milis
READ RESULT = Marina Lima
read_m 1
I'm reading
I'm finished read -> 170 milis
READ RESULT = João da Silva
```

Em um terceiro terminal, pode-se verificar os tamanhos dos dados armazenados localmente; nos arquivos de metadados existem dados binários, porém apenas os caracteres textuais foram incluídos na listagem a seguir. Note que o tamanho do arquivo *clientevalue*1001 é igual à quantidade de caracteres da frase 'João da Silva', mais um caractere de quebra de linha.

```
$ cd ds-local/cloud1/

$ ls -la
total 20
drwxr-xr-x 2 friend friend 4096 Aug 29 17:05 .
drwxr-xr-x 6 friend friend 4096 Aug 29 17:05 ..
-rw-r--r-- 1 friend friend  788 Aug 29 17:05 clientemetadata
-rw-r--r-- 1 friend friend   14 Aug 29 17:05 clientevalue1001
-rw-r--r-- 1 friend friend   11 Aug 29 17:05 clientevalue2001

$ cat clientevalue1001
João da Silva

$ cat clientemetadata
clientevalue2001
versionNumber = 2001
versionHash = 98e1d18783872be2c877837ec27801d7921646e4
allDataHash = [B@14669f26

clientevalue1001
versionNumber = 1001
versionHash = ded70e73e90c3fb8beaa443f1b1c4d5f7b52100c
allDataHash = [B@73edc5f6
```

Para a execução do protocolo DepSky-CA, utiliza-se o seguinte comando:

```
 $ sh DepSky_Run.sh 1 1 1
```

Novamente seguem comandos que demonstram a manipulação de dados:

[28] Note que a sequência está exibida em duas colunas.

```
pick_du vendedor
DataUnit 'vendedor' selected!
write Sebastião Homenish
WRITING: Sebastião Homenish
I'm finished write -> 890 milis
read
I'm reading
I'm finished read -> 172 milis
READ RESULT = Sebastião Homenish
write Suzan Croft Silva
WRITING: Suzan Croft Silva
I'm finished write -> 268 milis
read
I'm reading
I'm finished read -> 182 milis
READ RESULT = Suzan Croft Silva
read_m 1
I'm reading
I'm finished read -> 169 milis
READ RESULT = Sebastião Homenish
```

Para observar os conteúdos dos dados armazenados, foram utilizados os comandos a seguir. Note que nos metadados há informações sobre os compartilhamentos secretos (PVSS[29]) e sobre os *erasure codes* (*reed_sol_van*[30]).

```
$ cd ds-local/cloud1/

$ ls -la
total 20
drwxr-xr-x 2 friend friend 4096 Aug 30 09:45 .
drwxr-xr-x 6 friend friend 4096 Aug 30 09:44 ..
-rw-r--r-- 1 friend friend 1002 Aug 30 09:45 vendedormetadata
-rw-r--r-- 1 friend friend  685 Aug 30 09:45 vendedorvalue1001
-rw-r--r-- 1 friend friend  685 Aug 30 09:45 vendedorvalue2001

$ cat vendedorvalue1001
sr depskys.core.ECKSObject ec_bytest [BL ec_filenamet Ljava/lang/String;sk_sharet
Lpvss/Share;xpur pvss.Share Iindex[ LencryptedSharet Ljava/math/BigInteger;L
interpolationPointq~ proofcq~ proofrq~ Lshareq~ xpuq~ java.math.BigInteger bitCountI
bitLengthI firstNonzeroByteNumI lowestSetBitI signum[ magnitudeq~ xr java.lang.Number
xpsq~ xsq~ xsq~ Aqx (dados binários foram omitidos)

$ cat vendedormetadata
vendedorvalue2001
versionNumber = 2001
versionHash = 2d2ef19f772c6d67e3375c22466dc5b969389d6b
allDataHash = [B@7337e552
versionFileId = vendedorvalue2001
versionPVSSinfo = 4;2;2373168401;1964832733;1476385517
versionECinfo = 32;2 2 8 0 32;reed_sol_van;0;1;

vendedorvalue1001
versionNumber = 1001
versionHash = 3001c5b10f1d8c2ece05aa2bc9b32d46bb1deafc
allDataHash = [B@572b06ad
versionFileId = vendedorvalue1001
versionPVSSinfo = 4;2;2373168401;1964832733;1476385517
versionECinfo = 32;2 2 8 0 32;reed_sol_van;0;1;v (dados binários foram omitidos)
```

Um outro teste foi realizado com o objetivo de verificar os tamanhos dos blocos de dados utilizando-se os dois protocolos DepSky-A e DepSky-CA. Foram gerados arquivos de diversos tamanhos[31]; os conteúdos dos arquivos foram copiados e colados diante do comando *write*, na execução do cliente do DepSky.

[29] PVSS significa *public verifiable secret sharing*, e refere-se ao fato dos compartilhamentos serem verificáveis publicamente, ou seja, por entidades externas ao sistema.

[30] *Reed Solomon* é uma técnica de geração de códigos de erro; no caso, foi especificada essa técnica para ser utilizada pelo *erasure code* na geração dos fragmentos redundantes.

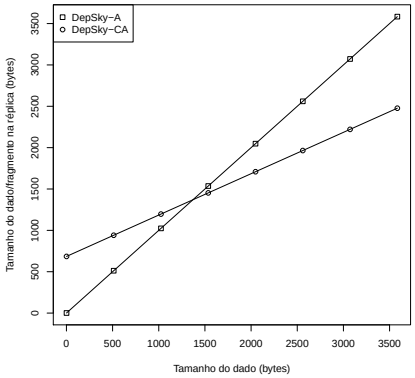
[31] Como exemplo, para gerar um arquivo de 1Kb, foi usado o comando: tr -dc A-Za-z0-9 < /dev/urandom | head -c 1024 > 1k.txt

A tabela 1.25a mostra resultados do armazenamento de dados usando os protocolos DepSky-A e DepSky-CA e variando o dado de 1 bytes até 3584 bytes, com um fator incremental de 256 bytes. Naturalmente, o tamanho do bloco de dado no DepSky-A é exatamente igual ao tamanho do dado, visto que a replicação é integral. No DepSky-CA, o tamanho refere-se ao tamanho do fragmento do *erasure code*. O valor máximo do tamanho do dado igual a 3584 deve-se ao fato de a execução do cliente do DepSky, no terminal de comandos, não aceitar 4096 bytes no comando de escrita. Para testes com maiores tamanhos de dados, o DepSky deve ser usado como código de biblioteca em um programa Java. A figura 1.25b contém os dados da tabela 1.25a; é possível observar o ponto de inversão a partir do qual os dados armazenados pelo protocolo DepSky-CA apresentam vantagem, de fato, da redução de tamanho por meio do uso de fragmentos ao invés de replicar o dado integralmente.

| Tamanho do dado/DepSky-A | DepSky-CA |
|---|---|
| 1 | 685 |
| 512 | 941 |
| 1024 | 1197 |
| 1536 | 1453 |
| 2048 | 1709 |
| 2560 | 1965 |
| 3072 | 2221 |
| 3584 | 2477 |

(a)




(b)

Figura 1.25: Protocolos do DepSky e ocupação de bytes pelo dado armazenado.

### 1.4.5. SCFS

SCFS (*Shared Cloud-backed File System*) é uma arquitetura modular que permite acessar serviços de armazenamento chave-valor em provedores de armazenamento em nuvens (a partir deste ponto denominados provedores, para fins de brevidade) por meio da abstração de sistema de arquivos. Os provedores comerciais oferecem apenas consistência eventual, que pode não ser suficiente para alguns tipos de aplicações. O SCFS traz como principal contribuição o provimento de consistência forte utilizando provedores de consistência eventual.

Dois componentes principais compõem a arquitetura do SCFS (figura 1.26): um serviço de armazenamento (SS) e um coordenador de serviços (CA). O SS é representado por provedores passivos como a Amazon S3 e Windows Azure Storage; estes provedores contém primitivas que realizam apenas leitura e escrita de dados. O CA deve ser instanciado por servidores que tenham a capacidade de realizar processamento, como a Amazon EC2. Sistemas como o ZooKeeper[32] ou o DepSpace [7] são apropriados para realizar o serviço de coordenação que este componente requer. O cliente possui um *cache* que contém todas as informações do sistema de arquivos replicado, e possui também um agente

[32]http://zookeeper.apache.org/

do SCFS que comunica-se com o SS e com o CA.

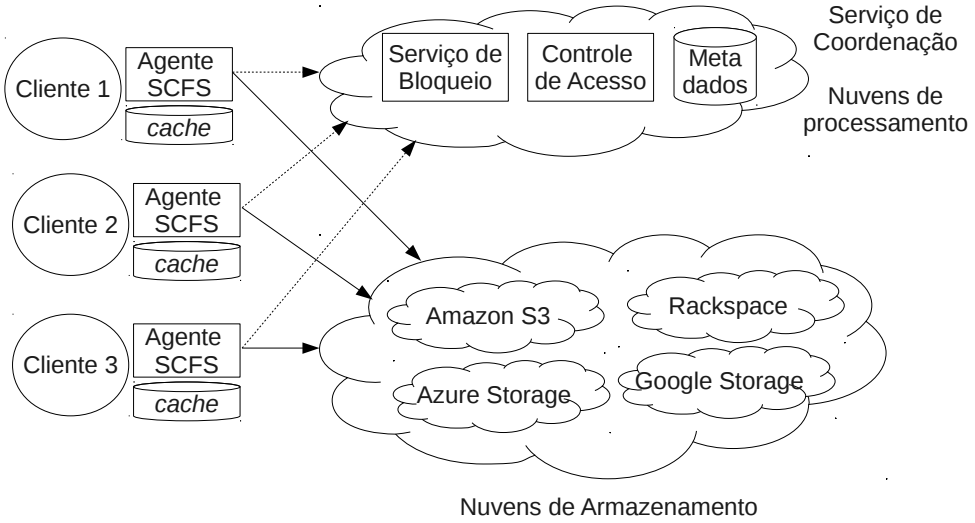



Figura 1.26: Arquitetura do SCFS [6].

O SCFS consegue prover consistência forte utilizando provedores que trabalham sob consistência eventual. Para alcançar essa característica, os procedimentos de escrita e leitura foram modificados segundo os algoritmos mostrados na figura 1.27 e representados na figura 1.28. Para realizar uma escrita de dados, considere o par chave-valor como sendo os identificadores *id* e *v*. Inicialmente, o agente gera um *hash H* de *v*; em seguida, *v* é gravado no SS, sob a chave *id+H* (concatenação de *id* e *H*). Após a gravação de *v*, *H* é gravado no CA, sob a chave *id*. Estas informações que ficam armazenadas no CA garantem a consistência forte e são chamadas de âncoras de consistência.

| WRITE($id, v$): | READ($id$): |
|---|---|
| **w1:** $h \leftarrow \mathsf{Hash}(v)$ | **r1:** $h \leftarrow$ CA.read($id$) |
| **w2:** SS.write($id\|h, v$) | **r2:** **while** $v = null$ **do** $v \leftarrow$ SS.read($id\|h$) |
| **w3:** CA.write($id, h$) | **r3:** **return** $(\mathsf{Hash}(v) = h)?v : null$ |

Figura 1.27: Algoritmos de escrita e leitura que alcançam consistência forte.

Para ler um dado, o agente solicita ao CA o *hash* do dado. Após obter *H*, o agente busca o dado no SS, utilizando a chave *id+H*, até obter o valor *v*. Por fim, é verificado se o *hash* do valor retornado pelo SS é igual ao *hash* fornecido pelo CA. Caso não seja, é retornado nulo.

O SCFS conta com um *cache* local para otimizar as operações. Quando um usuário abre um arquivo, o agente lê o metadado (de CA), opcionalmente cria um *lock* se a operação for escrita, lê o conteúdo do arquivo (de SS) e salva este conteúdo no *cache* local. As operações de leitura e escrita (descritas a seguir) são operações locais, ou seja, operam sobre o *cache*. Quando ocorre escrita em um arquivo, é feita atualização do conteúdo em *cache* e em algumas partes do metadado (tamanho, data da última atualização). A leitura de um arquivo envolve apenas a busca do dado no *cache*, já que o conteúdo foi carregado nessa memória quando o arquivo foi aberto. Fechar um arquivo requer a

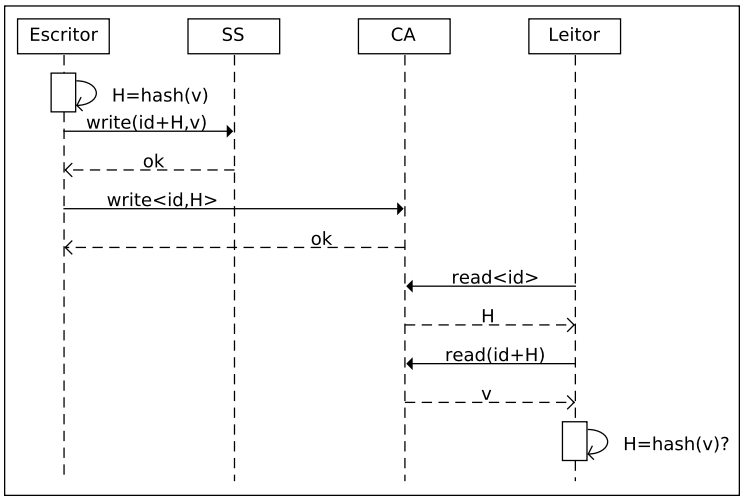



Figura 1.28: Sequência de escrita e leitura no SCFS.

sincronização dos dados e metadados. Primeiramente, o arquivo é copiado para o disco local e para as nuvens. A seguir, o metadado no *cache* é modificado, e enviado ao serviço de coordenação. Por fim, o *lock* é removido, caso o arquivo tenha sido aberto para escrita.

No SCFS, arquivos não-compartilhados são tratados de forma diferente: ao montar o sistema de arquivos, o agente SCFS obtém as entradas PNS (*Private Name Spaces*) do serviço de coordenação, e os metadados do SS, bloqueando (no CA) as PNS para evitar inconsistências, caso dois usuários estejam conectados simultaneamente com a mesma conta. Para abrir um arquivo, o usuário acessa o metadado localmente, e se necessário busca os dados dos SS. Ao fechar um arquivo modificado, dados e metadados são enviados ao SS.

Há um *garbage collector* que desativa versões antigas do dado. O sistema usa ACL's (*access control lists*) para controle de acesso. São avaliadas três versões do sistema de arquivos: bloqueante, não-bloqueante e não-compartilhado. O serviço de armazenamento pode ser o DepSky ou diretamente uma nuvem apenas. Foram usadas máquinas da Amazon EC2 para implementar a coordenação de serviços, e máquinas Amazon S3 para o serviço de armazenamento.

### 1.4.6. CloudSec

O CloudSec [14] é um *middleware* capaz de compartilhar documentos sigilosos utilizando provedores de armazenamento em nuvens públicas para hospedar os documentos, e módulos de segurança criptográficos[33] (MSC) - que integram uma nuvem privada - para gerenciar as chaves criptográficas. O *middleware* localiza-se em uma nuvem híbrida onde o controle de acesso é deixado a cargo da organização, que geralmente já possui um sistema de controle de acesso estabelecido. A figura 1.29 apresenta a arquitetura do CloudSec. Os gerenciadores de chaves criptográficas encontram-se instalados nos MSC, e utilizam criptografia baseada em identidades para a criação e fornecimento de chaves. Os provedores de nuvens públicas possuem o controle de acesso aos blocos de dados e

[33]Exemplo de um MSC em http://www.kryptus.com/#!asi-hsm/c11e6

são responsáveis pelo armazenamento do conteúdo dos documentos sigilosos. O usuário é responsável por editar, cifrar e decifrar os documentos. No *middleware*, o controlador contém as APIs que o usuário utiliza para manipular os documentos; o manipulador de chaves trata das operações de cifrar e decifrar, bem como a criação de chaves simétricas e assimétricas; e o manipulador de arquivos envolve o armazenamento e a recuperação de dados, interagindo com as nuvens públicas.

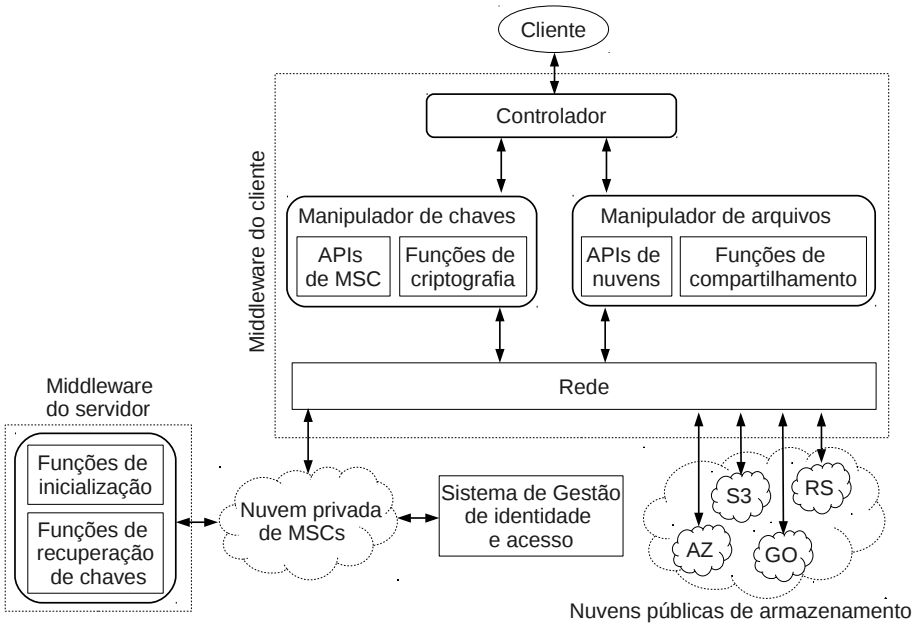



Figura 1.29: Arquitetura do CloudSec [14].

O CloudSec utiliza um mecanismo de geração de chave distribuída de maneira que nenhum dos MSC possuem a chave privada dos usuários. Ao solicitar uma chave aos MSC, é preciso que ao menos dois MSC forneçam suas respostas para que o usuário possa recuperar sua chave privada. Assim, mesmo que um MSC seja acessado indevidamente (o que é pouco provável, dadas as características inerentes desse componente), a segurança das chaves não seria comprometida, visto que um MSC não possui acesso claro às chaves privadas.

O processo de autenticação no CloudSec segue os passos ilustrados na figura 1.30. Inicialmente, para que um usuário possa ter acesso ao provedor de nuvem pública para armazenamento, o mesmo já deve ter sido cadastrado em um sistema de gestão de identidades (SGI) em uma aplicação separada. Nesse caso, os mesmos dados para autenticação serão utilizados perante os gerenciadores de chaves criptográficas (GCC). Esses irão receber os dados dos usuários (número 1 da figura) e entrarão em contato com um SGI para conferir se os dados conferem (número 2 da figura). Caso os dados estejam corretos, o SGI informará o GCC que o usuário autenticou-se corretamente e informará quais dados pode ter acesso (número 3 da figura). O GCC então irá entrar em contato com os provedores de nuvem pública para solicitar *tokens* de acesso para o usuário com as restrições necessárias (número 4 da figura). Os provedores de nuvem para armazenamento retornam tokens de acesso com as devidas restrições e um tempo de expiração (número 5 da figura). Após receber os *tokens*, o GCC então enviará os mesmos para o usuário que o uti-

lizará para ter acesso por um tempo pré-determinado de uso e com direitos limitados para acessar apenas os dados que pode ter acesso (número 6 da figura). O usuário então utilizará esse *token* para autenticar-se perante os provedores de nuvem para armazenamento e assim conseguirá ter acesso aos dados (número 7 da figura).

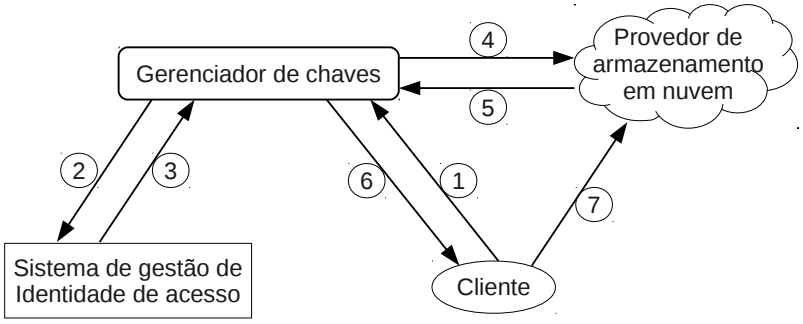



Figura 1.30: Processo de autenticação no CloudSec [14].

O SGI é compartilhado entre todos os GCCs, fazendo com que o controle de acesso não precise ser replicado. Após a autenticação, cada GCC irá emitir todos os *tokens* necessários para acessar os provedores de armazenamento em nuvem. Dessa forma, caso o usuário consiga receber corretamente os *N* tokens corretamente do primeiro GCC, não necessita mais receber *tokens* dos outros GCCs para se autenticar perante os provedores de nuvem para armazenamento. Cada provedor de nuvem possui uma peculiaridade com relação aos *tokens*, portanto, o GCC deverá emitir *tokens* conforme as propriedades necessárias para cada provedor de nuvem. Por exemplo, o provedor de nuvem para armazenamento da empresa Google utiliza o mecanismo OAuth 2 para a emissão do *token* de autorização; a empresa Amazon com seu serviço de armazenamento utiliza conceitos de *token* com o nome *Token Vending Machine*.

Após a obtenção do *token*, os documentos sigilosos são cifrados com a chave simétrica criada pelo usuário (aleatória). A chave simétrica é cifrada com uma chave privada que é construída a partir de um identificador *ID*, que é composto pelo nome do documento, um nome de grupo e um número de versão do documento. A chave simétrica cifrada é então codificada em compartilhamentos por meio da técnica de compartilhamento secreto (seção 1.3.1). Os dados cifrados são codificados com o mecanismo de *erasure code* (seção 1.3.3.2) para possibilitar economia no armazenamento de dados. As partes cifradas das chaves e documentos são assinadas com o algoritmo de Hess, para garantir a integridade dos dados.

Com a integração das técnicas de gerenciamento de chaves e armazenamento de dados em provedores de nuvens, o CloudSec provê os seguintes benefícios:

- Apenas o usuário possui a chave privada; as entidades distribuidoras possuem apenas componentes necessários à geração da chave privada.
- Após a retirada de um usuário do grupo, o mesmo não terá mais acesso aos documentos cifrados, nem às próximas versões do documento. Isto deve-se ao fato de o identificador do documento conter a versão do documento.
- Faltas bizantinas são toleradas, com a utilização replicada (em $3f + 1$) de provedores de dados (nuvens públicas) e provedores de chaves.

## 1.5. Conclusões

A computação em provedores de nuvens é uma realidade que estabeleceu-se e possui demanda crescente no espaço da Internet. A comunidade científica também avança nas pesquisas sobre esse tema, expondo trabalhos nas conferências mais relevantes da área, listadas na tabela 1.6, na percepção dos autores deste minicurso. Nota-se que os eventos que possuem tópicos como sistemas operacionais, confiabilidade e sistemas distribuídos têm tratado de artigos na área de computação em nuvens. Isto deve-se ao fato de que nos bastidores do provedor de nuvens as tecnologias relacionadas a esses tópicos são predominantes na construção da arquitetura e na operacionalização dos serviços.

| Sigla | Evento | Vínculo |
|---|---|---|
| DSN | International Conference on Dependable Systems and Networks | IEEE |
| FAST | Conference on File and Storage Technologies | Usenix |
| NSDI | Symposium on Networked Systems Design and Implementation | Usenix |
| OSDI | Symposium on Operating Systems Design and Implementation | Usenix |
| SOSP | Symposium on Operating Systems Principles | ACM |
| SRDS | International Symposium on Reliable Distributed Systems | IEEE |
| ATC | Annual Technical Conference | Usenix |

Tabela 1.6: Principais conferências com artigos sobre armazenamento em nuvens.

Um dos maiores desafios continua sendo a garantia de confidencialidade em dados e operações, em um ambiente que não pode ser considerado confiável, pois encontra-se em local não conhecido pelo cliente e sob administração de terceiros. Além disso, recentemente tem-se buscado direitos à privacidade, bem como o "direito a ser esquecido"; empresas como a Google estão promovendo discussões[34] sobre como lidar com o armazenamento de informações na Internet de forma a permitir garantias de exclusão de dados, quando requisitadas pelo usuário.

A busca pelo equilíbrio entre o esforço para manter os dados confidenciais e a transparência na utilização do serviço remoto motiva o desenvolvimento de protocolos e arquiteturas como essas que foram explanadas neste minicurso. Como toda novidade, a computação em nuvens talvez encontre seu ponto de equilíbrio na utilização das nuvens híbridas, com parte de informações críticas, ou metadados, no controle das organizações e clientes, e volumes de dados ou cópias adicionais de segurança sob a responsabilidade dos grandes provedores de nuvens, que possuem capacidades e infraestrutura praticamente ilimitados, sendo o custo o maior fator de ponderação de sua utilização.

## Referências

[1] Marcos K Aguilera, Arif Merchant, Mehul Shah, Alistair Veitch, and Christos Karamanolis. Sinfonia: a new paradigm for building scalable distributed systems. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 159–174. ACM, 2007.

[2] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM (JACM)*, 42(1):124–142, 1995.

[34] http://www.bbc.com/news/technology-28344705

[3] Algirdas Avizienis and John PJ Kelly. Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80, 1984.

[4] Rida A Bazzi and Yin Ding. Non-skipping timestamps for byzantine data storage systems. In *Distributed Computing*, pages 405–419. Springer, 2004.

[5] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. *Transactions on Storage*, 9(4):12:1–12:33, November 2013.

[6] Alysson Bessani, Ricardo Mendes, Tiago Oliveira, Nuno Neves, Miguel Correia, Marcelo Pasin, and Paulo Verissimo. Scfs: a shared cloud-backed file system. In *Proc. of the USENIX ATC*, 2014.

[7] Alysson Neves Bessani, Eduardo Pelison Alchieri, Miguel Correia, and Joni Silva Fraga. Depspace: a byzantine fault-tolerant coordination service. In *ACM SIGOPS Operating Systems Review*, volume 42, pages 163–176. ACM, 2008.

[8] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229. Springer, 2001.

[9] Eric Brewer. Cap twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, 2012.

[10] Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. Distributed systems (2nd ed.). chapter The Primary-backup Approach, pages 199–216. ACM Press/Addison-Wesley Pub., NY, USA, 1993.

[11] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[12] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[13] George F Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed systems: concepts and design*. Pearson Education, 2012.

[14] Rick Lopes de Souza, H.V. Netto, L. C. Lung, and R. F. Custodio. Cloudsec - um middleware para compartilhamento de informacoes sigilosas em nuvens computacionais. In *XIV Simposio Brasileiro de Seguranca da Informacao e de Sistemas Computacionais - SBSEG*, Belo Horizonte, MG, 2014.

[15] Yves Deswarte, Laurent Blain, and J-C Fabre. Intrusion tolerance in distributed computing systems. In *Proc. of Symposium on Security and Privacy*, pages 110–121. IEEE, 1991.

[16] Idilio Drago, Marco Mellia, Maurizio M Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. Inside dropbox: understanding personal cloud storage services. In *Proc. of Conference on Internet Measurement*, pages 481–494. ACM, 2012.

[17] Joni Fraga and David Powell. A fault-and intrusion-tolerant file system. In *Proc. of International Conference on Computer Security*, volume 203, pages 203–218, 1985.

[18] JL Gonzalez, Jesus Carretero Perez, Victor Sosa-Sosa, Juan F Rodriguez Cardoso, and Ricardo Marcelin-Jimenez. An approach for constructing private storage services as a unified fault-tolerant system. *Journal of Systems and Software*, 86(7):1907–1922, 2013.

[19] Glenn Greenwald and Ewen MacAskill. Nsa prism program taps in to user data of apple, google and others. *The Guardian*, 7(6):1–43, 2013.

[20] Paul T Jaeger, Jimmy Lin, Justin M Grimes, and Shannon N Simmons. Where is the cloud? geography, economics, environment, and jurisdiction in cloud computing. *First Monday*, 14(5), 2009.

[21] Hugo Krawczyk. Secret sharing made short. In *Advances in Cryptology - CRYPTO'93*, pages 136–146. Springer, 1993.

[22] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[23] Leslie Lamport. On interprocess communication. *Distributed computing*, 1(2):86–101, 1986.

[24] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[25] Prince Mahajan, Srinath Setty, Sangmin Lee, Allen Clement, Lorenzo Alvisi, Mike Dahlin, and Michael Walfish. Depot: Cloud storage with minimal trust. *ACM TOCS*, 29(4):12:1–12:38, 2011.

[26] Dahlia Malkhi and Michael Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[27] Ricardo Padilha and Fernando Pedone. Belisarius: Bft storage with confidentiality. In *10th International Network Computing and Applications (NCA) Symposium on*, pages 9–16. IEEE, 2011.

[28] Ricardo Padilha and Fernando Pedone. Augustus: scalable and robust storage for cloud applications. In *Proc. of the 8th European Conference on Computer Systems*, pages 99–112. ACM, 2013.

[29] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proc. of the Int. Conf. on Management of Data*, SIGMOD, pages 109–116, NY, USA, 1988. ACM.

[30] Michael O Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[31] Chhanda Ray et al. *Distributed Database Systems*. Pearson Education India, 2009.

[32] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM CSUR*, 22(4):299–319, 1990.

[33] Adi Shamir. How to share a secret. *Communic. of the ACM*, 22(11):612–613, 1979.

[34] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer, 1985.

[35] Andrew Tanenbaum and Maarten Van Steen. *Distributed systems*. Pearson Prentice Hall, 2007.

[36] Werner Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009.

[37] Zhe Wu, Michael Butkiewicz, Dorian Perkins, Ethan Katz-Bassett, and Harsha V Madhyastha. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proc. of SOSP*, pages 292–308. ACM, 2013.

# Capítulo 2

# Device Fingerprinting: Conceitos e Técnicas, Exemplos e Contramedidas

Adriana Rodrigues Saraiva, Pablo Augusto da Paz Elleres, Guilherme de Brito Carneiro e Eduardo Luzeiro Feitosa

***Abstract***

*Fingerprinting methods are those used to identify (or re-identify) a user or a device through a set of attributes (device screen size, versions of installed software, and so on) and other observable features during communication process. Commonly known as Device Fingerprinting, such techniques can be used as a security measure (to authenticate users, for example) and as mechanism for sales/marketing. Unfortunately, they can also be considered a potential threat to Web users' privacy, since personal and sensitive data can be captured and used for malicious purposes in various types of attacks and fraud. This Chapter introduces the concepts and techniques of device fingerprinting that allow to obtain data on users and their devices. Also presents tools and countermeasures to deal with the problem.*

***Resumo***

*Técnicas de fingerprinting são aquelas empregadas para identificar (ou reidentificar) um usuário ou um dispositivo através de um conjunto de atributos (tamanho da tela do dispositivo, versões de softwares instalados, entre muitos outros) e outras características observáveis durante processo de comunicação. Comumente conhecidas por Device Fingerprinting, tais técnicas podem ser usadas como medida de segurança (na autenticação de usuários, por exemplo) e como mecanismo para vendas/marketing. Infelizmente, também podem ser consideradas uma ameaça potencial a privacidade Web dos usuários, uma vez que dados pessoais e sigilosos podem ser capturados e empregados para fins maliciosos nos mais variados tipos de ataque e fraudes. Este Capítulo apresenta os conceitos e as técnicas de device fingerprinting que permitem obter dados relativos aos usuários e seus dispositivos. Apresenta também ferramentas e contramedidas para lidar com o problema.*

## 2.1. Introdução

A Internet se tornou uma realidade na vida de milhões de pessoas. Busca por informações, comunicação em redes sociais, compras, jogos on-line e até Internet Banking são atividades populares e cotidianas. O fato é que os inúmeros avanços tecnológicos proporcionaram aos usuários uma maior acessibilidade, agilidade e dinamismo no acesso as informações e na execução de tarefas. Entretanto, esses mesmos avanços acabaram ou permitiram a manipulação de uma das características básicas da origem da Internet, o anonimato. O simples ato de navegar e acessar sites e serviços de forma anônima, ou através do uso de pseudônimos, já não é mais verdade.

O trabalho de [Nikiforakis et al. 2014], entitulado "Browser Fingerprinting and the on-line-Tracking Arms Race" relata um fato de julho de 1993, onde uma charge de Peter Steiner, publicada pelo jornal The New Yorker, mostrava um labrador sentado em uma cadeira na frente de um computador, digitando e comentando para um companheiro (um beagle) ao lado: "Na Internet, ninguém sabe que você é um cachorro". Passadas duas décadas, as partes interessadas na comunicação do cachorro, não só saberiam que ele é um cão, como teriam uma boa ideia da sua cor, de quantas vezes ele foi ao veterinário e qual o seu biscoito favorito. O fato é que hoje é muito fácil identificar um usuário na Internet.

A premissa básica é que o ato de navegar deixa vários rastros. Por exemplo, ao acessar um site, um usuário tem o seu dispositivo (computador, tablet, smartphone, entre outros) identificado através do endereço IP. Assim, basta uma simples consulta a bases de dados e a serviços de localização, como o Whois[1], para descobrir onde o usuário está e de qual local ele está fazendo o acesso. Outro modo de identificar usuários é através de Cookies, pequenos pedaços de texto (arquivos) que os sites fazem o navegador do usuário salvar com o objetivo de identificá-lo em um futuro acesso. Embora tenham sido criados para personalizar a navegação dos usuários, os Cookies passaram a ser utilizados para registrar informações com fins comerciais, em especial com o objetivo de vender o endereço eletrônico (caso do spam) do usuário ou seus hábitos de navegação e de consumo a sites de anúncios. Assim, a capacidade de rastrear um usuário se tornou uma atividade altamente lucrativa para empresas especializadas em anúncios.

Empresas de publicidade on-line atuam em parceria com vários sites para coletar dados dos usuários durante a navegação e assim constroem um perfil detalhado dos interesses e atividades desses usuários. Mesmo que alguns usuários não vejam problemas nisso (que mal existe em receber anúncios on-line relevantes?), a posse de dados de usuários, coletados sem consentimento, por empresas de publicidade on-line ou mesmo por qualquer empresa ou agência governamental que queria comprá-los, traz consequências potencialmente desastrosas para a privacidade das pessoas e, em casos mais graves, pode envolvê-las em roubos, fraudes e ataques maliciosos.

Várias tentativas de solucionar o problema foram propostas. Em 1997, um padrão para o uso de Cookies foi especificado na RFC 2109 [Kristol and Montulli 1997], no qual proibia o uso de Cookies de terceiros. Embora, na época, nenhum dos dois navegadores tradicionais, Netscape *Navigator* e Internet Explorer, tenham adotado a especificação, o

[1]Whois é um protocolo voltado para consulta de informações sobre domínios

padrão vingou e hoje, por exemplo, um recente estudo sobre Cookies, feito pela comScore [Weinberger 2011], mostra que aproximadamente um em cada três usuários apagam os Cookies dentro de até um mês após visitarem um site.

Em 2005, os desenvolvedores de navegadores começaram a adicionar um modo de navegação privada em seus produtos. Tal solução dava aos usuários a opção de visitar sites sem deixar Cookies de longo prazo. Desenvolvedores independentes também começaram a produzir extensões para preservar a privacidade dos usuários. AdBlock Plus[2], uma extensão para o navegador Firefox que rejeita anúncios e Cookies de terceiros é um bom exemplo desse tipo de solução. Hoje, ferramentas mais modernas como Ghostery[3] e Lightbeam[4] revelam o número de rastreadores existentes em cada site e mostram como esses rastreadores colaboram com locais (sites) aparentemente não relacionados.

Entretanto, os interessados em continuar nesse "ramo de atividade" não ficaram parados. Aproveitando-se do surgimento e proliferação dos dispositivos móveis, com possibilidades de comunicação e uso cada vez maiores, eles mudaram a forma de rastreamento. Agora não é mais necessário usar servidores Web que deixam rastros (migalhas) na máquina do usuário. Ao invés disso, resolveram apostar em ferramentas de reconhecimento/rastreio de usuários através de seu próprio conjunto de hardware/software. Esse conceito é conhecido como *Device Fingerprinting* e parte do pressuposto que cada usuário opera o seu próprio hardware. Assim, a identificação de um dispositivo é o mesmo que identificar a pessoa por trás dele. Segundo [Nikiforakis et al. 2014], *device fingerprinting* faz uso de um conjunto de atributos do sistema, extraídos do dispositivo do usuário, que geram uma combinação de valores únicos capazes de identificar um usuário/dispositivo.

As técnica de *device fingerprinting* estão se tornando comuns e, ainda assim, muitos usuários sabem pouco ou quase nada sobre o assunto. Mesmo quando estão cientes de que estão sendo monitorados, por exemplo, como uma medida de proteção contra a fraude, eles, em essência, simplesmente confiam que as informações coletadas não serão utilizadas para outros fins.

Assim, o objetivo deste Capítulo é desmarcar o mundo por trás do *device fingerprinting*. A ideia é apresentar conceitos, tecnologias e métodos de desenvolvimento, bem como o alcance e o consequente perigo representado por tais técnicas. Além de possibilitar a compreensão dos principais problemas e desafios a serem superados no desenvolvimento e captura de informações, especialmente via Web, através de técnicas de *device fingerprinting*, este Capítulo identifica oportunidades de estudos na área, bem como aponta alguns trabalhos futuros, visto que o emprego de tais técnicas deve crescer ainda mais nos próximos anos [Tanner 2013].

Para alcançar os objetivos propostos, o restante deste Capítulo está organizado da seguinte forma. A Seção 2.2 caracteriza *device fingerprinting*, apresentando conceitos, definições, classificaçoes e uma breve discussão sobre os problemas de privacidade causados. A Seção 2.3 trata do ambiente porta de entrada para um *device fingerprinting*, o navegador. Questões como a guerra entre os fabricantes, a arquitetura e como é possível identificar um navegador são relatadas. Na Seção 2.4 são discutidas as principais tec-

[2]https://adblockplus.org

[3]https://www.ghostery.com/

[4]https://addons.mozilla.org/pt-br/firefox/addon/lightbeam/

nologias empregadas (ou melhor dizendo, alvo) em um *device fingerprinting*. HTML5, JavaScript, Flash, Canvas, CSS, WebGL e Silverlight são avaliadas em termos de funcionalidades e potencial para obtenção de informações sem consentimento do usuário. Exemplos e trechos de código também são apresentados.

Embora o foco deste Capítulo não seja ensinar o leitor a realizar atividades de *fingerprinting*, a Seção 2.5 apresenta um roteiro para se construir um ferramenta simples de *device fingerprinting*. Como prova de conceito, são expostos alguns resultados obtidos de um site público de *device fingerprinting*. A Seção 2.6 discute as soluções existentes que realizam *fingerprinting*, onde trabalhos acadêmicos e soluções comerciais são apresentadas. Já a Seção 2.7 faz o caminho inverso. Apresenta as contramedidas contra *fingerprinting*. Esta seção tem o intuito de apresentar ferramentas, aplicações e soluções disponíveis (implementadas) na contenção de *device fingerprinting*. Por fim, a seção 2.8 apresenta os comentários finais sobre o tema e aponta algumas questões em aberto.

## 2.2. Device Fingerprinting

O termo *fingerprinting* ganhou força na área da computação nos anos de 1990 com o surgimento de várias ferramentas especializadas em realizar ataques a redes de computadores. Isso porque percebeu-se que para efetuar um ataque bem sucedido era necessário descobrir/identificar corretamente a máquina (computador/servidor) alvo, o sistema operacional que ela executava e os aplicativos ativos. Ferramentas como o Nmap[5] surgiram nessa época e são bastante utilizadas até hoje.

Num âmbito mais voltado a Web e foco deste Capítulo, o termo *fingerprinting* veio a tona em 2009, quando Mayer [Mayer 2009] observou que as características de um navegador e seus plug-ins podiam ser identificadas e os usuários rastreados, e quando, em 2010, o trabalho de Peter Eckersley [Eckersley 2010] mostrou como informações (atributos) fornecidas pelo navegador dos usuários eram suficientes para identificar a grande maioria das máquinas que navegam na Internet. Eckersley desenvolveu um algoritmo para investigar o grau em que os navegadores modernos estão sujeitos as técnicas de *device fingerprinting*. Dos mais de 470.000 usuários que participaram de seu projeto público Panopticlick[6], 84% tiveram seus navegadores identificados (“fingerprintados”).

Formalmente, o termo *fingerprint* foi definido na RFC 6973 [Cooper et al. 2013] como “um conjunto de elementos de informação que define um dispositivo ou uma instância de uma aplicação” e *fingerprinting* como “o processo pelo qual um observador ou atacante identifica, de maneira única e com alta probabilidade, um dispositivo ou uma instância de um aplicativo com base em um conjunto de múltiplas informações”. Este Capítulo partilha da visão que *fingerprinting* é parte de um conjunto amplo de tecnologias e técnicas, também conhecidas como *Device Intelligence*, *Machine Fingerprinting*, *Browser Fingerprinting*, *Web Fingerprinting* ou *Device Fingerprinting*, usadas para identificar (ou reidentificar) um usuário ou um dispositivo, através de um conjunto de configurações, atributos (tamanho da tela do dispositivo, versões de software instalado, entre muitos outros) e outras características observáveis durante comunicações. Neste Capítulo, os termos *device fingerprinting* e *fingerprinting* serão usados para representar essas técnicas de

[5]http://nmap.org

[6]http://panopticlick.eff.org

identificação com foco na Web.

Dois aspectos interessantes e relevantes podem ser observados sobre *device fingerprinting*. O primeiro é que embora tenham sido popularizadas no ambiente Web, tais técnicas estão presentes no mundo móvel. Trabalhos como [Giura et al. 2014] e [Goodin 2013] propõem diferentes soluções, métodos e técnicas para identificar um usuário baseado em conjuntos únicos de atributos como números discados, tempo das ligações, conexão com a estação rádio base, entre outras. Segundo, *device fingerprinting* tem um grande potencial de ameaça a privacidade dos usuários na Web. As questões relacionadas aos problemas de privacidade serão discutidas na Seção 2.2.2. É importante também ressaltar que os trabalhos apontam o uso dessas técnicas para engendrar ataques (persistentes e direcionados) [Forshaw 2011, Contextis 2011].

### 2.2.1. Classificação

De acordo com o W3C [W3C 2014], as técnicas de *device fingerprinting* podem ser classificadas em três tipos:

1. **Passivo**: Também chamado de *browser fingerprinting*, é aquele baseado nas características observáveis no conteúdo de solicitações Web, sem a utilização de qualquer código em execução no lado do cliente. Eventualmente, esse tipo de *fingerprinting* inclui Cookies, bem como o conjunto de cabeçalhos de solicitação HTTP, endereço IP e outras informações do nível de rede.

2. **Ativo**: Levam em consideração técnicas onde o site é executado via JavaScript ou por outro código, no lado do cliente, para observar características adicionais sobre o navegador. Técnicas para *fingerprinting* ativo podem incluir o acesso ao tamanho da janela, enumerar fontes ou plug-ins, avaliação das características de desempenho ou os padrões de renderização de gráficos.

3. **Cookie-like**: Nessa categoria, usuários, *user-agents*[7] e dispositivos também podem ser reidentificados por um site que primeiro configura e depois recupera o estado armazenado pelo *user-agent* do navegador ou dispositivo. Permite a reidentificação de um usuário ou inferências sobre um usuário da mesma forma que os Cookies permitem o gerenciamento de estado para o protocolo HTTP (RFC6265 [Barth 2011]). Também podem contornar as tentativas do usuário em limitar ou apagar os Cookies armazenados pelo *user-agent*, como demonstrado em [Kamkar 2010].

### 2.2.2. Privacidade

Embora grande parte do desenvolvimento das técnicas de *device fingerprinting* esteja relacionada a identificação de usuários como medida de segurança e mecanismo anti-fraude (empresas como BlueCava[8], Iovation[9] e ThreatMetrix[10] são bons exemplos dessa finalidade), os impactos na privacidade dos usuários causados por essas técnicas são ameaças reais de segurança. Três problemas na privacidade são apontados em [W3C 2014]:

[7] *user-agent* é um componente do cabeçalho do protocolo HTTP.
[8] http://www.bluecava.com
[9] http://www.iovation.com
[10] http://www.threatmetrix.com

1. **Identificação do usuário**: Até por questões de padronização, a maioria dos usuários preferem ficar anônimos quando navegam na Internet. Segurança física e pessoal, discriminação, sigilo de dados, entre outros, são motivos para os usuários ficarem no anonimato. Mas, um *fingerprinting* tem o potencial de obter esses dados, sem autorização prévia, deixando o usuário exposto a todos esses fatores.

2. **Correlacionar as atividades da navegação**: Problemas com a privacidade ocorrem mesmo que o usuário não seja identificado. Os usuários podem se surpreender ao perceber que terceiros (empresas de publicidade on-line em sua maioria) podem correlacionar suas várias visitas ao mesmo ou diferentes sites para elaborar um perfil do usuário. A preocupação aumenta uma vez que essa invasão de privacidade pode acontecer com ou sem autorização do usuário, sem mencionar que ferramentas, como as que fazem a limpeza de Cookies, não impedem essa correlação.

3. **Inferências sobre o usuário**: As informações coletadas durante um *fingerprinting* podem revelar dados sobre os quais se pode tirar conclusões sobre o usuário. A versão do sistema operacional e informações sobre a CPU são exemplos de dados que podem ser usados para inferir, por exemplo, o poder de compra do usuário. [W3C 2014] afirma que tais inferências permitem oferecer e mostrar ao usuário produtos em determinada faixa de preço, o que pode ser uma forma discriminatória de publicidade. Sem falar que os usuários podem se sentir tratados de forma diferente.

Um exemplo de como o *fingerprinting* é capaz de monitorar e correlacionar atividades de navegação de um usuário, dentro e através de sessões, e coletar informações que podem inferir sobre suas preferências e hábitos é apresentado na Figura 2.1.

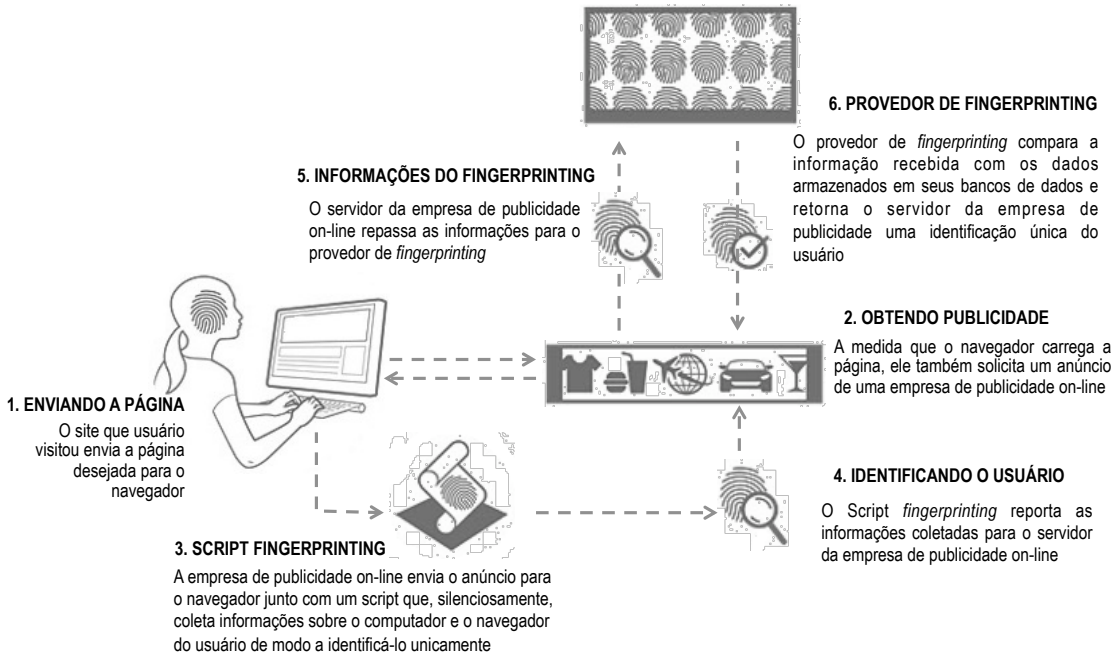



**Figura 2.1. Exemplo do processo de *fingerprinting*. Fonte: [Nikiforakis et al. 2014]**

## 2.3. Navegador: tudo começa com ele

Nos dias de hoje, é impossível negar e contestar a presença dos navegadores Web (normalmente chamados de *Browsers* e a partir deste ponto tratados apenas como navegadores) na rotina dos usuários de qualquer sistema de computação. O motivo é bastante simples. As aplicações Web tornaram-se tão populares a ponto de rivalizar com software e aplicações nativas. Neste contexto, os navegadores tornaram-se a interface dominante (mecanismo padrão) de conexão dos usuários com sistemas e aplicações Web, bem como conteúdos e serviços de seu interesse.

Três fatos comprovam a ascensão dos navegadores. O primeiro é a grande movimentação por parte das fabricantes de Sistemas Operacionais em desenvolverem seus próprios navegadores [Mulazzani et al. 2013]. A Microsoft tem o Internet Explorer, a Apple criou o Safari, o Google criou um sistema operacional, o ChromeOS, baseado inteiramente no navegador Chrome, e a Mozilla criou o FirefoxOS. O segundo é que os fabricantes de navegadores disponibilizam novas versões com uma rapidez impressionante, alguns com intervalos de tempo bem reduzidos e sempre trazendo ou agregando novas funcionalidades, especialmente nas versões voltadas aos dispositivos móveis. O terceiro e último é a grande competitividade no mercado mundial de navegadores. Recente pesquisa da StatCounter [StatCounter 2014] aponta o navegador Chrome como o mais utilizado entre os internautas no período de Janeiro a Agosto de 2014, englobando 46,26% da preferência dos usuários. Em seguida vem o Internet Explorer (IE) com 20,31%, o Firefox com 17,5%, o Safari com 10,81% e o Opera com 1,47%.

A Figura 2.2 apresenta o gráfico de uso dos cinco (5) navegadores mais populares nos últimos cinco (5) anos.

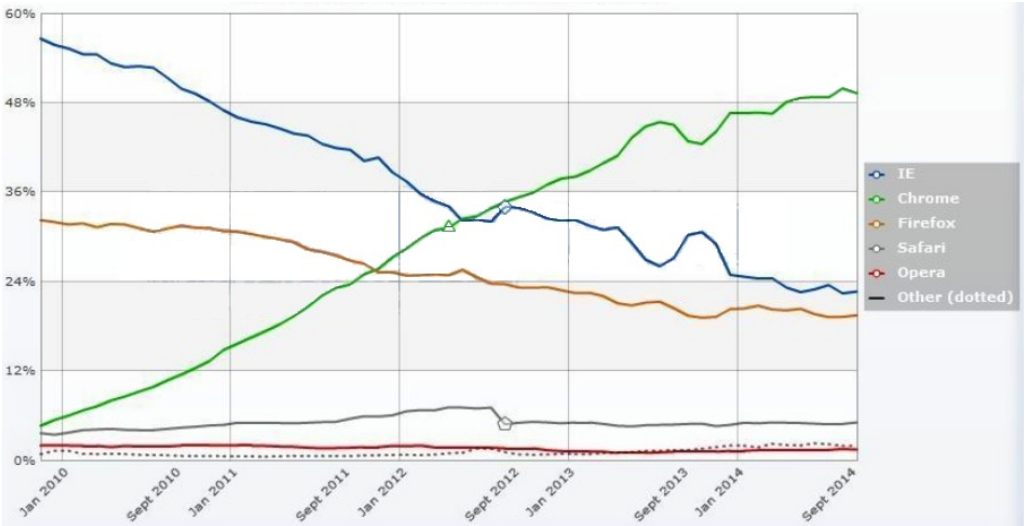



**Figura 2.2. Crescimento dos 5 maiores navegadores nos últimos 5 anos. Fonte: [StatCounter 2014]**

No que diz respeito ao *device fingerprinting*, o problema com os navegadores é a falta de padrão. Toda a interação e a forma como o navegador interpreta e exibe os recursos solicitados pelo usuário segue padrões definidos e mantidos pelo W3C (*World Wide*

*Web Consortium*)[11]. Contudo, os fabricantes de navegadores mantiveram-se, por muito tempo, parcialmente de acordo com essas especificações, o que lhes permitiu desenvolver suas próprias extensões e com isso agregar novas funcionalidades. Embora hoje a maioria dos navegadores estejam relativamente de acordo com as especificações, essa diferença nos padrões causa sérios problemas de compatibilidade. Além da questão estética, a não completa adoção dos padrões trás implicações relevantes na segurança e privacidade dos usuários e informações. É exatamente neste ponto que entram ou se encaixam as técnicas de *device fingerprinting*.

### 2.3.1. Arquitetura do Navegador

A arquitetura de um navegador une diversas funcionalidades de forma a tornar capaz de apresentar as páginas Web, mas, para que isso ocorra, alguns itens são imprescindíveis. Usando a Figura 2.3 como modelo, os componentes de um navegador são [CCEE 2012]: Interface de Usuário, Motor de Navegação, Motor de Renderização, Networking, Motor de Interpretação de JavaScript, UI Backend e Data Storage.

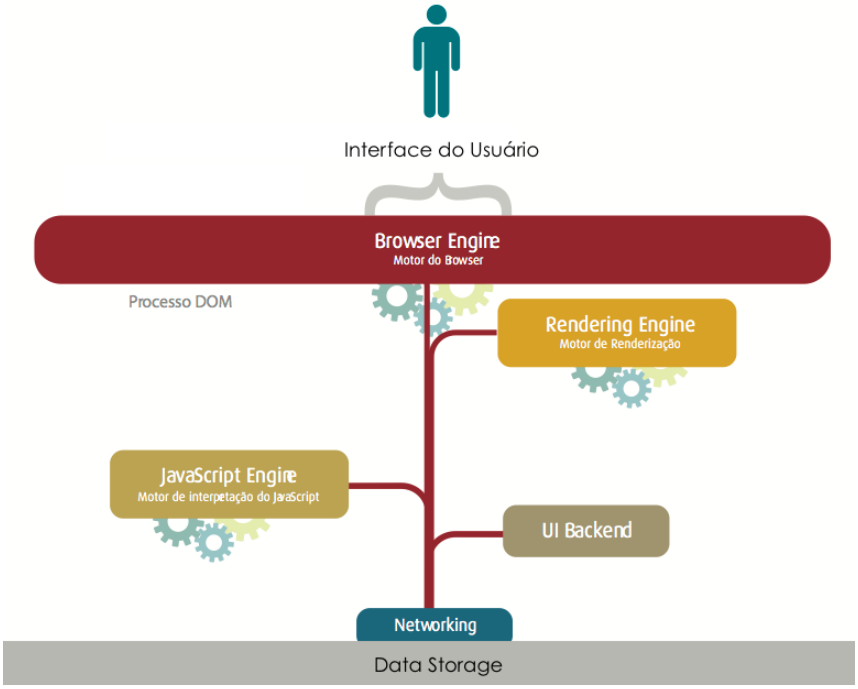



**Figura 2.3. Arquitetura simplificada de um navegador. Fonte [CCEE 2012]**

A **Interface de Usuário** é o espaço de interação entre os usuários e o navegador. Barra de endereços, botões de avanço e retorno, página inicial, atualização, favoritos, entre outros, são os elementos da interface do usuário (UI). Em outras palavras, tudo o que o navegador exibe faz parte desse componente, exceto pela janela onde o usuário vê a página requisitada.

O **Motor de Navegação** (*Browser Engine*) é o encarregado pela comunicação das entradas da interface do usuário em conjunto com o motor de renderização (*Rendering*

[11] http://www.w3c.org

*Engine*). O papel do motor de navegação é consultar e manipular o motor de renderização, de acordo com as requisições que ocorrem entre a aplicação e a interface de usuário.

O **Motor de Renderização** é o componente responsável por exibir o conteúdo solicitado na tela do navegador. Também chamado de motor de layout, tem, por padrão, a função de exibir documentos HTML, XML e imagens. Pode também exibir outros tipos de dados através de plug-ins ou extensões, isto é, para exibir documentos PDF é preciso um plug-in visualizador de PDF. Os navegadores usam diferentes motores de renderização. O Internet Explorer usa Trident, o Firefox usa Gecko, o Safari usa WebKit. O Chrome (a partir da versão 28) e o Opera (a partir da versão 15) usam Blink, um fork do WebKit. Por fim, o motor de renderização está interligado com execuções do interpretador de JavaScript em processos que ocorrem em tempo de execução.

**Networking** é a parte do código do navegador responsável por gerenciar as chamadas de rede, como, por exemplo, o envio de solicitações HTTP para o servidor.

O **Motor de Interpretação de JavaScript** é usado para interpretar e executar códigos JavaScript.

**UI Backend** é a parte do código usada para desenhar os elementos de design básicos do navegador como caixas de combinação e janelas. Este *backend* expõe uma interface genérica que não é plataforma específica. Ela usa métodos de interface de usuário do sistema operacional.

O **Data Storage** é um componente persistente onde o navegador pode salvar todos os tipos de dados locais, como arquivos diversos, cache, Cookies, entre outros. Os navegadores também suportam mecanismos de armazenamento tais como banco de dados indexados, WebSQL e sistemas de arquivo.

### 2.3.2. Como identificar um navegador?

Existem várias formas de se identificar um navegador. Esta seção aborda apenas duas das mais simples técnicas capazes desse feito.

#### *User-agent*

Para descobrir informações sobre o navegador (e sobre a máquina que o hospeda), a ideia mais simples é pedir ao próprio navegador para revelar "voluntariamente" sua verdadeira identidade. O mecanismo mais simples para isso é o *user-agent*, um componente do cabeçalho do protocolo HTTP.

De acordo com a RFC 2616 [Fielding et al. 1999], *user-agent* é um campo enviado quando o navegador faz a solicitação de uma página para: (i) fins estatísticos; (ii) descobrir violações no protocolo; (iii) reconhecimento automatizado do *user-agent* para evitar limitações específicas e melhor exibir o conteúdo solicitado.

O *user-agent* é uma sequência de caracteres composta por um conjunto de elementos, chamados *tokens*, listados em ordem de importância de acordo com o padrão. Uma típica resposta ao *user-agent* de um navegador é a seguinte:

**Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.78.2 Version/7.0.6 Safari/537.78.2**

A Tabela 2.1 explica os dados obtidos do *user-agent* anterior.

**Tabela 2.1. Campos de um *user-agent***

| Token | Descrição |
|---|---|
| Nome da Aplicação | Mozilla |
| Versão | 5.0 |
| Plataforma | Macintosh |
| Sistema Operacional | Intel Mac OS X 10_9_4 (Versão 10_9_4 ) |
| Motor de Layout | AppleWebKit (build 537.78.2) |
| Versão do Navegador | 7.0.6 |
| Nome do Navegador | Safari (build 537.78.2) |

Como forma de proteção, nem sempre muito eficaz, os navegadores utilizam extensões que permitem alterar (falsificar) determinadas configurações enviadas pelo *user-agent*, uma vez que é uma sequência string. Para o Internet Explorer existe o plug-in *UAPick*[12]. Para o Firefox, o U*ser Agent Switcher*[13] é o plug-in mais usado. A mesma extensão é usada no Chrome[14]. O Safari possui o *User Agent Browser*[15]. No Opera, o *User Agent Changer* é a extensão que permite manipular o *user-agent*.

Na prática, *user-agent* é mais utilizado através do objeto *Navigator*.

### Objeto *Nagivator*

Outra forma de identificar um navegador é utilizar o objeto *Navigator*. Para exibir o conteúdo de uma página, o navegador cria automaticamente uma hierarquia de objetos DOM (*Document Object Model*) refletindo alguns elementos inseridos na página. Existem três objetos básicos: *Screen*, *Window* e *Navigator*. Este último é o que representa o próprio navegador e através dele é possível controlar seu comportamento, além de obter informações sobre suas características. A especificação do HTML5 [W3C 2014] diz que qualquer informação sobre o navegador é feita através do atributo *navigator* da interface *Window*, que deve retornar uma instância da interface *Navigator*.

A Listagem 2.1 apresenta um simples exemplo, em JavaScript, de como obter dados para *device fingerprinting* via objeto *Navigator*. No exemplo, são obtidas informações do *user-agent*, da linguagem do navegador, da plataforma e a classe da CPU. Esta última propriedade só funciona para navegadores IE, o que é uma característica que permite a fácil identificação do navegador.

**Listagem 2.1. JavaScript para obter os dados do *user-agent***

```
1  function fingerprint_useragent() {
2     "use strict";
3     var strSep, strTmp, strUserAgent, strOut;
4
5     strSep = "|";
6     strTmp = null;
7     strUserAgent = null;
8     strOut = null;
```

[12]http://www.enhanceie.com/ietoys/uapick.asp

[13]https://addons.mozilla.org/pt-BR/firefox/addon/user-agent-switcher/

[14]https://chrome.google.com/webstore/detail/user-agent-switcher-for-c/djflhoibgkdhkhhcedjiklpkjnoahfmg

[15]https://itunes.apple.com/br/app/user-agent-browser-simply/id414229165?mt=8

```
9
10      /* navigator.userAgent is supported by all major browsers */
11      strUserAgent = navigator.userAgent.toLowerCase();
12      /* navigator.platform is supported by all major browsers */
13      strTmp = strUserAgent + strSep + navigator.platform;
14      /* navigator.cpuClass only supported in IE */
15      if (navigator.cpuClass) {
16         strTmp += strSep + navigator.cpuClass;
17      }
18      /* navigator.browserLanguage only supported in IE, Safari and Chrome */
19      if (navigator.browserLanguage) {
20         strTmp += strSep + navigator.browserLanguage;
21      } else {
22         strTmp += strSep + navigator.language;
23      }
24      strOut = strTmp;
25      return strOut;
26   }
```

As propriedades do objeto *Navigator* são exibidas na Tabela 2.2. É importante ressaltar que todas essas propriedades são somente de leitura (*ready-only*).

**Tabela 2.2. Propriedades do objeto *Navigator***

| **Propriedades** | **Descrição** |
|---|---|
| appCodeName | Retorna o codinome do navegador. |
| appName | Retorna uma string DOM com o nome real do navegador |
| appVersion | Retorna uma string DOM com a versão do navegador |
| *user-agent* | Retorna o cabeçalho do *user-agent* enviado pelo navegador para o servidor |

## 2.4. Tecnologias empregadas

Os navegadores se tornaram as plataformas mais sofisticadas para execução de aplicações, assumindo mais funcionalidades do que as tradicionalmente fornecidas pelo sistema operacional [Mowery and Shacham 2012]. Grande parte deste aumento de sofisticação foi impulsionado por tecnologias e conjuntos de especificações que fornecem a capacidade de desenhar dinamicamente em área da tela (<Canvas>), gráficos tridimensionais (WebGL), armazenamento de dados estruturados do lado do cliente, serviços de geolocalização, capacidade de manipular o histórico e o cache do navegador, reprodução de áudio e vídeo, e muito mais.

Esta seção irá explicitar as principais tecnologias utilizadas nos navegadores, dentre as quais destacam-se: JavaScript, Flash Player, Canvas, WebGL, CSS e Silverlight, todas alvo de *device fingerprinting*. Contudo, antes de iniciar a apresentação de cada uma delas, é preciso falar sobre o padrão HTML5.

### 2.4.1. HTML5

A HTML5 é a quinta versão do padrão HTML (*Hypertext Markup Language*), que desde de dezembro 2012 recebeu o status de "Candidata a Recomendação" do W3C. Assim como suas sucessoras, a principal finalidade da HTML5 é estruturar o conteúdo que se apresenta na Web, através do suporte as mais recentes tecnologias multimídia. Este novo

padrão engloba não só a HTML, mas também a XHTML1 [Pemberton 2002] e o HTML DOM Nível 2 [Hors et al. 2003]. Por isso é vista como uma tentativa de definir uma única linguagem de marcação que pode ser escrita em qualquer uma das sintaxes mencionadas anteriormente.

O padrão HTML5 define novos elementos, atributos e comportamentos, bem como possui suporte a um conjunto bem maior de tecnologias, o que permite o desenvolvimento de aplicações Web e sites mais dinâmicos e poderosos. Embora as especificações da HTML5 ainda não tenham sido finalizadas e estejam sujeitas as mudanças, os principais navegadores, especialmente o Mozilla, começaram a implementar partes deste padrão. Pelas mesmas razões já citadas, HTML5 também é um candidato em potencial para aplicações móveis multiplataforma. Muitos recursos da HTML5 foram construídos com o requisito de serem capazes de executar em dispositivos de baixa potência, como smartphones e tablets.

A Figura 2.4 exemplifica as tecnologias e mudanças no HTML5.

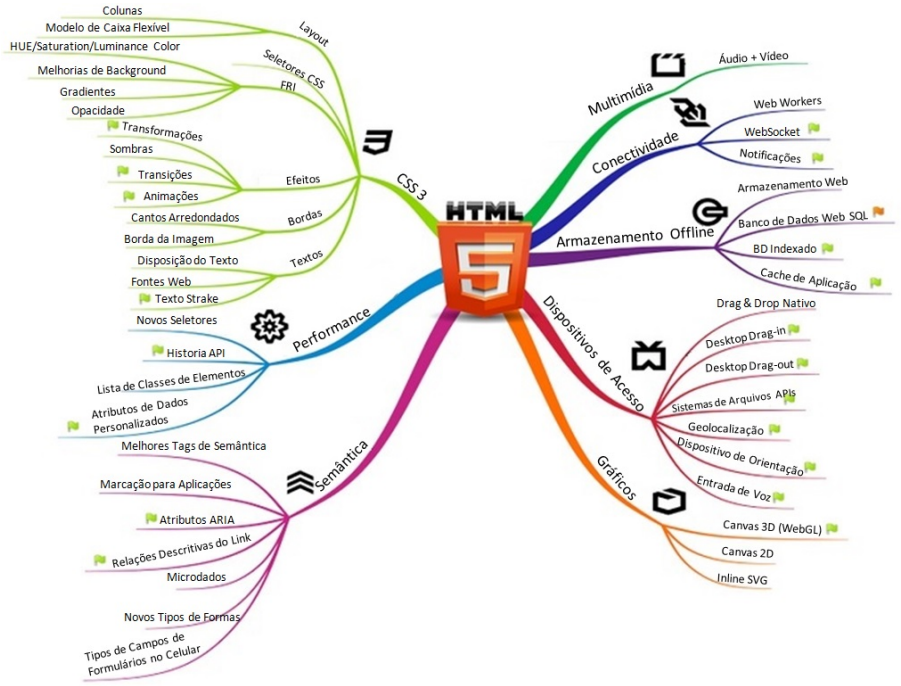



**Figura 2.4. Funcionalidades e Características do HTML5**

Contudo, as funções melhoradas da HTML5, visando fornecer aos usuários a experiência de um aplicativo nativo em um tempo significativamente menor e a um bom custo, também trouxeram problemas como *fingerprinting*. A título de exemplo, a Listagem 2.2 apresenta dois trechos de código, em JavaScript, que utilizam uma biblioteca capaz de detectar o suporte a muitas das características do HMTL5 e CSS3, a Modernizr [Modernizr 2014].

O primeiro trecho verifica se o navegador tem suporte para trabalhar off-line, ou seja, se aplicações Web podem operar off-line, sem conexão com a Internet. Na prática, a capacidade de operar off-line começa como aplicações Web on-line. A primeira vez que

o navegador visita um site habilitado para off-line, o servidor Web informa ao navegador quais arquivos que ele precisa para trabalhar off-line. Uma vez que o navegador baixou todos os arquivos necessários, o site pode ser acessado (revisado) mesmo sem conexão com à Internet.

O segundo trecho verifica se o navegador suporta a API de geolocalização. Se sim, haverá uma propriedade de geolocalização no objeto *Navigator*.

**Listagem 2.2. Trechos de exemplos para checar o suporte a propriedades do HMTL5**

```
if (Modernizr.applicationcache) {
  // window.applicationCache is available!
} else {
  // no native support for off-line :(
  // try a fallback or another third-party solution
}
...
if (Modernizr.geolocation) {
  // let's find out where you are!
} else {
  // no native geolocation support available :(
  // try geoPosition.js or another third-party solution
}
```

### 2.4.2. JavaScript

O JavaScript é uma linguagem de programação interpretada, implementada como parte dos navegadores para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor [Crockford 2008]. Desenvolvida pela Netscape Communications Corp., por Brendan Eich, possui uma sintaxe similar ao Java e ao C++ e é orientada a objetos, o que permite tratar todos os elementos da página como objetos distintos, facilitando a tarefa da programação.

O JavaScript pode ser ativado ou desativado nos navegadores. Assim, uma vez desativado poderá prevenir técnicas de *fingerprinting*. Entretanto, ao se desativar o JavaScript, partes do conteúdo de sites, como vídeos e gráficos interativos, deixam de ser exibidas.

Mas como informações do usuário podem ser obtidas com o Javascript habilitado? De modo geral, através de um objeto DOM pode-se revelar informações importantes a respeito do navegador, tais como o *user-agent*, a arquitetura, o idioma do sistema operacional, o tempo do sistema, a resolução de tela, entre outros.

As Listagens 2.3 e 2.4 apresentam duas simples funções de exemplo do uso de JavaScript para *device fingerprinting*. A primeira descobre se a conexão está sendo feita via um Proxy Web, uma tentativa de garantir anonimato na Internet. A segunda lista os plug-ins instalados no navegador.

**Listagem 2.3. JavaScript para detecção de Proxy Web**

```
var our_proto = "https";
var our_host = "zorrovpn" + "." + "com"; // mask address
var our_request = "show-js-ip";

// Detect web-proxy version
var webproxy = "No";

if (window["_proxy_jslib_SCRIPT_URL"]) {
```

```
  webproxy = "CGIProxy (" + window["_proxy_jslib_SCRIPT_URL"] + ")";
} else if (window["REAL_PROXY_HOST"]) {
  webproxy = "Cohula (" + window["REAL_PROXY_HOST"] + ")";
} else if (typeof ginf != 'undefined') {
  webproxy = "Glype (" + ginf.url + ")";
} else if (window.location.hostname != our_host) {
  webproxy = "Unknown (" + window.location.hostname + ")";
}

// Trick for CGIProxy
window["_proxy_jslib_THIS_HOST"] = our_host;
window["_proxy_jslib_SCRIPT_NAME"] = "/" + our_request + "?#";
window["_proxy_jslib_SCRIPT_URL"]
  = our_proto + "://" + our_host + "/" + window["_proxy_jslib_SCRIPT_NAME"];

document.write("<b>Detected IP address:</b> ");
document.write('<script sr');
document.write('c="' + our_proto + ":" + '//' + our_host + '/' + our_request);
document.write('"><\/script>');
document.write("<br><b>Web-proxy:</b> " + webproxy);
```

**Listagem 2.4. JavaScript para enumerar e listar os plug-ins instalados**

```

var plug-ins = (function(){
    var found = {};
    var version_reg = /[0-9]+/;

    /* Differentiate between IE (detection via ActiveXObject)
     * and the rest (detection via navigator.plug-ins) */
    if (window.ActiveXObject) {
        var plug-in_list = {
            flash: 'ShockwaveFlash.ShockwaveFlash.1',
            pdf: 'AcroPDF.PDF',
            silverlight: 'AgControl.AgControl',
            quicktime: 'QuickTime.QuickTime'
        }

        for (var plug-in in plug-in_list){
            var version = msieDetect(plug-in_list[plug-in]);
            if (version){
                var version_reg_val = version_reg.exec(version);
                found[plug-in] = (version_reg_val && version_reg_val[0]) || '';
            }
        }

        if (navigator.javaEnabled()){
            found['java'] = '';
        }
    } else {
        var plug-ins = navigator.plug-ins;
        var reg = /Flash|PDF|Java|Silverlight|QuickTime/;
        for (var i = 0; i < plug-ins.length; i++) {
            var reg_val = reg.exec(plug-ins[i].description);
            if (reg_val){
                var plug-in = reg_val[0].toLowerCase();
                /* Search in version property, if not available concat name
                 *  and description and search for a version number in there   */
                var version = plug-ins[i].version ||
                    (plug-ins[i].name + ' ' + plug-ins[i].description);
                var version_reg_val = version_reg.exec(version);
                if (!found[plug-in]) {
                    found[plug-in] = (version_reg_val && version_reg_val[0]) || '';
                }
            }
        }
    }

    return found;

```

```
48      /* Return version number if plug-in installed
49       * Return true if plug-in is installed but no version number found
50       * Return false if plug-in not found */
51      function msieDetect(name){
52          try {
53              var active_x_obj = new ActiveXObject(name);
54              try {
55                  return active_x_obj.GetVariable('$version');
56              } catch(e) {
57                  try {
58                      return active_x_obj.GetVersions();
59                  } catch (e) {
60                      try {
61                          var version;
62                          for (var i = 1; i < 9; i++) {
63                              if (active_x_obj.isVersionSupported(i + '.0')){
64                                  version = i;
65                              }
66                          }
67                          return version || true;
68                      } catch (e) {
69                          return true;
70                      }
71                  }
72              }
73          } catch(e){
74              return false;
75          }
76      }
77  }
```

Dentre os argumentos para empregar JavaScript em *device fingerprinting*, pode-se citar: (i) JavaScript é uma tecnologia bem estabelecida, padronizado como ECMAScript [ECMA International 2011]; (ii) é suportada por todos os principais navegadores; (iii) funciona em dispositivos móveis; (iv) é utilizado por um percentual muito grande de sites; e (v) é ativado por padrão em todos os principais navegadores. Comparada com outras abordagens de identificação, o uso do JavaScript é mais robusto e não pode ser facilmente portada de um navegador para outro.

Recentemente, dois trabalhos utilizaram JavaScript de forma diferente para obter informações sobre o navegador e os usuários. Em [Mowery et al. 2011], os autores implementaram e avaliaram a identificação do navegador através da análise do desempenho e do tempo de execução de JavaScript. Para tanto, utilizaram uma combinação de 39 diferentes *benchmarks* JavaScript, bem conhecidos e bem estabelecidos, e geraram um *fingerprinting* normalizado a partir de padrões de tempo de execução. Como resultado da classificação, dos 1015 casos de teste, 810 foram classificados corretamente, permitindo uma acurácia de 79.8% na identificação do navegador.

Já o trabalho de Mulazzani et al. [Mulazzani et al. 2013] propôs uma pesquisa voltada a identificar a segurança de um navegador baseado no uso de um mecanismo de *fingerprinting* do JavaScript. Para tanto, os autores utilizaram o Test262[16], um conjunto de testes oficiais para ECMAScripts. Foram usados 11.148 casos de teste únicos para navegadores de desktop e 11.570 casos de teste para navegadores móveis. Os autores concluíram que um único caso de teste pode ser suficiente para distinguir dois navegadores específicos. Para tanto, basta um dos navegadores falhar em um caso particular de teste e o outro não. No exemplo apresentado no artigo, o Opera versão 11.64 só falhou

[16]http://test262.ecmascript.org

em 4 dos mais de 10 mil casos testados, enquanto o Internet Explorer 9 falhou em quase 400 casos de testes.

### 2.4.3. Flash Player

O Adobe Flash Player é um padrão para entrega de rico conteúdo Web, no intuito de atrair e envolver os usuários. Através dele é possível exibir animações e interfaces de aplicativos, que são implantadas imediatamente em todos os navegadores e plataformas. Dentre as principais características relacionadas ao plug-in Adobe Flash Player, pode-se destacar: (i) a entrega de um console de jogos com qualidade para o navegador, (ii) a produção de impressionantes experiências de mídia e (iii) a implantação de conteúdo dinâmico em um tempo de execução mais seguro.

Apesar dessas características atrativas aos usuários, Adobe Flash Player é uma prato cheio para *device fingerprinting*. O estudo de Soltani et al. [Soltani et al. 2010] observou um abuso de Cookies Flash, também chamados de objetos em locais compartilhados (LSO - *Local Shared Objects*), uma vez que Cookies HTTP previamente removidos foram regenerados. A técnica ficou conhecida como *respawning* (reaparecimento). No trabalho, 54 dos 100 mais populares sites (de acordo com a Quantcast) armazenavam Cookies Flash. Além disso, eles analisaram o *respawning* e descobriram que vários sites, incluindo aol.com, regeneravam Cookies HTTP previamente removidos através de Cookies Flash.

A Figura 2.5 ilustra as etapas do processo de *respawning* de Cookies Flash. Sempre que um usuário visita um site que usa Cookies, o site emite um ID e o armazena em vários mecanismos de armazenamento, incluindo Cookies, LSOs e armazenamento local. Na Figura 2.5a, o valor 123 é armazenado em Cookies HTTP e Flash. Quando o usuário remove os Cookies HTTP (Figura 2.5b), o site reaparece com uma cópia do Cookie com o mesmo valor (123) através da leitura do valor (ID) em um Cookie Flash que o usuário pode não conseguir remover (Figura 2.5c).

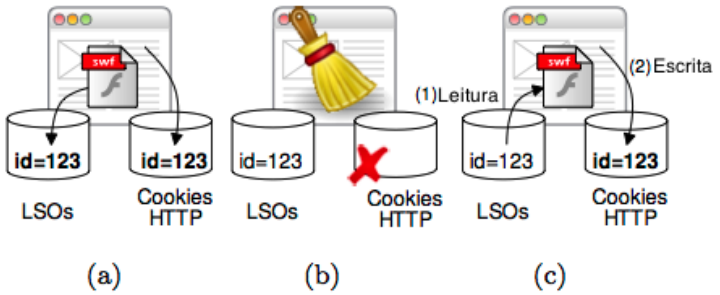



**Figura 2.5. Exemplo de *Respawning*: (a) a página Web armazena um Cookie HTTP e um Cookie Flash (LSO), (b) o usuário remove o Cookie HTTP, (c) a página Web "reaparece" com o Cookie HTTP copiando o valor do Cookie Flash. Fonte: [Acar et al. 2014]**

Embora o uso de Cookies Flash tenha sofrido um declínio de uso e os sites que o empregam não sejam mais identificados de forma global, o *fingerprinting* baseado em Flash ainda é ameaça [McDonald and Cranor 2011]. De modo geral, se um *fingerprinting* conseguir mais ou menos 15 ou 20 bits de informações da identificação, ele pode ter dados

suficientes para identificar o navegador e seu endereço IP.

A Listagem 2.5 representa um ActionScript para determinar o Timezone do computador em Flash.

**Listagem 2.5. ActionScript para determinar o Timezone em Flash**

```
1  public class TimeZoneUtil
2      {
3          import com.adobe.utils.DateUtil;
4
5          /**  List of timezone abbreviations and matching GMT times. **/
6          private static var timeZoneAbbreviations:Array = [
7              ...
8              /* Atlantic Standard/Daylight Time */
9              {abbr:"AST", zone:"GMT-0400"},
10             {abbr:"ADT", zone:"GMT-0300"},
11             ...     ];
12
13         /** Return local system timezone abbreviation.**/
14         public static function getTimeZone():String
15         {
16             var nowDate:Date = new Date();
17             var DST:Boolean = isObservingDTS();
18             var GMT:String = buildTimeZoneDesignation(nowDate, DST);
19
20             return parseTimeZoneFromGMT(GMT);
21         }
22
23         /** Determines if local computer is observing daylight savings time for US
               and London. **/
24         public static function isObservingDTS():Boolean
25         {
26             var winter:Date = new Date(2011, 01, 01); // after daylight savings time
                   ends
27             var summer:Date = new Date(2011, 07, 01); // during daylight savings time
28             var now:Date = new Date();
29
30             var winterOffset:Number = winter.getTimezoneOffset();
31             var summerOffset:Number = summer.getTimezoneOffset();
32             var nowOffset:Number = now.getTimezoneOffset();
33
34             if((nowOffset == summerOffset) && (nowOffset != winterOffset)) {
35                 return true;
36             } else {
37                 return false;
38             }
39         }
40
41         /** Goes through the timze zone abbreviations looking for matching GMT time.
               **/
42         private static function parseTimeZoneFromGMT(gmt:String):String
43         {
44             for each (var obj:Object in timeZoneAbbreviations) {
45                 if(obj.zone == gmt){
46                     return obj.abbr;
47                 }
48             }
49             return gmt;
50         }
51
52         /** Method to build GMT from date and timezone offset and accounting for
               daylight savings. **/
53         private static function buildTimeZoneDesignation( date:Date, dts:Boolean  ):
               String
54         {
55             if ( !date ) {    return "";    }
56
57             var timeZoneAsString:String = "GMT";
```

```
58              var timeZoneOffset:Number;
59
60              // timezoneoffset is the number that needs to be added to the local time
                    to get to GMT, so
61              // a positive number would actually be GMT -X hours
62              if ( date.getTimezoneOffset() / 60 > 0 && date.getTimezoneOffset() / 60 <
                     10 ) {
63                  timeZoneOffset = (dts)? ( date.getTimezoneOffset() / 60 ):( date.
                        getTimezoneOffset() / 60 - 1 );
64                  timeZoneAsString += "-0" + timeZoneOffset.toString();
65              } else if ( date.getTimezoneOffset() < 0 && date.timezoneOffset / 60 >
                    -10 ) {
66                  timeZoneOffset = (dts)? ( date.getTimezoneOffset() / 60 ):( date.
                        getTimezoneOffset() / 60 + 1 );
67                  timeZoneAsString += "+0" + ( -1 * timeZoneOffset ).toString();
68              } else {
69                  timeZoneAsString += "+00";
70              }
71
72              // add zeros to match standard format
73              timeZoneAsString += "00";
74              return timeZoneAsString;
75          }
76      }
```

### 2.4.4. HTML5 Canvas

Canvas é um novo elemento da HTML5 que fornece uma área da tela que pode ser utilizada via programação. Através de JavaScript, Canvas permite o acesso a um conjunto completo de funções de desenho, permitindo que gráficos sejam gerados dinamicamente. Os autores em [Fulton and Fulton 2011] destacam que Canvas é o equivalente a uma lona utilizada por artistas como superfície de pintura.

A Listagem 2.6 ilustra um código em HTML5 Canvas [Slivinski 2011], cujo resultado é um quadrado vermelho e um texto escrito “Hello World” (Figura 2.6).

**Listagem 2.6. Exemplo de Código contendo HTML5 Canvas**

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Canvas Test</title>
5      <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
6    </head>
7    <bod>
8      <Canvas id="Canvas" width="200" height="200">
9        Se você estiver vendo isso, o seu navegador não suporta WebGL.
10     </Canvas>
11     <script type="text/javascript">
12       var Canvas = document.getElementById(?Canvas?);
13       var context = Canvas.getContext(?2d?);
14       context.fillStyle = "rgb(255, 0, 0)";
15       context.fillRect(30, 30, 50, 50);
16       context.font = "20px serif";
17       context.fillStyle = "rgb(0, 0, 255)";
18       context.fillText("Hello World", 100, 100);
19     </script>
20   </body>
21 </html>
```

Dentre os principais recursos que o HTML5 Canvas disponibiliza, pode-se destacar:

- **Interatividade**: Canvas pode responder às ações do usuário, através dos eventos de

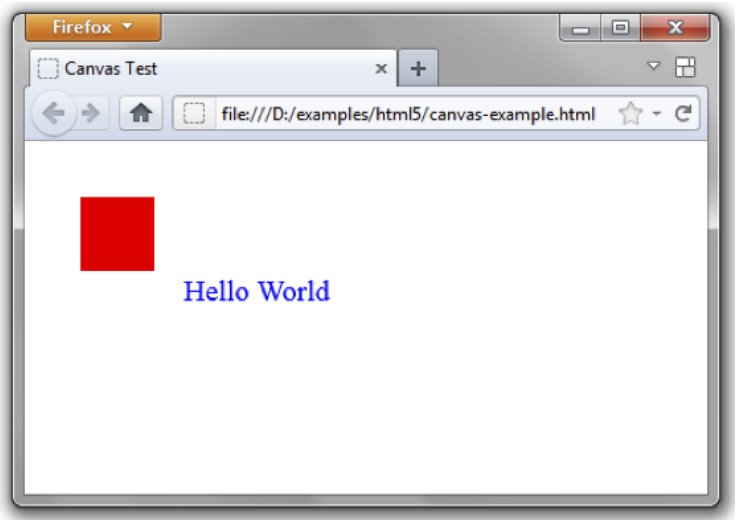



**Figura 2.6. Resultado da execução do Código contendo HTML5 Canvas. Fonte: [Slivinski 2011]**

teclado, o mouse ou toque;

- **Animação**: Cada objeto desenhado na tela pode ser animado;
- **Flexibilidade**: É possível que os desenvolvedores criem qualquer coisa;
- **Padrão Web**: Canvas é uma tecnologia aberta que faz parte da HTML5;
- **Portabilidade**: Ao contrário do Flash e Silverlight, uma aplicação com Canvas pode ser executada em praticamente qualquer lugar.

Entretanto, HTML5 Canvas faz parte das tecnologias modernas aptas a atuar em *device fingerprinting*. Estudos iniciais mostram que o Canvas pode fornecer uma identificação única rapidamente. O trabalho de Mowery e Shacham [Mowery and Shacham 2012] observou que é possível relacionar o navegador, com maior intimidade, as funcionalidades do hardware e do sistema operacional. Os autores desenvolveram uma técnica de *fingerprinting* onde quando um usuário visita um site que utiliza a técnica, o navegador é instruído a desenhar uma linha oculta de texto ou de gráfico 3D, que é então convertida em um sinal digital. Desta maneira, as variações nas quais a GPU está instalada ou o driver gráfico causam variações em *tokens* digitais. O *token* pode ser armazenado e compartilhado com empresas de publicidade para identificar os usuários quando eles visitam sites afiliados. Um perfil pode ser criado como atividade de navegação de um usuário, permitindo que anunciantes direcionem sua publicidade de acordo com as preferências do usuário.

O estudo de Kirk [Kirk 2014] relata uma pesquisa que encontrou um código utilizado para *fingerprinting*, usando Canvas, que estava em uso no início deste ano em mais ou menos 5000 sites populares, sem qualquer tipo de conhecimento para seus usuários. Entretanto, nem todos os locais observados com *fingerprinting* faziam o compartilhamento de conteúdo para empresas de publicidade. Ele ressalta ainda que as empre-

sas europeias estão à procura de novas maneiras de entregar publicidade segmentada aos usuários, afastando-se dos Cookies.

Um exemplo ilustrativo acerca desta técnica de *fingerprinting* é apresentado na Figura 2.7.

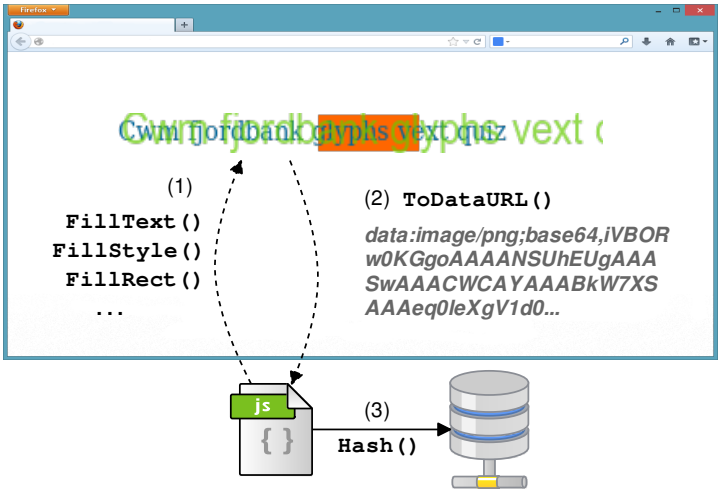



**Figura 2.7. Exemplo de *Fingerprinting* utilizando Canvas. Fonte: [Acar et al. 2014]**

A Figura 2.7 mostra o fluxo de operações do *fingerprinting* com Canvas. Quando um usuário visita uma página, o script *fingerprinting* primeiro desenha um texto com a fonte e o tamanho de sua escolha e acrescenta cores de fundo (1). Em seguida, o script chama o método **ToDataURL**, da API Canvas, para obter os dados de pixel da tela em formato **DataURL** (2), que é basicamente uma representação codificada em Base 64 dos dados de pixel binários. Por fim, o script leva o *hash* dos dados de pixel codificada de texto (3), que serve como *fingerprint* e pode ser combinada com outras propriedades de alta entropia do navegador, como a lista de plug-ins, a lista de fontes ou a string *user-agent*.

### 2.4.5. WebGL

WebGL (*Web Graphics Library*) é uma API multiplataforma que traz a linguagem OpenGL ES 2.0 para a Web, de forma a suportar desenhos 3D dentro do HTML [Group 2014]. Desenvolvida pelo Khronos Group[17], ela fornece uma API JavaScript para renderização de gráficos em 3D em um elemento Canvas da HTML5. Em outras palavras, oferece suporte para renderização de gráficos 2D e 3D. Atualmente, WebGL é suportado por todos os navegadores, mas somente está habilitado no Chrome, Firefox e Opera. O Safari vem com o WebGL desativado por questões de segurança, como explicado a seguir.

Em estudo de 2011, Forshaw [Forshaw 2011] descobriu que existe uma série de problemas de segurança graves com a especificação e implementação do WebGL. Estas questões podem permitir que um invasor forneça um código malicioso, através de um navegador, que permita ataques contra os drivers de GPU e gráficos. Estes ataques po-

[17] http://www.khronos.org

dem inutilizar o processamento de toda a máquina. Além disso, os perigos trazidos pela WebGL incluem ataques de negação de serviço [Shankland 2011].

No quesito *device fingerprinting*, Forshaw [Forshaw 2011] afirma que dentro do contexto de WebGL é possível se obter parâmetros de identidade do navegador, tais como: nome do fabricante, nome do navegador, motor de renderização e outras informações. Assim, os navegadores que permitem WebGL por padrão podem colocar seus usuários em risco. Uma contramedida ofertada pelos navegadores para mitigar os padrões de mau comportamento e incidentes graves com WebGL consiste em apenas permitir acesso um conjunto de placas gráficas listadas em uma *whitelist*.

Num estudo posterior [Contextis 2011], foi descoberta uma vulnerabilidade que permite que qualquer imagem de vídeo que fosse exibida no sistema pudesse ser roubada por um atacante, lendo dados não inicializados da memória gráfica. Essa vulnerabilidade não se limita ao conteúdo WebGL, mas inclui outras páginas Web, o computador do usuário e outras aplicações. A Figura 2.8 ilustra essa vulnerabilidade e suas consequências.

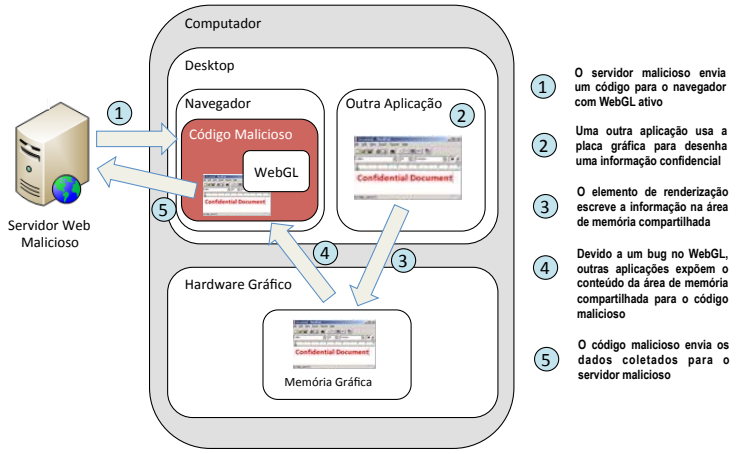



**Figura 2.8. Exemplo de um ataque através do WebGL. Fonte: [Contextis 2011]**

### 2.4.6. CSS

O *Cascading Style Sheets* (CSS) é uma linguagem de estilo utilizada para descrever a aparência e a formatação de um documento escrito em uma linguagem de marcação [Bos et al. 2011]. CSS foi projetada para permitir a separação do conteúdo do documento de apresentação de documentos, através de propriedades de estilo que incluem elementos de layout, cores, fontes, entre outros. Além disso, é independente da HTML e pode ser usado com qualquer linguagem de marcação baseada em XML. A separação do código HTML do CSS faz com que seja mais fácil de manter e melhorar a acessibilidade, proporcionando maior flexibilidade e controle na especificação de características de apresentação dinâmica. Em termos gerais, CSS é a parte do código que de fato da forma ao conteúdo.

Como já mencionado, o principal benefício da CSS é permitir um estilo consistente para ser aplicado em uma série de páginas Web. Já as desvantagens incluem: (i) Problemas de velocidade, já que o download de um arquivo HTML e um arquivo CSS levará mais tempo; (ii) Sintaxe diferente do HTML, e considerada bastante desajeitada e

não amigável para o usuário; e (iii) Templates complicados, mesmo quando associado a sistemas de gerenciamento de conteúdo (CMSs) como Joomla[18] e Drupal[19].

Embora as vantagens do CSS superem suas desvantagens, os projetistas de páginas Web ainda precisam estar cientes dos perigos que podem surgir a partir de seu uso, especialmente no que diz respeito a técnica de *device fingerprinting*. Os autores em [Mulazzani et al. 2013] afirmam que as diferenças nos mecanismos de layout permitem identificar um determinado navegador pelas propriedades CSS que ele suporta. Um caso bem conhecido ocorre quando propriedades CSS ainda estão em status "Candidatas a Recomendação". Assim, os navegadores inserem um prefixo específico antes do fornecedor do motor de layout para indicar que a propriedade é suportada apenas para este navegador. Uma vez que uma propriedade muda para o status "Recomendada", os prefixos são descartados pelo navegador. Por exemplo, no Firefox 3.6 a propriedade *border-radius* teve um prefixo adicionado, resultando em *moz-border-radius*, enquanto que o Chrome 4.0 e o Safari 4.0 usaram *webkit-border-radius*. Quando o Firefox passou a versão 4.0, o Safari para a 5.0 e Chrome também, esse recurso foi uniformemente implementado como *border-radius*.

No site *http://www.caniuse.com* é apresentada uma visão geral de como as propriedades CSS são suportadas nos diferentes navegadores e em seus motores de layout. Assim, uma vez que os navegadores podem diferir em relação a como suportam CSS, seu uso é muito adequado para *fingerprinting* do navegador. Basicamente, existem duas maneiras para testar as propriedades CSS no objeto de estilo de uma página Web. A primeira maneira é simplesmente testar se o navegador suporta uma propriedade específica, usando-a como palavra-chave em um objeto arbitrário. A segunda é olhar para o valor de uma propriedade, uma vez que ela foi definida. Pode-se definir uma propriedade CSS arbitrária em um elemento e consultar o objeto de estilo JavaScript depois. Os valores de retorno indicam se a propriedade de estilo CSS é suportada pelo navegador.

A Listagem 2.7 ilustra um trecho de código JavaScript capaz de obter o histórico da navegação do usuários através do CSS.

**Listagem 2.7. Código JavaScript para obter o histórico do navegador via CSS**

```
1  var agent = navigator.userAgent.toLowerCase();
2  var is_mozilla = (agent.indexOf("mozilla") != -1);
3
4  // popular websites. Lookup if user has visited any.
5  var websites = [
6    "http://ha.ckers.org",
7    "http://mail.google.com/",
8   ...
9  ];
10
11 /* prevent multiple XSS loads */
12 if (! document.getElementById('xss_flag')) {
13
14   var d = document.createElement('div');
15   d.id = 'xss_flag';
16   document.body.appendChild(d);
17
18   var d = document.createElement('table');
19   d.border = 0;
```

[18]http://www.joomla.org
[19]http://www.drupal.org

```
20    d.cellpadding = 5;
21    d.cellspacing = 10;
22    d.width = '90%';
23    d.align = 'center';
24    d.id = 'data';
25    document.body.appendChild(d);
26
27    document.write('');
28    for (var i = 0; i <> '');
29
30    /* launch steal history */
31
32  if (is_mozilla) {
33    stealHistory();
34  }
35
36  }
37
38  function stealHistory() {
39
40    // loop through websites and check which ones have been visited
41    for (var i = 0; i < websites.length; i++) {
42            var link = document.createElement("a");
43            link.id = "id" + i;
44            link.href = websites[i];
45            link.innerHTML = websites[i];
46            document.body.appendChild(link);
47            var color = document.defaultView.getComputedStyle(link,null).
                  getPropertyValue("color");
48            document.body.removeChild(link);
49  // check for visited
50        if (color == "rgb(0, 0, 255)") {
51            document.write('' + websites[i] + '');
52         } // end visited check
53
54    } // end visited website loop
55
56  } // end stealHistory method
```

### 2.4.7. Silverlight

Silverlight é um framework criado pela Microsoft para o desenvolvimento e execução de aplicações ricas para a Internet, com recursos e propostas similares ao Adobe Flash. De acordo com a Microsoft [Microsoft 2014], o Silverlight é um instrumento poderoso de desenvolvimento para a criação de experiências atrativas ao usuário, interativas para Web e aplicações móveis. Seu ambiente de execução está disponível por meio de um plug-in gratuito para navegadores Internet Explorer, Mozilla Firefox e Google Chrome, que executam sob o sistema operacional Windows e Mac OS X, além de plataformas móveis como Windows Mobile e Symbian. Em termos técnicos, o Microsoft Silverlight é uma aplicação *cross-browser*, *cross-platform* do framework .Net.

As primeiras versões do Silverlight focavam no *streaming* de mídia, mas as versões atuais suportam multimídia, gráficos e animação, e dão aos desenvolvedores suporte para idiomas e ferramentas de desenvolvimento. Assim como Flash, permite a criação de jogos 3D acelerados via hardware [Slivinski 2011].

Embora o Silverlight possua várias características importantes e atraentes para os usuários, este plug-in também está vulnerável as técnicas de *device fingerprinting*. Similar as outras tecnologias, é possível obter informações do sistema, tais como: versão do sistema operacional, contagem do processador, fuso horário, fontes instaladas, sistema,

região e idioma do sistema operacional, entre outros.

A Listagem 2.8 ilustra um trecho de código JavaScript para verificar a presença de Silverlight no Navegador. Já a Listagem 2.9 apresenta uma função Silverlight que lista os dispositivos disponíveis no computador. O código usa uma infraestrutura, chamada WIA (*Windows Image Acquisition*), que permite coletar (listar) dispositivos externos como scanners, câmeras de vídeo e câmeras fotográficas conectadas ao computador local. O resultado da função da Listagem 2.9 é apresentado na Figura 2.9.

**Listagem 2.8. Verificando a existência de Silverlight no Navegador**

```
function fingerprint_silverlight() {
   ...
    try {
        try {
            objControl = new ActiveXObject('AgControl.AgControl');
            if (objControl.IsVersionSupported("5.0")) { strSilverlightVersion = "5.x"
                ; }
            else if (objControl.IsVersionSupported("4.0")) {  strSilverlightVersion =
                "4.x"; }
            else if (objControl.IsVersionSupported("3.0")) {  strSilverlightVersion =
                "3.x"; }
            else if (objControl.IsVersionSupported("2.0")) {  strSilverlightVersion =
                "2.x"; }
            else { strSilverlightVersion = "1.x";  }
            objControl = null;
        } catch (e) {
            objplug-in = navigator.plug-ins["Silverlight Plug-In"];
            if (objplug-in) {
                if (objplug-in.description === "1.0.30226.2") { strSilverlightVersion
                    = "2.x"; }
                else { strSilverlightVersion = parseInt(objplug-in.description[0],
                    10);
                }
            } else \textit{{ strSilverlightVersion = "N/A";  }}
        }
        strOut = strSilverlightVersion;
        return strOut;
    } catch (err) {  return strOnError;  }
}
```

**Listagem 2.9. Função Silverlight para listar os dispositivos no computador**

```
private void btnIterateWIA_Click(object sender, RoutedEventArgs e)
{
    StringBuilder sb = new StringBuilder();
    sb.AppendLine("List of available external devices and commands:");
    using (dynamic DeviceManager = ComAutomationFactory.CreateObject("WIA.
        DeviceManager"))
    {
        var deviceInfos = DeviceManager.DeviceInfos;
        for (int i = 1; i <= deviceInfos.Count; i++)
        {
            var IDevice = deviceInfos.Item(i).Connect();
            var DeviceID = IDevice.DeviceID;
            var DeviceName = IDevice.Properties("Name").Value;
            var Commands = IDevice.Commands;
            for (int j = 1; j <= Commands.Count; j++)
            {
                var IDeviceCommand = Commands.Item(j);
                var CommandName = IDeviceCommand.Name;
                var CommandDescription = IDeviceCommand.Description;
                sb.AppendLine("    " + DeviceName + " (" + DeviceID + "), " +
                    CommandName + ": " + CommandDescription);
                // Execute with: IDevice.ExecuteCommand(IDeviceCommand.CommandID);
            }
        }
```

```
23          }
24          MessageBox.Show(sb.ToString());
25      }
```

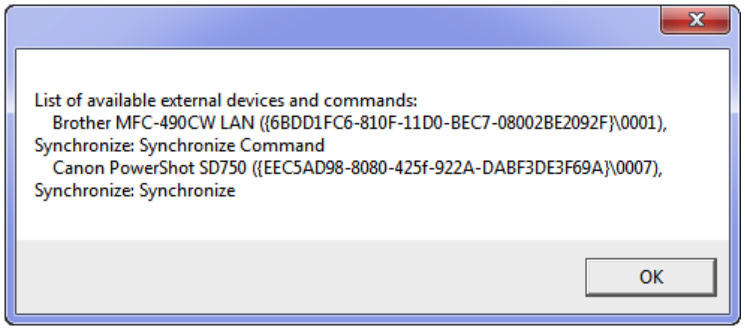



**Figura 2.9. Exemplo de lista de dispositivos externos detectados**

## 2.5. Construindo um Device Fingerprinting

Esta seção enumerará os passos necessários para o desenvolvimento de um *device fingerprinting* bem como elucidará os principais dados obtidos durante o processo de *fingerprinting*. Para tanto, a seção será dividida em duas partes. A primeira emprega o trabalho [Nikiforakis et al. 2013] como roteiro para construção de tais mecanismos. Visando melhorar ainda mais o entendimento, a segunda parte apresenta resultados das etapas de um *device fingerprinting*.

### 2.5.1. Roteiro de Construção

Desde a publicação do trabalho de Eckersley [Eckersley 2010], informações sobre o *user-agent*, o cabeçalho HTTP, a resolução da tela, o timezone, a lista de plug-ins, o sistemas de fontes, entre outras propriedades, tem servido como roteiro para criação de *device fingerprinting*.

Uma vez que todos essas informações (objetos *Navigator* e *Screen* do HTML) já foram discutidas e exemplificadas neste Capítulo, esta seção vai, primeiramente, discutir dois aspectos necessários para melhor se obter (escolher) as informações desejadas. São eles:

- **A escolha de plug-ins populares**: O trabalho de Eckersley [Eckersley 2010] provou que os usuários, tipicamente, possuem vários plug-ins instalados em seus navegadores. Embora a grande maioria deles seja proprietária (o Flash é o melhor exemplo disso), eles possuem ampla adoção, mesmo que as funções que executam agora possam ser realizadas por novos padrões e especificações como a HTML5. No contexto de *device fingerprinting*, o fato é que quanto mais formas de identificar um dispositivo melhor. Por exemplo, um processo de identificação de um usuário Linux, executando o navegador Firefox em uma máquina de 64 bits, responde como "Linux x86-64" quando perguntado (consultado) sobre a plataforma de execução (objeto *Navigator*). Por outro lado, a mesma pergunta feita utilizando um código Flash proporciona uma resposta genérica da versão do kernel (Linux 3.2.0-26-genérico). O mesmo comportamento é visto na chamada que informa a resolução da

tela do usuário. Em implementações Linux do plug-in do Flash (Adobe e Google), quando um usuário utiliza uma configuração com dois monitores (*dual-monitor*), o Flash relata a largura da tela como sendo a soma das duas telas individuais.

- ***Fingerprinting* com foco em vários fabricantes**: Para obter melhores resultados, grande parte dos *fingerprintings* não tentam trabalhar da mesma forma em todos os navegadores. Por exemplo, quando o reconhecimento é focado no Internet Explorer, os códigos buscam extensivamente propriedades específicas como *navigator.securityPolicy* e *navigator.systemLanguage*.

Discutidos estes dois aspectos, a construção de um simples *device fingerprinting* se resume a quatro passos:

1. **Enumeração dos objetos *Navigator* e *Screen***: O objetivo é obter uma lista de todas as propriedades relacionadas a estes objetos disponíveis no navegador. Nesta etapa, os dois aspectos discutidos anteriormente precisam ser levados em consideração. Análises realizadas no trabalho de [Nikiforakis et al. 2013] mostram que a ordem das propriedade durante a enumeração dos objetos *Navigator* e *Screen* é sempre diferente entre as famílias de navegadores, versões de cada navegador e, em alguns casos, entre as implementações da mesma versão em diferentes sistemas operacionais. Além disso, apesar de hoje em dia o padrão HTML ser governado pelo W3C e o JavaScript pelo ECMA, os fabricantes de navegadores ainda adicionam novos recursos que não pertencem a nenhuma norma específica. Em muitos casos, os nomes desses novos recursos começam com um prefixo específico do fornecedor, como *screen.mozBrightness* para o Mozilla Firefox e *navigator.msDoNotTrack* para o Microsoft Internet Explorer. A Figura 2.10 ilustra os possíveis dados que podem ser obtidos.

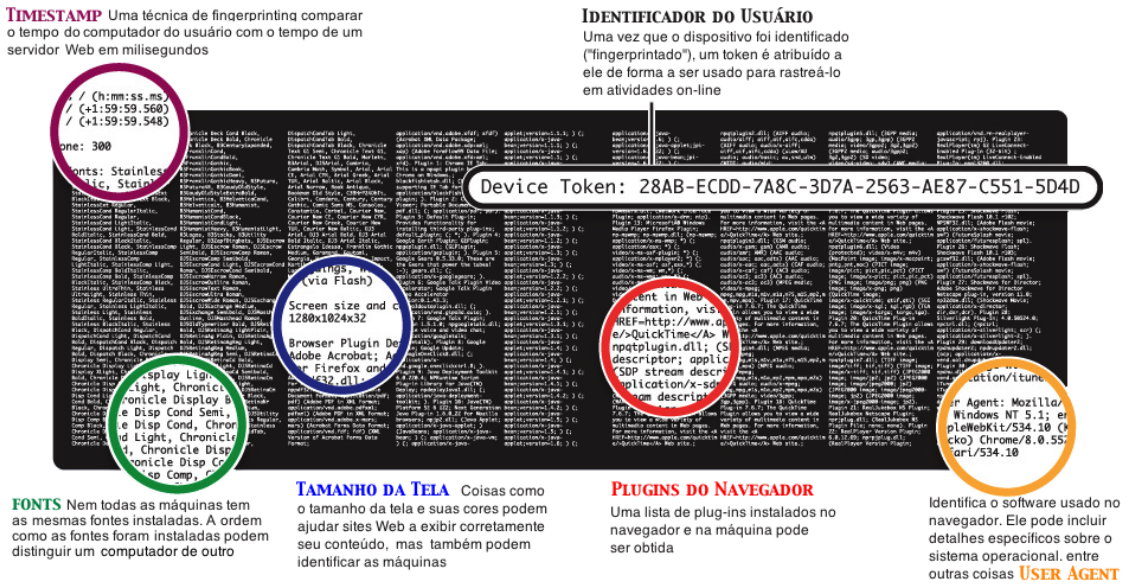



**Figura 2.10. Possíveis dados obtidos com um *device fingerprinting***

2. **Enumeração do objeto *Navigator* novamente**: A ideia é verificar se a ordem da enumeração feita no passo anterior não foi modificada, ou seja, se nenhum dos valores recebidos está diferente. Se houverem mudanças, é muito provável que contramedidas *fingerprinting* estejam em uso.

3. **Exclusão e alteração de propriedades dos objetos**: O objetivo é tentar excluir e/ou alterar uma propriedade dos objetos *Navigator* e *Screen*. Isso porque somente o Google Chrome permite que um script exclua uma propriedade a partir do objeto *Navigator*. Já no quesito alteração, somente o Google Chrome e Opera permitem a modificação do valor de uma propriedade do objeto *Navigator*. O mesmo vale para atributos do objeto *Screen*.

4. **Geração do identificador único**: Usando os dados recolhidos pelo script *fingerprinting*, é possível gerar um identificador único para o dispositivo.

### 2.5.2. Dados coletados durante um device fingerprinting

Além dos diferentes trabalhos já expostos neste Capítulo, esta seção apresenta as principais informações que podem ser facilmente adquiridas, utilizando os conceitos já apresentados. Para isso, o site noc.to foi empregado como modelo.

A primeiras informações "expostas pelo navegador" dizem respeito ao **Sistema**, tais como: (i) Endereço IP e porta; (ii) Sistema operacional; (iii) *user-agent* do navegador; (iv) *user-agent* do sistema operacional; (v) Número de processadores; (vi) Resolução da tela; (vii) Horário do servidor e do cliente.

Em seguida, o *device fingerprinting* do noc.to apresenta informações sobre **Cookies** e o cabeçalho HTTP. Por fim, são relatadas informações JavaScript sobre os objetos *Navigator* e *Screen*, **JavaScript Data** e **JS Display Data**, respectivamente. Vale ressaltar que JavaScript possui muitos campos para ambos objetos e que nem todos os campos existem em todos os navegadores. A Figura 2.11 ilustra todos os tipos dados obtidos.

As próximas informações obtidas são sobre a **Geolocalização do Navegador** e a **Geolocalização do IP** do cliente (Figura 2.12). Para a geolocalização do navegador é utilizada a API JavaScript de gelocalização da HTML5, onde dados como altitude, velocidade e outros valores serão informados se estiverem disponíveis. Já para a geolocalização do IP, utiliza-se informações de geolocalização da MaxMind[20].

Por fim, o noc.to ainda exibe outras informações, incluindo os dados dos plug-ins Silverlight e Flash bem como a lista de plug-ins disponíveis no navegador. Por restrição de espaços, tais dados não serão ilustrados.

## 2.6. Soluções Existentes

Esta Seção apresenta algumas sites, ferramentas e frameworks que implementam técnicas de *device fingerprinting*. Primeiramente, serão discutidas três soluções acadêmicas, sendo a primeira [Eckersley 2010] considerada a pioneira na área. Depois serão apresentadas soluções comerciais, onde uma discussão, usando o trabalho de [Nikiforakis et al. 2013], revelará mais sobre como e o que essas empresas usam em seus scripts *fingerprinting*.

[20]https://www.maxmind.com/

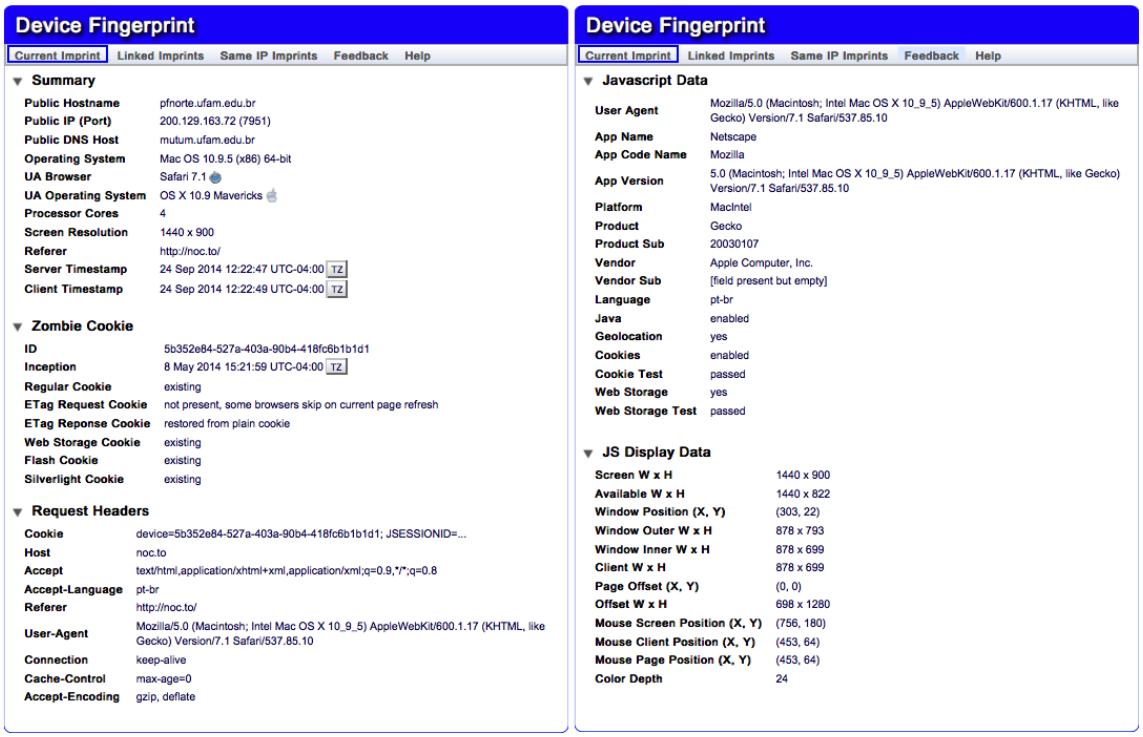



**Figura 2.11. Dados referentes ao Sistema, aos Cookies, ao cabeçalho HTTP e JavaScript coletados pelo *device fingerprinting* do noc.to**

### 2.6.1. Pesquisas Acadêmicas

#### How Unique Is Your Web Browser?

Peter Eckersley [Eckersley 2010], em seu artigo "How Unique Is Your Web Browser", relata sua experiência ao verificar o quanto a configuração de um navegador é única e possível mecanismo de rastreamento de usuários na Internet. Sua pesquisa investigou o grau em que os navegadores modernos estão sujeitos a *device fingerprinting* através da análise das versões e informações de configuração que os navegadores possuem e que podem ser coletadas, com ou sem autorização prévia dos usuários, como, por exemplo, as dimensões de tela, fusos horários e lista de fontes instaladas, entre outras. No final, todas essas informações podem ser combinadas para gerar um identificador único para o dispositivo.

Para realizar a pesquisa, Eckersley desenvolveu um site de teste, denominado **Panopticlick** (https://panopticlick.eff.org) que utiliza um algoritmo, também desenvolvido por ele, cuja finalidade é aplicar mecanismos de *fingerprinting* no navegador do usuário e, consequentemente, gerar um identificador único. Os dados dos navegadores foram recolhidos a partir de visitas feitas ao site de teste. Inicialmente, a pesquisa contou com 470.161 acessos de navegadores operados pelos participantes informados do tema ao visitaram o site.

Como resultado, Eckersley observou que mesmo com os usuários conscientes, foi possível gerar 83,6% de identificações únicas e apenas 5,3% de navegadores/usuários conseguiram se manter no anonimato. Ao verificar se os navegadores possuíam Adobe

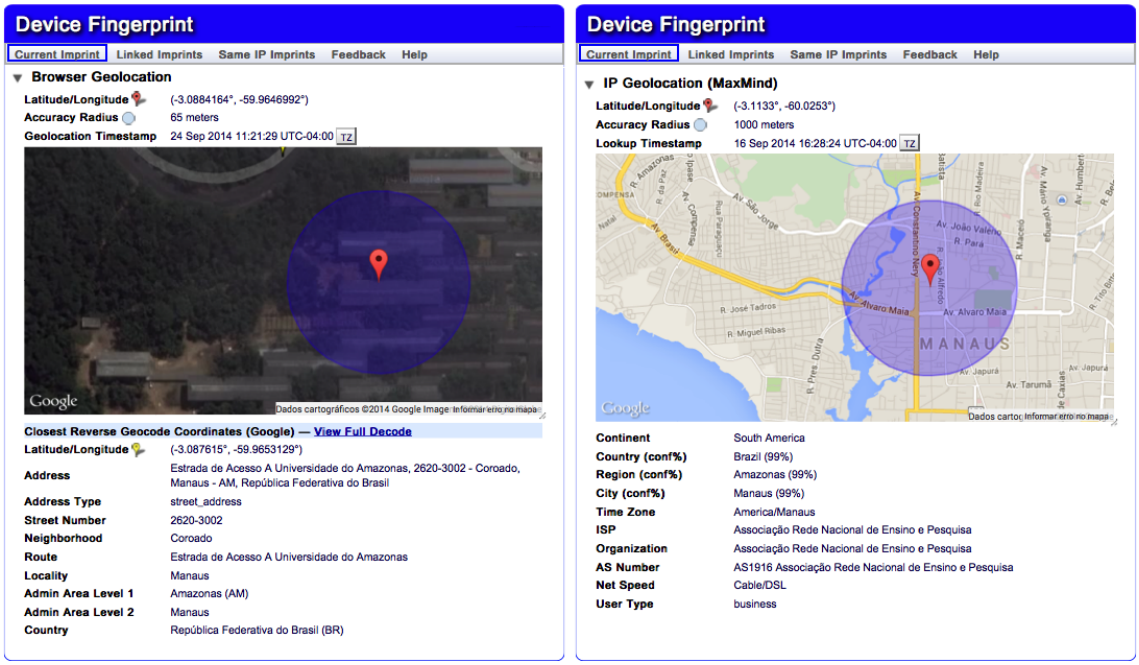



**Figura 2.12. Dados referentes a Geolocalização do navegador coletados pelo *device fingerprinting* do noc.to**

Flash ou tinham a Máquina Virtual Java habilitada, 94,2% apresentaram instantaneamente identificadores únicos. Apenas 1,0% dos navegadores com Flash ou Java manteve o anonimato. Mas, esses percentuais não são definitivos, pois quando houverem mudanças no navegador como, por exemplo, atualizações de versão, atualização de um plug-in, desabilitação dos Cookies, instalação de uma nova fonte, um aplicativo externo que incluiu fontes ou alterou a resolução da tela, a identificação única feita não será mais válida.

Em particular, Eckersley usou Cookies para reconhecer os navegadores que estavam retornando e assim verificar se suas identificações sofreram mudanças. O preocupante é o fato de um algoritmo, considerado simples, ser capaz de identificar todas as mudanças ocorridas, chegando a um percentual de 99,1% de acerto, com uma taxa de falso positivo de apenas 0,87%.

O autor também destaca que se o navegador estiver com o JavaScript bloqueado ou se estiver utilizando algum plug-in de bloqueio, o Panopticlick não consegue capturar as informações do navegador. Assim, outros autores, baseados na descoberta de Eckersley, desenvolveram pesquisas e apresentaram outras implementações de *device fingerprinting*.

### FPDetective: Dusting the Web for Fingerprinters

A pesquisa de Acar et al. [Acar et al. 2013], descrita no artigo "FPDetective: Dusting the Web for Fingerprinters", relata o desenvolvimento de um framework, denominado FPDetective, que centra-se na detecção de códigos de *fingerprinting*. Ao realizar uma análise em grande escala, com milhões de sites populares da Internet, os autores descobriam que a adoção de técnicas *device fingerprinting* é muito maior do que os estudos anteriores haviam estimado, bem como a lista de códigos de *fingerprinting* conhecidos.

O framework FPDetective está disponível gratuitamente e pode ser obtido a par-

tir do endereço http://homes.esat.kuleuven.be/gacar/fpdetective. A Figura 2.13 ilustra a arquitetura do FPDetctive. O framework foi desenvolvido utilizando Python, C++, JavaScript e MySQL, com o objetivo de ser flexível e poder ser usado para realizar estudos de privacidade Web em larga escala.

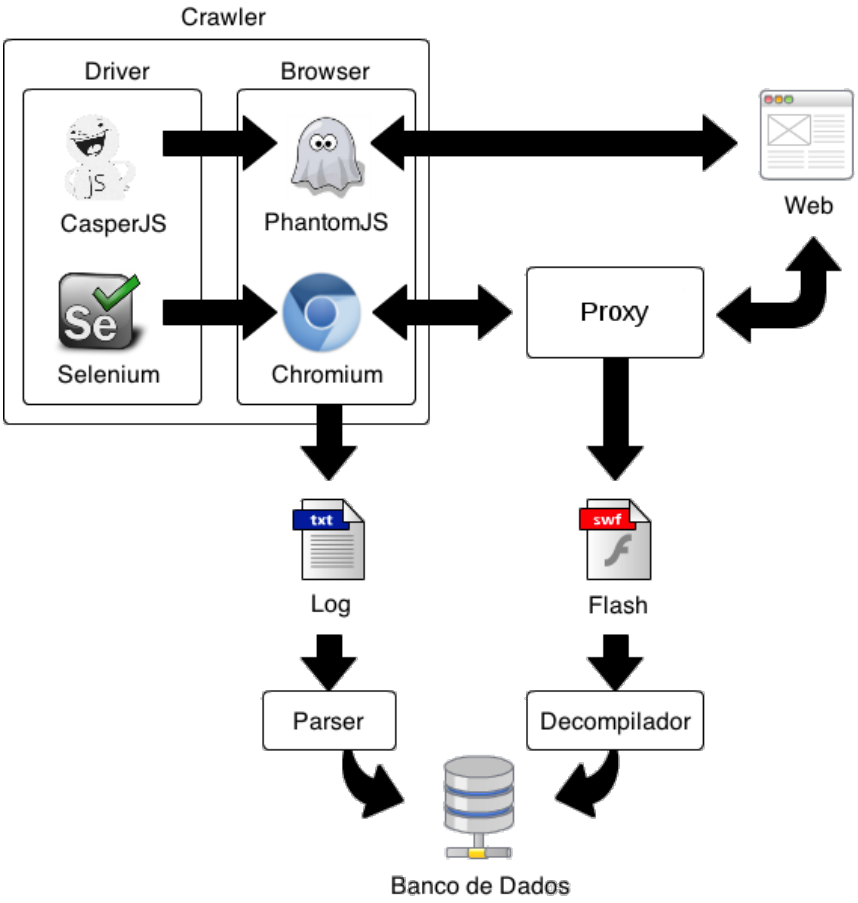



**Figura 2.13. Arquitetura do Framework FPDetective**

Uma descrição dos componentes do FPDetective é apresentada a seguir:

1. **Crawler**: O Crawler possui dois navegadores, PhantomJS[21] e Chromium[22]. Ambos foram instrumentados, ou seja, tiveram partes do código-fonte do WebKit, o motor de renderização, modificados para se adequar ao trabalho. O PhantomJS foi escolhido para coletar *fingerprinting* baseado em JavaScript, uma vez que faz um uso mínimo de recursos. O Chromium foi usado para investigar *fingerprinting* baseado em Flash, já que o PhantomJS não tem suporte a este plugin. CasperJS[23] e Selenium[24] foram usados para conduzir os navegadores até os sites e navegar através das páginas.

[21] http://phantomjs.org
[22] http://chromium.org
[23] http://casperjs.org/
[24] http://docs.seleniumhq.org/

2. **Parser**: O Parser é usado para extrair dados relevantes, a partir dos dados gerados pelo Crawler, e armazená-los em um banco de dados. Os scripts *fingerprinting* conhecidos e/ou encontrados nas solicitações HTTP também foram processados pelo Parser.

3. **Proxy**: A fim de obter os arquivos Flash para análise estática, todo o tráfego HTTP foi redirecionado para mitmproxy[25], um proxy capaz de interceptar SSL, e a biblioteca libmproxy para analisar e extrair arquivos Flash que fazem *sniffing*.

4. **Decompilador**: A biblioteca JPEXS Free Flash Decompiler[26] foi usada para descompilar arquivos Flash e obter o código-fonte do ActionScript. O Decompilador procura por chamada de funções (*enumerateFonts* e *getFontList*, por exemplo) de *fingerprinting* para obter um vetor binário de ocorrências.

5. **Banco de Dados**: Uma vez que o Crawler pode ser executado em várias máquinas, foi utilizado um banco de dados central para armazenar, combinar e analisar os diferentes resultados com o mínimo de esforço. Os dados armazenados incluem o conjunto de chamadas de funções JavaScript, a lista de pedidos e respostas HTTP e a lista de fontes carregadas ou solicitados. Para os experimentos Flash, também é armazenado um vetor binário que representa a ocorrência de chamadas da API do ActionScript que podem estar relacionados ao *fingerprinting*.

Como resultado da pesquisa, após a verificação realizada pelo FPDetective, foram encontrados 404 sites que estão no Top Rank do site Alexa.com que se utilizam de JavaScript para *fingerprinting*. A Tabela 2.3 apresenta o detalhamento deste resultado.

**Tabela 2.3. Resultado do FPDetective para sondagem com JavaScript. Fonte: [Acar et al. 2013]**

| | | | | Número de sites usando FP baseados em JS | | |
|---|---|---|---|---|---|---|
| | | | | 1M | 100K | |
| **Provedor do Fingerprinting** | **Nome do Script** | **No. de Fontes** | **Top Rank** | **Homepage** | **Homepage** | **Página Interna** |
| BlueCava | BCAC5.js | 231/167/62 | 1.390 | 250 | 24 | 24 |
| Perferencement | tagv22.pkmin.js | 153 | 49.979 | 51 | 6 | 6 |
| CoinBase | application- 773a[...snipped...].js | 206 | 497 | 28 | 4 | 4 |
| MaxMind | device.js | 94 | 498 | 24 | 5 | 5 |
| Inside graph | ig.js | 355 | 98.786 | 18 | 1 | 1 |
| SiteBlackBox | URL não definida | 389 | 1.687 | 14 | 10 | 10 |
| Analytics-engine | fp.js | 98 | 36.161 | 6 | - | - |
| Myfreecams | o- mfccore.js | 71 | 422 | 3 | 1 | 1 |
| Mindshare Tech. | pomegranate.js | 487 | 109.798 | 3 | - | - |
| Cdn.net | cc.js | 297 | 501.583 | 3 | - | - |
| AFK Media | fingerprint.js | 503 | 199.319 | 2 | - | - |
| Anonymizer | fontdetect.js | 80 | 118.504 | 1 | - | - |
| | | | | **404** | **51** | **51** |

Já em relação ao Flash, foram encontrados elementos de *fingerprinting* em 145 homepages no Top Rank da Alexa.com, o que indica que *fingerprinting* baseado em Flash é mais prevalente. Os autores atribuem esse maior número de detecções em Flash devido as capacidades ampliadas para enumeração de fonte, detecção de proxy e compatibilidade com navegadores. A Tabela 2.4 apresenta apenas *fingerprinting* Flash que usam enumeração de fonte (95 deles).

[25]http://mitmproxy.org

[26]http://code.google.com/p/chromium/issues/detail?id=55084

**Tabela 2.4. Objetos Flash com enumeração de fontes. Fonte: [Acar et al. 2013]**

| Provedor do Fingerprinting | No. de Sites | Top Rank | Cookies Flash | Detecção de Proxy | Identificação de HW e SO | Interação com JS |
|---|---|---|---|---|---|---|
| BB Elements | 69 | 903 | ✓ | | | ✓ |
| Piano Media | 12 | 3.532 | ✓ | | ✓ | ✓ |
| Bluecava | 6 | 1.390 | ✓ | | | ✓ |
| ThreatMetrix | 6 | 2.793 | | ✓ | ✓ | |
| Alipay | 1 | 83 | ✓ | | ✓ | ✓ |
| meb.gov.tr (Turkish Ministry of Education) | 1 | 2.206 | ✓ | | ✓ | ✓ |

#### 2.6.1.1. SHPF

O SHPF (*Session Hijacking Prevention Framework*) é um framework desenvolvido para combater o sequestro de sessões (*Session Hijacking*) em navegadores Web através de *fingerprinting* [Unger et al. 2013]. Ele oferece um conjunto de múltiplos mecanismos de detecção que podem ser usados independentemente um dos outros. SHPF protege especialmente contra sequestro de sessões bem como contra XSS (*Cross-site scripting*). A ideia do framework é: se o navegador do usuário muda de repente, por exemplo, do Firefox, no Windows 7 de 64 bits, para um navegador Webkit baseado em Android em uma faixa de IP totalmente diferente, é de se supor que algum tipo de atividade maliciosa está acontecendo.

Assim, o framework SHPF consegue melhorar o gerenciamento de sessões HTTP(S) através de um *device fingerprinting*. Para tanto, dois novos métodos baseados em CSS e HTML5 foram desenvolvidos.

Em relação a implementação, o SHPF foi escrito em PHP5 e consiste de múltiplas classes que podem ser carregadas independentemente. A Figura 2.14 apresenta a arquitetura geral e as funcionalidades básicas do SHPF.

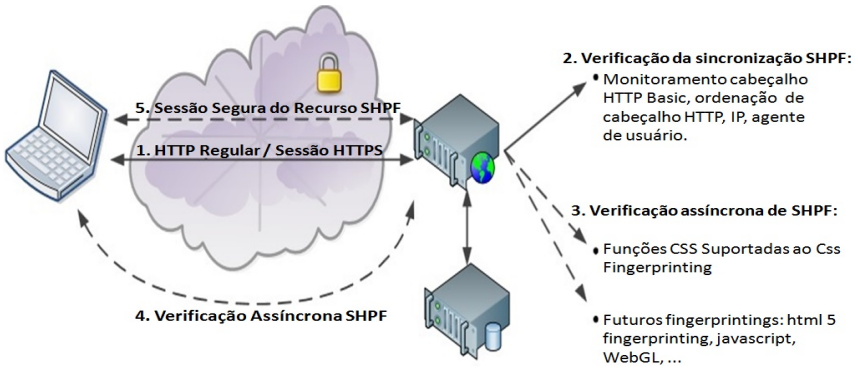



**Figura 2.14. Arquitetura e funcionamento do SHPF. Fonte: [Unger et al. 2013]**

As principais partes da estrutura do framework são chamadas de *features*. Uma *feature* é uma combinação de diferentes verificações para detecção e mitigação do sequestro de sessão. O protótipo implementa as seguintes *features*: monitoramento do cabeçalho HTTP, *fingerprinting* CSS e SecureSession (que implementa e amplia o protocolo SessionLock [Unger et al. 2013]. As *features* também são os meios de extender o framework e fornecer classes base para o desenvolvimento rápido de novas *features*. Uma *feature* consiste de um ou mais *checkers*, que são utilizados para executar certos testes. Existem

dois tipos diferentes (ou classes) de *checkers*: os síncronos, que podem ser utilizados se os testes incluídos forem executados apenas a partir de dados existentes e são passivos por natureza, tais como solicitações HTTP; e os assíncronos, que são usados se os testes têm de solicitar ativamente alguns dado adicional do cliente e o framework tem que processar a resposta.

No que diz respeito ao *fingerprinting*, a *feature* que utiliza um mix de propriedades CSS3 foi capaz de identificar 23 elementos adequados, conforme descrito na Tabela 2.5.

**Tabela 2.5. Propriedades CSS e valores identificados pelo SHPF**

| **CSS Status - Recomendado** | |
|---|---|
| ***Característica*** | ***Valor*** |
| display | nline-block |
| min-width | 35px |
| position | fixed |
| display | table-row |
| opacity | 0.5 |
| background | hsla(56, 100%, 50%, 0.3) |
| **CSS Status - Candidato a Recomendação** | |
| ***Característica*** | ***Valor*** |
| box-sizing | border-box |
| border-radius | 9px |
| box-shadow | inset 4px 4px 16px 10px #000 |
| column-count | 4 |
| **CSS Status - Draft** | |
| ***Característica*** | ***Valor*** |
| transform | rotate(30deg) |
| font-size | 2rem |
| text-shadow | 4px 4px 14px #969696 |
| background | linear-gradient (left, red, blue 30%, green) |
| transition | background-color 2s linear 0.5s |
| animation | name 4s linear 1.5s infinite alternate none |
| resize | both |
| box-orient | horizontal |
| transform-style | preserve-3d |
| font-feature-setting | dlig=1,ss01=1 |
| width | calc(25% -1em) |
| hyphens | auto |
| object-fit | contain |

### 2.6.2. Soluções Comerciais

Após o trabalho de Eckersley mostrar a possibilidade de realizar um *fingerprinting* em navegadores Web, a fim de rastrear usuários sem a necessidade de instalar qualquer código ou solução no lado do cliente, ficou claro que esse tipo de atividade já existia e que empresas haviam se especializado nisso. Embora existam várias empresas atuando na

identificação de usuários, todas para fins de autenticação e anti-fraude, três delas (BlueCava, Iovation e ThreatMetrix, todas já mencionadas anteriormente neste Capítulo) são consideradas as grandes, devido sua popularidade e gama de clientes.

A Bluecava (http://www.bluecava.com) é uma empresa que fornece tecnologia de reconhecimento (*fingerprinting*), permitindo às empresas de comércio eletrônico evitar fraudes e melhorar a eficácia de sua publicidade on-line. A tecnologia criada pela empresa identifica e direciona usuários da Internet, sem o uso de Cookies. Além disso, funciona em dispositivos como PCs, smartphones, laptops, consoles de jogos e set-top boxes.

Os serviços e soluções que a BlueCava oferece visam utilizar, principalmente, dados do dispositivo, como o endereço IP, versão do navegador e tipo de dispositivo. Dados de consumo são anonimizados e categorizados em segmentos públicos amplos como "homens 18-34" ou "interesse por esporte", por exemplo. Através de várias implementações, a BlueCava identifica, analisa e sincroniza dezenas de dados de entrada do dispositivo. O resultado é apresentado como um gráfico de associação, que permite aos comerciantes implementarem uma *cross-screen* de segmentação, mensuração e otimização ao seu alcance de audiência e frequência.

A Iovation (http://iovation.com) é uma empresa anti-fraude que têm por objetivo evitar que seus clientes façam negócios com fraudadores. Para isso, é capaz de estabelecer uma identificação única para cada dispositivo e rastrear as atividades, de modo que possam identificar comportamentos fraudulentos. Atualmente, a empresa possui 2 bilhões de informações sobre dispositivos, cobrindo 84% de todas as transações consideradas fraudulentas em sites dos seus clientes. O software responsável por isso é o **Reputation Manager 360**.

Criado em 2004 e em continua evolução desde então, o software é capaz de relatar as empresas se o dispositivo acessando seu site ou aplicativo móvel tem um histórico conhecido de fraude em cartão de crédito, roubo de identidade, entre outros comportamentos abusivos. Também é capaz de informar se o dispositivo em questão está associado com quaisquer outros dispositivos que contenham históricos de fraude.

A última empresa, a ThreatMetrix (http://www.threatmetrix.com/), possui uma plataforma que ajuda a proteger a integridade das transações on-line das organizações (incluindo serviços financeiros e bancários on-line, e-commerce, seguros, redes sociais, governo e grandes empresas). As avançadas tecnologias de identificação de dispositivo, elaboradas pelas empresa, ajudam as organizações a determinar se os visitantes on-line são clientes legítimos ou potenciais criminosos virtuais.

A ThreatMetrix apresenta uma série de aplicativos de defesas, dentre os quais pode-se destacar:

- **TrustDefender**: um plataforma de proteção contra crimes cibernéticos que atua na autenticação baseada em contexto para protege usuários contra fraude de pagamento, registro de conta fraudulenta e ataques de malware.
- **ThreatMetrix SmartID**: uma ferramenta de identificação, que não usa Cookies, que permite criar avançados perfis de identificação exclusivamente baseados nos atributos do dispositivo.

#### 2.6.2.1. Comparação entre as soluções

O trabalho de Nikiforakis et al. [Nikiforakis et al. 2013], no artigo "Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting" relata uma comparação entre as empresas BlueCava, Iovation e ThreatMetrix, apresentadas anteriormente, com o objetivo de investigar de que forma essas empresas estão se utilizando das técnicas de *device fingerprinting*.

Para realizar a análise dos scripts de *fingerprinting* dessas três soluções comerciais, sites foram avaliados para descobrir se utilizam os serviços oferecidos por essas empresas. O software Ghostery, detalhado na próxima Seção, foi utilizado para realizar o rastreamento e obter a lista de domínios que usam os scripts de *fingerprinting*. Assim, para quantificar esses sites populares, 10.000 sites do Alexa.com foram analisadas, com profundidade de até 20 páginas, a procura de inclusões de scripts e *iframe* originários dos domínios das três empresas. No final, descobriu-se que 40 site (0,4% dos 10.000 sites) utilizavam *device fingerprinting* dos três provedores comerciais.

O primeiro resultado obtido pela pesquisa foi uma taxonomia das possíveis funções que podem ser adquiridas através de uma biblioteca de *fingerprinting*. Esta taxonomia abrange todos os recursos descritos por Peter Eckersley [Eckersley 2010] (Panopticlick), bem como as características utilizadas pelas três empresas, conforme mostra a Tabela 2.6.

As categorias propostas nessa taxonomia resultaram da análise em camadas, onde a "camada de aplicação" é o navegador e qualquer informação coletada do navegador. No topo dessa taxonomia estão os scripts que buscam identificar todas as personalizações do navegador. Nos níveis mais baixos estão os scripts que possuem informações específicas do usuário em todo o navegador, o funcionamento sistema e até mesmo o hardware e a rede do usuário da máquina [Nikiforakis et al. 2013].

**Tabela 2.6. Comparação entre Panopticlick,BlueCava, Iovation e ThreatMetrix. Fonte: [Nikiforakis et al. 2013]**

| **Categoria** | **Panopticlick** | **BlueCava** | **Iovation** | **ThreatMetrix** |
|---|---|---|---|---|
| Customizações do navegador | Enumeração de plugins (JS); Enumeração de Mime-type (JS); ActiveX + 8 CLSIDs[27] (JS) | Enumeração de plugins (JS); ActiveX + 53 CLSIDs (JS); Google Gears Detection (JS) | | Enumeração de plugins (JS); Enumeração de Mime-type (JS); ActiveX + 6 CLSIDs (JS); Fabricante Flash (FLASH) |
| Configurações do navegador no nível do usuário | Cookies ativos (HTTP); Timezone (JS); Flash ativo (JS) | Linguagem do Sistema/Navegador/Usuário (JS); Timezone (JS), Flash ativo (JS); Do-Not-Track (JS); Política de Segurança MSIE (JS) | Linguagem do navegador (HTTP, JS); Timezone (JS); Flash ativo (JS); Data e hora (JS); Detecção de Proxy (FLASH) | Linguagem do navegador (FLASH); Timezone (JS,FLASH); Flash ativo (JS); Detecção de Proxy (FLASH) |
| Família e versão do navegador | User-agent (HTTP); ACCEPT-Header (HTTP); Teste de Super Cookie (JS) | User-agent (JS); Constantes matemáticas (JS); Implementação AJAX (JS) | User-agent (HTTP, JS) | User-agent (JS) |
| Sistema Operacional e Aplicações | User-agent (HTTP), Detecção de fontes (JAVA, FLASH) | User-agent (JS); Detecção de fontes (JS, FLASH); Registro do Windows (SFP) | User-agent (HTTP, JS); Registro do Windows (SFP); Chave dos Produtos MSIE (SFP) | User-agent (JS); Detecção de fonte (FLASH); Versão do sistema operacional e Kernel (FLASH) |
| Hardware e Rede | Resolução da tela (JS) | Resolução da tela (JS); Enumeração de drivers (SFP); Endereço IP (HTTP); Parâmetros TCP/IP (SFP) | Resolução da tela (JS); Identificação de dispositivos (SFP); Parâmetros TCP/IP (SFP) | Resolução da tela (JS, FLASH) |

O resultado da pesquisa revelou que todas as empresas usam Flash, além de JavaScript. Outro fator interessante para identificação do dispositivo através de *fingerprinting* é o endereço IP, visto que a distinção entre um usuário legitimo e outro fraudulento é crucial e pode ser feita localização do IP. Assim, é do interesse do provedor de *fingerprinting* detectar o endereço IP real do usuário ou, pelo menos, descobrir que o usuário está utilizando um servidor proxy. Ao analisar o código do *ActionScript* incorporado ao códigos Flash, de duas das três empresas, encontraram-se evidências de que o código é capaz contornar os proxies definidos pelo usuário no navegador, ou seja, o aplicativo Flash carregado foi capaz de contactar um host remoto diretamente, desprezando qualquer configuração HTTP proxy.

A pesquisa também provou que valores do Registro do Windows (um ambiente rico para *fingerprinting*) podem ser lidos por plugins. *Fingerprinters* submetidos via DLL foram capazes de ler uma infinidade de valores específicos do sistema, tais como identificador do disco rígido, parâmetros de TCP/IP, o nome do computador, identificador de produto, a data de instalação do Windows e os drivers do sistema instalado (entradas marcadas com SFP na Tabela 2.6). Todos estes valores combinados proporcionam uma forte identificação que JavaScript ou Flash jamais poderiam construir.

Por fim, o site mais popular fazendo uso de *fingerprinting* foi o skype.com, enquanto as duas maiores categorias de sites que usam *fingerprinting* são: Pornografia (15%) e Encontros/namoro (12,5%). Para os sites pornográficos, uma explicação razoável é que as *fingerprinting* são utilizadas para detectar credenciais compartilhadas ou roubadas de membros pagantes, enquanto que para sites de namoro é para assegurar que os atacantes não criem múltiplos perfis para fins de engenharia social.

## 2.7. Contramedidas

Esta seção relatará o que está sendo discutido e utilizado como contramedida para o rastreamento de informações. O uso do Navegador Tor [Tor Project 2014], plug-ins como Firegloves [FireGloves 2014], especificações como Do Not Track (DNT) [Mayer et al. 2011], entre outras, são exemplos do que será apresentado.

### 2.7.1. Navegador Tor

Tor (*The Onion Router*)[28] [Tor Project 2014] é uma rede de túneis virtuais que permite aos usuários e grupos se comunicarem de forma segura, aumentando assim sua segurança e privacidade na Internet. Ele permite que os desenvolvedores de software criem novas ferramentas de comunicação com recursos internos de privacidade.

Originalmente concebido, implementado e implantado como a terceira geração do projeto *onion routing*, do laboratório de pesquisa Naval dos Estados Unidos, com a finalidade de proteger as comunicações do governo, tornou-se um software livre capaz de permitir que os usuários naveguem na Internet anonimamente, sem que suas atividades e localização possam ser descobertas. Tor redireciona o tráfego Internet através de uma rede mundial de voluntários, formada por mais de cinco mil nós livres, para ocultar a localização do usuário e uso de qualquer pessoa a realização de vigilância de rede ou de

[28] O termo "onion routing" refere-se as camadas de aplicação de criptografia, comparadas com as camadas de uma cebola, usadas para tornar anônima a comunicação.

análise de tráfego. Com isso, torna mais difícil rastrear o tráfego de determinado usuário na Internet, ou seja, visitas a sites, emails, mensagens instantâneas e outras formas de comunicação são protegidas, bem como a liberdade e habilidade para conduzir comunicação confidencial. De modo resumido, permite que organizações e indivíduos partilhem informações através de redes públicas sem comprometer sua privacidade.

**Funcionamento**

De modo sucinto, Tor criptografa os dados originais, incluindo o endereço IP de destino, várias vezes, e o envia através de um circuito virtual que compreende sucessivas transmissões selecionadas aleatoriamente. Cada transmissão decifra uma camada de criptografia para revelar apenas o próximo nó do circuito. O último nó decodifica a camada mais interna da criptografia e envia os dados originais para o seu destino, sem revelar, ou mesmo saber, o endereço IP de origem. Uma vez que o encaminhamento da comunicação é parcialmente escondido em cada nó co circuito Tor, este método elimina qualquer ponto único em que a comunicação pode ser descoberta através de vigilância de rede, que depende do conhecimento de qual é a origem e o destino.

A Figura 2.15 ilustra o funcionamento do Tor.

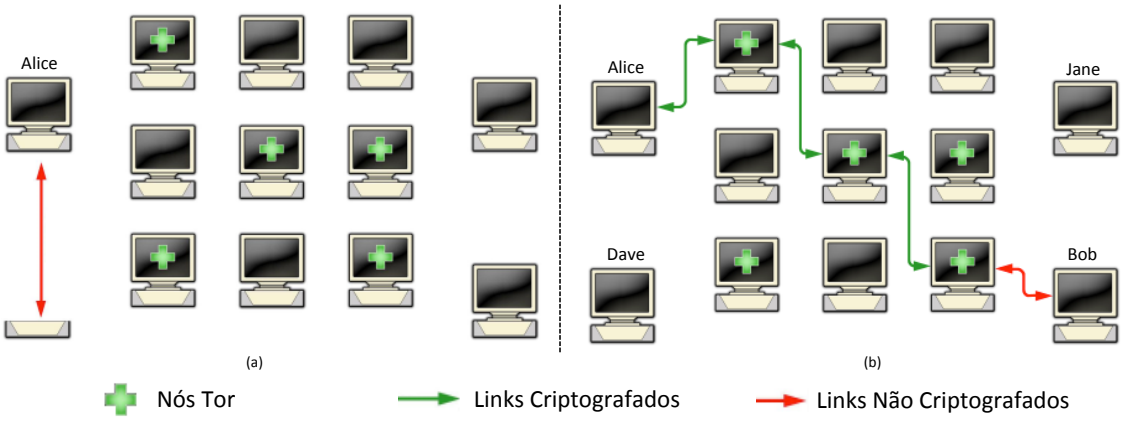



**Figura 2.15. Funcionamento do Tor. Fonte: [Tor Project 2014]**

Quando Alice quer se comunicar com Bob usando a rede Tor, ela primeiro faz uma conexão não criptografada para um servidor de diretório centralizado que conhece todos os endereços dos nós Tor (a). Após receber a lista de endereços, o cliente Tor na máquina de Alice irá se conectar a um nó aleatório (o nó de entrada) da rede Tor, através de uma conexão criptografada. Por sua vez, o nó de entrada fará uma conexão criptografada a um segundo nó aleatório, que por sua vez faz o mesmo para se conectar a um terceiro nó aleatório. Esse terceiro nó, o nó de saída, se conecta a Bob (b).

É importante ressaltar que um mesmo nó não pode ser usado duas vezes, em uma comunicação. Isso porque se a matriz de nós for utilizada por um longo período de tempo, uma conexão Tor seria vulnerável a análise estatística. Por isso, o software no cliente Tor muda o nó de entrada a cada dez minutos.

Diferente de outras soluções similares, especialmente as que utilizam proxies onde todos os usuários entram e saem através do mesmo servidor, Tor passa seu tráfego através

de pelo menos três servidores diferentes antes de enviá-lo para o destino. Ele simplesmente retransmite o seu tráfego, completamente criptografado, através da rede Tor e tem que sair em outro lugar do mundo, completamente intacto. O cliente Tor é necessário para administrar a criptografia e o caminho escolhido pela rede.

Tor apresenta características de segurança relacionadas ao anonimato, a privacidade e a liberdade de expressão. Tor permite o **anonimato**, já que é usado por ativistas políticos, denunciantes, jornalistas, sobreviventes de violência doméstica e pessoas comuns ao redor do mundo que precisam para proteger suas identidades. Também garante a **privacidade**, uma vez que ajuda a esconder o endereço de IP de origem e evita o *fingerprinting* do navegador, tornando difícil que rastreadores on-line e até mesmo governos possam vigiar os usuários. Por fim, garante a **liberdade de expressão**, já que em países onde enormes áreas da Internet são bloqueados, as pessoas usam Tor para acessar a Web sem censura.

**Defesas contra o Fingerprinting**

Um *draft* do projeto Tor [Perry et al. 2013] enumera as defesas que o navegador Tor oferece contra as técnicas de *fingerprinting*. São elas:

1. **Plug-ins**: Além da sua própria funcionalidade (que pode ser desconhecida), plug-ins permitem que as técnicas de *fingerprinting* os descubram (através do objeto *Navigator*) e os explorem. Atualmente, todos os plug-ins são desativados no navegador Tor. No entanto, devido à popularidade do Flash, Tor permite que os usuários possam reativar Flash, mas objetos Flash são bloqueados por trás de uma barreira *click-to-play*, que fica disponível após o usuário habilitar o plug-in. Além disso, no Firefox, Tor define a preferência *plug-in.expose_full_path* para "false", para evitar o vazamento de informações com a instalação de plug-ins.

2. **Extração de imagem via HTML5 Canvas**: Os autores do documento acreditam que o HTML5 Canvas é a maior ameaça de *fingerprinting* a navegadores hoje em dia. Diferenças sutís na placa de vídeo, pacotes de fontes e até mesmo tipo de letra e versões de bibliotecas gráficas permitem produzir um *fingerprinting* simples e de alta entropia de um computador. Para reduzir esta ameaça, Tor recomenda que o navegador seja notificado antes de retornar dados de imagem válidos para as APIs do Canvas. Se o usuário não permitiu o site de acessar dados de imagem da tela, uma imagem em branco é devolvida para a API JavaScript.

3. **WebGL**: WebGL é passível de *fingerprinting* tanto através de informações que são expostas sobre o driver e suas otimizações quanto sobre sua performance. Por este motivo, Tor emprega uma estratégia similar ao dos plug-ins para WebGL. Primeiro, WebGL Canvas têm espaços reservados de *click-to-play* (fornecidos pelo NoScript) onde não executam até serem autorizados pelo usuário. Em segundo lugar, as informações do driver são ofuscadas através das preferências *webgl.disable-extensions* e *webgl.min_capability_mode*, que reduzem a informação fornecida pelas seguintes chamadas da API WebGL: **getParameter()**, **getSupportedExtensions()** e **getExtension()**.

4. **Fontes**: As fontes permitem identificação do navegador via Flash, plug-ins Java,

CSS e JavaScript [Eckersley 2010]. Como a solução para o problema das fontes seria o navegador possuir todas as fonte para todas as línguas, tipo de letra e estilo em uso no mundo (o que é praticamente impossível), Tor desativa os plug-ins, o que impede enumeração de fontes. Além disso, para limitar tanto o número de consultas de fontes CSS bem como o número total de fontes que podem ser usadas em um documento, duas preferências foram criadas e definidas, *browser.display.max_font_attempts* e *browser.display.max_font_count*.

5. **Resolução do desktop, Consultas a mídia CSS e sistema de cores**: Tanto o CSS quanto o JavaScript têm acesso a uma série de informações sobre a resolução de tela, sistema operacional, tamanho da barra de ferramentas, tamanho da barra de título, cores do tema do sistema, entre outros recursos de desktop que não são relevantes para renderização e servem apenas para fornecer informações *fingerprinting*. Tor tem várias estratégias implementadas para combater essas falhas.

6. **User-agent e cabeçalhos HTTP**: Todos os usuários do navegador Tor devem fornecer *user-agent* e cabeçalhos HTTP idênticos para um determinado tipo de solicitação. Para isso, Tor estabelece preferências semelhantes as já definas no Firefox para controlar os cabeçalhos *accept-Language* e *accept-charset*. Além disso, ele remove o acesso de scripts de conteúdo para *components.interfaces*, que podem ser usados para *fingerprinting* de sistema operacional e plataforma.

7. **Desempenho do JavaScript**: O desempenho do JavaScript já vem sendo usado [Mowery et al. 2011, Mulazzani et al. 2013] para identificar o motor de interpretação JavaScript e a CPU. Atualmente, embora existam várias possíveis soluções para o problema, somente uma solução está disponível no TOR, a desativação da propriedade "Navigation Timing" através da preferência *dom.enable_performance*.

8. **Implementação não uniformes da API HTML5**: Uma vez que pelo menos dois recursos HTML5 têm especificações diferentes da implementação nos principais fornecedores de sistemas operacionais (a API da bateria e a API de conexão de rede), Tor criou as preferências *dom.battery.enabled* e *dom.network.enabled*, no Firefox, para desativar essas APIs.

### 2.7.2. Extensões e plug-ins

#### 2.7.2.1. FireGloves

FireGloves é um plug-in para Mozilla Firefox, o qual tem como finalidade principal impedir o rastreamento baseado em *fingerprinting* [FireGloves 2014]. Foi desenvolvido por uma equipe de pesquisadores da Universidade de Tecnologia de Budapeste com intuito de demonstrar que é possível impedir técnicas de *fingerprinting*. A fim de confundir os scripts de *fingerprinting*, o FireGloves, quando consultado, retorna valores aleatórios para determinados atributos, como: a resolução da tela, a plataforma na qual o navegador está em execução, o fornecedor do navegador e a versão. Adicionalmente, ele limita o número de fontes que uma aba do navegador pode carregar através de relatórios com valores falsos das propriedades dos elementos *offsetWidth* e *offsetHeight* da HTML, para evitar a detecção de fontes baseada em JavaScript.

Recentemente, o plug-in passou por melhorias, mas testes realizados descobriram várias insuficiências, como aponta [Acar et al. 2013]. Por exemplo, em vez de confiar em valores para os elementos *offsetWidth* e *offsetHeight*, poderiam ser usados a largura e a altura do objeto *Rectangle*, retornado pelo método *getBoundingClientRect*, que fornece as dimensões do texto.

#### 2.7.2.2. Ghostery

O Ghostery é uma extensão para os principais navegadores, voltada para privacidade na Internet, que permite aos usuários detectarem e bloquearem objetos invisíveis e incorporados em páginas Web que recolhem dados sobre os hábitos de navegação do usuário [Evidon 2014]. O fabricante afirma que o Ghostery mostra todas as empresas que estão monitorando o usuário quando visita um site. Ghostery permite saber mais sobre as empresas e os tipos de dados que elas coletam dos usuários, bem como bloquear tais coletas.

Se por um lado o Ghostery é uma ferramenta valiosa para a privacidade, e seu fabricante é experiente o suficiente para saber que ele não pode ganhar dinheiro apenas bloqueando Cookies de rastreamento, por outro lado, a empresa teve uma idéia inteligente: levantar os dados que os usuários fornecem sobre os bloqueios e vendê-los para as próprias empresas bloqueadas. Para isso, a extensão possui uma função de relatório, chamada "GhostRank", que envia informações anônimas sobre a tecnologia de coleta de dados que estão observando, bem como sua origem. Os dados desses relatórios são analisados e vendidos as empresas, a fim de ajudá-las a auditar e gerenciar seu relacionamento com ferramentas de marketing. Nenhuma das informações que compartilham é sobre seus usuários, nem é armazenada outra forma que poderia ser usada para rastreá-los. Esses mesmos dados também são enviados a Comissão do Comércio Federal (*Federal Trade Commission*[29]), que monitora as operações da indústria de publicidade. Por padrão, o Ghostrank fica desativado, o que significa que seus usuários podem usá-lo sem compartilhar nada com a empresa.

O fato mais importante é que a Evidon está levando a indústria de anúncio para a autoregulação, um esforço que está se tornando mais importante uma vez que o número de redes de publicidade continua a crescer. O Ghostery acrescenta cerca de 70 novas empresas seguidoras a cada três meses [Evidon 2014].

O funcionamento do Ghostery se dá através da verificação do código HTML em cada página Web que o usuário visita, onde é verificado a existência de tags ou "trackers" colocados por uma empresa que trabalha com esse site. Ele pode monitorar todos os diferentes servidores Web que estão sendo chamados a partir de uma determinada página e armazená-los em uma biblioteca de coleta de dados. Se ele encontrar uma correspondência, Ghostery acrescenta a empresa a uma lista.

[29]http://www.ftc.gov

#### 2.7.2.3. Adblockplus

O Adblock Plus (ABP) é um filtro de conteúdo e um bloqueador de anúncios indesejados na Web [Adblock Plus 2014]. Compatível com maioria dos navegadores Firefox (incluindo a versão para dispositivos móveis), Google Chrome, Internet Explorer, Opera e Safari, o ABP possui também uma versão para dispositivos Android. Dentre as principais características do ABP, destacam-se: (i) Suporte para bloquear as imagens de fundo; (ii) Assinaturas de filtros com endereço fixo e atualizações automáticas; (iii) Capacidade de ocultar elementos HTML, permitindo uma maior variedade de imagens a ser bloqueada; (iv) Capacidade de ocultar os anúncios em uma base por site, em vez de todo o mundo. Além do bloqueio de anúncios, o Adblock Plus também faz o rastreamento de anúncios.

O funcionamento do Adblock Plus é apresentado na Figura 2.16. Na ilustração, quando um usuário faz a solicitação de um site, o ABP verifica em sua lista de filtros se no site existe algum tipo de anúncio que deva ser bloqueado. Se sim, ele realiza o bloqueio e apresenta no computador do usuário o site sem o anúncio indesejado.


**Figura 2.16. Funcionamento do Adblock Plus. Fonte: [Adblock Plus 2014]**

Assim como todos os softwares já apresentados até o momento, o Adblock Plus necessita ser ativado pelo usuário para que suas listas de filtros realizem o bloqueio nos sites. É importante relatar que ele possui duas listas de filtros habilitadas. A primeira, lista de *ad-blocking*, possui sites a serem bloqueados que foram selecionados com base no idioma do navegador. A segunda lista mantém os anúncios aceitáveis, ou seja, funciona como uma lista branca (*whitelist*). Contudo, o usuário é livre para desativar essas listas e adicionar ou criar outras [Adblock Plus 2014]. Outra vantagem do software é o fato de não coletar todos os dados do usuário e a maioria dos dados (por exemplo, os sites que o usuário visita) nunca é enviada para os servidores do fabricante [Adblock Plus 2014].

#### 2.7.2.4. StopFingerprinting

O StopFingerprinting é uma extensão, disponível nos navegadores Mozilla Firefox e Google Chrome, que acessa propriedades do navegador e do sistema operacional do usuário e, consequentemente, envia os dados coletados para um servidor seguro no INRIA [INRIA 2014]. Os dados coletados somente serão utilizados para fins de pesquisa dentro

INRIA e não serão compartilhadas com ninguém fora da INRIA. Em outras palavras, é uma extensão para proteção e análise de *fingerprinting* e sua função principal é recolher *fingerprinting* de navegadores, a fim de analisá-los.

Dentre as informações coletadas via StopFingerprinting, ignorando as comumente capturadas, destacam-se: (i) *Mime-types*; (ii) Constantes Matemáticas; (iii) informações das câmeras; (iv) Informações dos microfones, incluindo codecs, ganho, nível de silêncio, nível de supressão de ruído e muito mais; (v) Se existe um acelerômetro ativado e (vi) Se existem teclados virtuais e físicos.

#### 2.7.2.5. Lightbeam

O Lightbeam é um *add-on* para Firefox, no qual, uma vez instalado e ativado, registra todos os Cookies de rastreamento salvos no computador do usuário [Mozilla 2014]. Desta maneira, exibe um gráfico das interações e conexões de sites visitados e os locais de rastreamento para onde eles fornecem estas informações. De acordo com a Mozilla, através deste *add-on* os usuários podem ver onde e como eles estão sendo rastreados por anunciantes.

O funcionamento do Lightbeam é simples. Quando ativado e o usuário visita um site, o *add-on* cria uma visualização de todos os terceiros (empresas) que atuam nessa página em tempo real. A visualização padrão é chamada de gráfico de ponto de vista. Quando o usuário navega para um segundo site, o *add-on* destaca os terceiros que estão ativos lá e mostra aqueles que foram vistos em ambos os locais. A visualização cresce a cada site que o usuário visita e a todos os pedidos feitos a partir do navegador. Além da visão do gráfico, o usuário também pode ver seus dados em uma visualização tipo relógio, para examinar as conexões ao longo de um período de 24 horas ou em uma lista detalhada de sites individuais.

A Figura 2.17 dá uma demonstração da visão do Lightbeam.

### 2.7.3. Do-Not-Track (DNT)

O cabeçalho *Do-Not-Track* (DNT) é uma proposta de campo adicional no cabeçalho do HTTP, através do qual é possível que uma aplicação Web ou um usuário solicite a desativação do seu rastreamento. Definido inicialmente em [Mayer et al. 2011], DNT está sendo padronizado pelo W3C com o nome "Tracking Preference Expression" [W3C 2014]. Em linhas gerais, é um mecanismo de escolha do consumidor que abrange todas as formas de monitoramento de terceiros, seja para publicidade, análise ou qualquer outra finalidade. O primeiro navegador a implementar o recurso foi o Mozilla Firefox, mas hoje todos os fabricantes oferecem suporte ao DNT.

Sob um prisma mais técnico, [Eletronic Frontier Foundation 2014] define DNT como um mecanismo para proteger a privacidade on-line, uma vez que os anunciantes se utilizam de tecnologias de rastreamento cada vez mais sofisticadas. O DNT aceita três valores de configuração: (1) Caso o usuário não deseje ser rastreado (opt-out); (0) No caso do usuário autorizar ser rastreado (opt-in); (Nulo) - sem cabeçalho enviado - Se o usuário não manifestar uma preferência. O comportamento padrão exigido pela norma

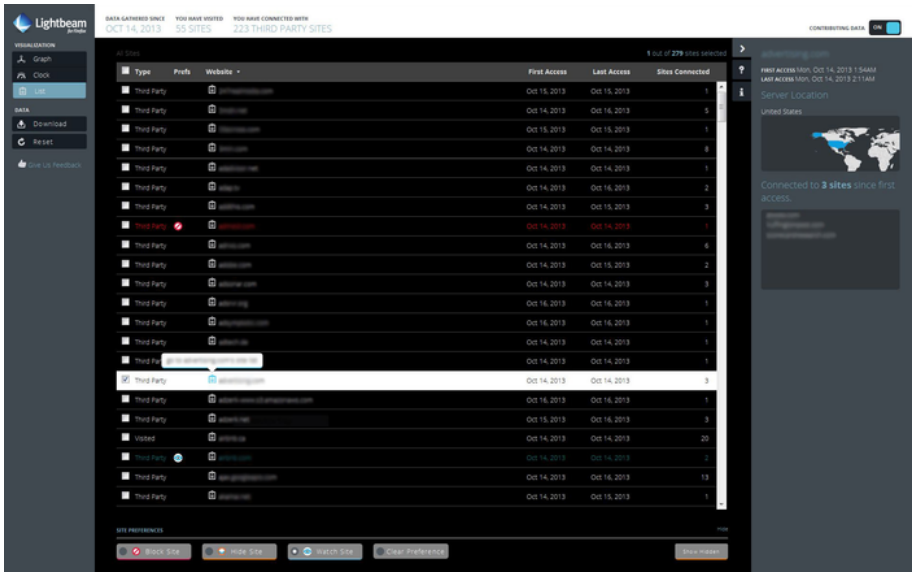



**Figura 2.17. Gráfico de navegação do Lightbeam. Fonte: [Mozilla 2014]**

é não enviar o cabeçalho, a menos que o usuário permita a configuração através do seu navegador ou sua escolha esteja implícita ao uso desse navegador específico.

### Funcionamento

A Figura 2.18 ilustra o funcionamento do DNT.

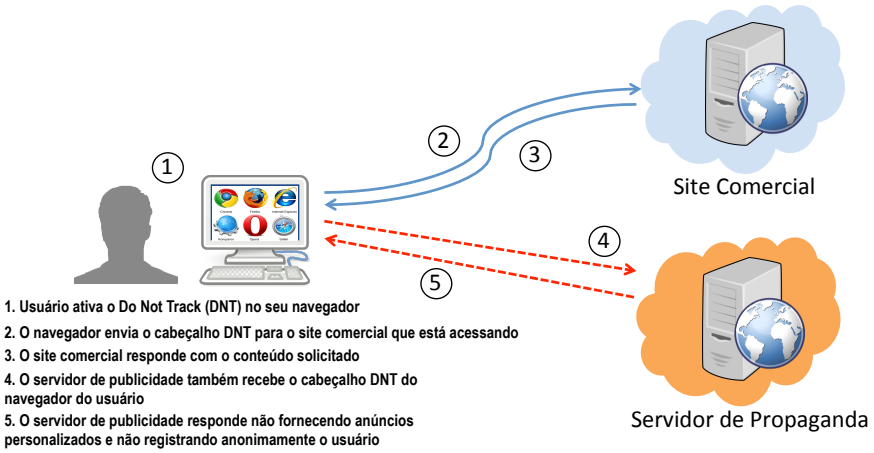



**Figura 2.18. Funcionamento do DNT**

Uma vez que o usuário configura seu navegador, todas as suas solicitações enviaram os cabeçalhos DNT indicando que aquele computador (usuário) não deseja ser rastreado. Assim, quando o usuário visita um site de comércio eletrônico, ele informa ao servidor do site comercial que não quer ser rastreado. Por sua vez, o site comercial entende e responde a solicitação feita pelo usuário. Assim, quando os dados começam a chegar no navegador, ele também envia aos servidores de publicidade on-line que também não quer ser rastreado. Os servidores de publicidade aceitam o uso do DNT e enviam ao navegador do usuário anúncios não personalizados e não ativam mecanismos de identifi-

cação.

Embora não haja nenhum padrão acordado, de modo universal, sobre o que uma empresa deve fazer quando detecta um cabeçalho DNT, um grupo de trabalho internacional de defensores de políticas, especialistas técnicos e empresas está tentando criar uma interpretação consensual. Atualmente, várias empresas implementam a política Do Not Track em seus sites. BlueCava, DailyMail, Pinterest e Twitter são alguns exemplos [Future of Privacy Forum 2014].

## 2.8. Considerações Finais

As técnicas de *device fingerprinting* estão se tornando cada vez mais presentes e comuns na Web e mesmo assim os usuários não são informados sobre isso. Mesmo aqueles que estão cientes de que estão sendo monitorados, por exemplo, como uma medida de proteção contra fraudes, confiam, sem nenhum tipo de prova, que as informações coletadas não são ou serão utilizadas para outros fins. Embora as formas, os mecanismos e suas consequências sejam conhecidos, questões como a privacidade dos usuários e as formas de evitar a coleta de informações ainda são motivo de discussão. A existência de uma indústria de propaganda especializada, evoluída e financeiramente feliz, aliada ao surgimento de novos serviços e tecnologias, a constante evolução tecnológica e ao crescente número de usuários (em atividades que envolvem dados pessoais e valores) impõe novos desafios na atividade de detectar e mitigar o *fingerprinting*.

Este Capítulo procurou introduzir o leitor no universo do *fingerprinting* na Web. Em primeiro lugar, o problema da coleta de informações de usuários/maquinas, sem prévia autorização, foi contextualizado e foram apresentadas definições bem como uma discussão sobre privacidade também foi efetuada. Os navegadores e as tecnologias atuais, causas ou fontes da capacidade de se executar um *fingerprinting*, foram apresentadas, tendo suas características e sua potencialidade para *fingerprinting* discutidas e exemplificadas. Em seguida, de maneira didática, foi apresentado um simples roteiro de construção de um mecanismo para *device fingerprinting*. Como prova da conceito e comprovação da capacidade de obter dados via Web, o resultado de um *fingerprinting* usando o site noc.to foi ilustrada. Perto do fim, as soluções existentes (tanto acadêmicas quanto comerciais) para realização de um *fingerprinting* bem como as contramedidas foram discutidas. Para estimular ainda mais o leitor, serão discutidas algumas questões em aberto e, por fim, serão feitas as observações finais.

### 2.8.1. Pesquisas em aberto

Algumas questões sobre *device fingerprinting* que ainda estão em aberto serão discutidas a seguir:

- Hoje em dia, as principais defesas contra *device fingerprinting* são os plug-ins e extensões para navegadores, que permitem ao usuário não informar dados ou alterar valores que identificam seu navegador para um servidor. Contudo, os trabalhos tem mostrado que as técnicas mais recentes de *device fingerprinting* podem facilmente superar tais extensões. Isso porque os navegadores modernos são enormes pedaços de software, cada um com suas peculiaridades. Assim, melhorar a construção dos navegadores é uma forma de limitar o *device fingerprinting* e uma fonte

de trabalhos.

- É consenso que quanto melhor for o software do navegador, melhor será sua resposta a *fingerprinting*. Nesse sentido, uma solução simples pode ser barrar os scripts de *fingerprinting* toda vez que forem carregados nos navegadores, semelhante à maneira como funcionam os bloqueadores de anúncios. Mantendo uma *blacklist* de roteiros problemáticos, uma extensão *anti-fingerprinting* poderia detectar seu carregamento e proibir sua execução. Contudo, surgem dois desafios. O primeiro é manter a *blacklist* constantemente atualizada a fim de acompanhar as mudanças que *trackers* certamente fariam em resposta. Outro desafio é que não se sabe se o carregamento de scripts de *fingerprinting* são necessários para o funcionamento de determinados sites. Mesmo que isso não seja necessário agora, os sites poderiam ser alteradas para recusar o carregamento de suas páginas a menos que os scripts *fingerprinting* estejam presentes e operacionais, o que desencorajaria as pessoas de tentar interferir com eles.

- Dado que a publicidade é a indústria número 1 da Web e que o rastreamento é um componente crucial dessa indústria, o mecanismos de geração de perfil do usuário e as técnicas de *device fingerprinting* vão existir por um bom tempo. Entretanto, regulamentos mais rigorosos e contramedidas mais eficazes podem desacelerar essa indústria e impedir piores e maiores abusos.

- Já existem empresas que oferecem navegação baseada em nuvem. Entretanto, não está claro se os navegadores que estão expostos a potenciais *fingerprinters* realmente operam na nuvem. Ainda assim, não há nenhuma razão para pensar que um sistema de prevenção de *device fingerprinting* para um navegador em nuvem não possa ser projetado.

### 2.8.2. Observações Finais

Atualmente, não existe uma lâmpada mágica que forneça a resposta definitiva contra *device fingerprinting*, mas algumas ações podem e devem ser tomadas:

- Aumentar o número de pesquisas nessa área visando, inicialmente, resolver problemas específicos. O apoio de agências governamentais, órgãos de fomento à pesquisa e a própria iniciativa privada devem servir de fontes financiadoras.

- Estimular a realização de eventos, workshops, conferências sobre o tema. Tais encontros servem como ponto de partida para troca de conhecimentos e o surgimento de novas parcerias.

- Desenvolver um modelo jurídico de alcance global para definir a privacidade na Web. Este trabalho deve ser conduzido por especialistas de modo a garantir da melhor forma a flexibilidade da Internet.

- Aumentar o desenvolvimento e uso de soluções contra *device fingerprinting*. Mesmo que não encerrem o problema, podem diminuir a incidência de problemas de privacidade.

- Por fim, educar os usuários Internet para torná-los mais conscientes dos riscos existentes para seus computadores e sistemas.

## Referências

[Acar et al. 2014] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and Diaz, C. (2014). The web never forgets: Persistent tracking mechanisms in the wild. In *21st ACM Conference on Computer and Communications Security*, ACM CCS 2014.

[Acar et al. 2013] Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., and Preneel, B. (2013). Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1129–1140, New York, NY, USA. ACM.

[Adblock Plus 2014] Adblock Plus (2014). Adblock plus surf the web without annoying ads. `https://adblockplus.org/`. [Online, Acessado 25/09/2014].

[Barth 2011] Barth, A. (2011). HTTP State Management Mechanism. RFC 6265 (Proposed Standard). `http://www.ietf.org/rfc/rfc6265.txt`.

[Bos et al. 2011] Bos, B., Celik, T., Hickson, I., and Lie, H. W. (2011). Cascading style sheets level 2 revision 1 (css 2.1) specification. W3c recommendation, W3C. `http://www.w3.org/TR/CCS2`.

[CCEE 2012] CCEE (2012). Cross-Browser Performance. `http://ccee.org.br/ccee/documentos/CCEE_056670`. [Online, Acessado 15/08/2014].

[Contextis 2011] Contextis (2011). WebGL - More WebGL Security Flaws. `http://www.contextis.com/resources/blog/webgl-more-webgl-security-flaws/`. [Online, Acessado 15/09/2014].

[Cooper et al. 2013] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and Smith, R. (2013). Privacy Considerations for Internet Protocols. RFC 6973 (Informational). `http://www.ietf.org/rfc/rfc6973.txt`.

[Crockford 2008] Crockford, D. (2008). *JavaScript - the good parts: unearthing the excellence in JavaScript*. O'Reilly.

[Eckersley 2010] Eckersley, P. (2010). How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg. Springer-Verlag.

[ECMA International 2011] ECMA International (2011). *Standard ECMA-262 - ECMAScript Language Specification*. 5.1 edition. `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.

[Eletronic Frontier Foundation 2014] Eletronic Frontier Foundation (2014). Do not track. `https://www.eff.org/issues/do-not-track`.

[Evidon 2014] Evidon (2014). Ghostery. http://www.ghostery.com/. [Online, Acessado 20/09/2014].

[Fielding et al. 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). http://www.ietf.org/rfc/rfc2616.txt.

[FireGloves 2014] FireGloves (2014). Firegloves - cross-browser fingerprinting test 2.0. http://fingerprint.pet-portal.eu/?menu=6. [Online, Acessado 21/09/2014].

[Forshaw 2011] Forshaw, J. (2011). WebGL - A New Dimension for Browser Exploitation. http://www.contextis.com/resources/blog/webgl-new-dimension-browser-exploitation/. [Online, Acessado 15/09/2014].

[Fulton and Fulton 2011] Fulton, S. and Fulton, J. (2011). *HTML5 Canvas - Native Interactivity and Animation for the Web.* O'Reilly.

[Future of Privacy Forum 2014] Future of Privacy Forum, F. (2014). About do not track. http://allaboutdnt.com. [Online, Acessado 20/09/2014].

[Giura et al. 2014] Giura, P., Murynets, I., Piqueras Jover, R., and Vahlis, Y. (2014). Is it really you?: User identification via adaptive behavior fingerprinting. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, CODASPY '14, pages 333–344, New York, NY, USA. ACM.

[Goodin 2013] Goodin, D. (2013). Stealthy technique fingerprints smartphones by measuring users movements. http://arstechnica.com/security/2013/10/stealthy-technique-fingerprints-smartphones-by-measuring-users-movements/. [Online, Acessado 19/09/2014].

[Group 2014] Group, K. (2014). Webgl 2 specification. Khronos draft, Khronos Group. https://www.khronos.org/registry/webgl/specs/latest/2.0/.

[Hors et al. 2003] Hors, A. L., Hégaret, P. L., and Stenback, J. (2003). Document object model (DOM) level 2 HTML specification. W3C recommendation, W3C. http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109.

[INRIA 2014] INRIA (2014). Stopfingerprinting - how trackable is your browser in a long term? https://stopfingerprinting.inria.fr. [Online, Acessado 25/09/2014].

[Kamkar 2010] Kamkar, S. (2010). Evercookie. http://samy.pl/evercookie/. [Online, Acessado 15/08/2014].

[Kirk 2014] Kirk, J. (2014). Canvas fingerprinting online tracking is sneaky but easy to halt. http://www.pcworld.com/article/2458280/canvas-fingerprinting-tracking-is-sneaky-but-easy-to-halt.html. [Online, Acessado 21/09/2014].

[Kristol and Montulli 1997] Kristol, D. and Montulli, L. (1997). HTTP State Management Mechanism. RFC 2109 (Proposed Standard). http://www.ietf.org/rfc/rfc2109.txt.

[Mayer et al. 2011] Mayer, J., Narayanan, A., and Stamm, S. (2011). Do not track: A universal third-party web tracking opt out. IETF Draft.

[Mayer 2009] Mayer, J. R. (2009). *"Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0*. PhD thesis, Princeton University.

[McDonald and Cranor 2011] McDonald, A. M. and Cranor, L. F. (2011). A survey of the use of adobe flash local share objects to respawn http cookies. *Journal of Information Policy*, 7(3):639–687.

[Microsoft 2014] Microsoft (2014). Microsoft silverlight 5. http://www.microsoft.com/silverlight/. [Online, Acessado 21/09/2014].

[Modernizr 2014] Modernizr (2014). Modernizr: the feature detection library for html5/css3. http://modernizr.com. [Online, Acessado 24/09/2014].

[Mowery et al. 2011] Mowery, K., Bogenreif, D., Yilek, S., and Shacham, H. (2011). Fingerprinting information in JavaScript implementations. In *Proceedings of Web 2.0 Security and Privacy 2011 (W2SP)*, San Franciso.

[Mowery and Shacham 2012] Mowery, K. and Shacham, H. (2012). Pixel perfect: Fingerprinting canvas in HTML5. In Fredrikson, M., editor, *Proceedings of W2SP 2012*. IEEE Computer Society.

[Mozilla 2014] Mozilla (2014). Lightbeam shine a light on whos watching you. https://www.mozilla.org/en-US/lightbeam/. [Online, Acessado 15/08/2014].

[Mulazzani et al. 2013] Mulazzani, M., Huber, M., Leithner, M., and Schrittwieser, S. (2013). Fast and reliable browser identification with javascript engine fingerprinting. In *Web 2.0 Workshop on Security and Privacy*, (W2SP).

[Nikiforakis et al. 2014] Nikiforakis, N., Acar, G., and Saelinger, D. (2014). Browse at your own risk. *Spectrum, IEEE*, 51(8):30–35.

[Nikiforakis et al. 2013] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. (2013). Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 541–555, Washington, DC, USA. IEEE Computer Society.

[Pemberton 2002] Pemberton, S. (2002). Xhtml 1.0: The extensible hypertext markup language (second edition). World Wide Web Consortium, Recommendation REC-xhtml1-20020801.

[Perry et al. 2013] Perry, M., Clark, E., and Murdoch, S. (2013). The design and implementation of the tor browser [draft]. https://www.torproject.org/projects/torbrowser/design/. [Online, Acessado 15/09/2014].

[Shankland 2011] Shankland, S. (2011). Microsoft declares webgl 'harmful' to security. http://www.cnet.com/news/microsoft-declares-webgl-harmful-to-security/. [Online, Acessado 24/09/2014].

[Slivinski 2011] Slivinski, A. R. (2011). Desenvolvimento de uma metodologia para avaliação de desempenho gráfico 3d de plataformas com suporte ao webgl. Technical report, Unioeste - Universidade Estadual do Oeste do Paraná.

[Soltani et al. 2010] Soltani, A., Canty, S., Mayo, Q., Thomas, L., and Hoofnagle, C. J. (2010). Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*. AAAI.

[StatCounter 2014] StatCounter (2014). Statcounter globalstats. http://gs.statcounter.com. [Online, Acessado 15/09/2014].

[Tanner 2013] Tanner, A. (2013). The web cookie is dying. here's the creepier technology that comes next. http://www.forbes.com/sites/adamtanner/2013/06/17/the-web-cookie-is-dying-heres-the-creepier-technology-that-comes-next/. [Online, Acessado 20/09/2014].

[Tor Project 2014] Tor Project (2014). The design and implementation of the tor browser. https://www.torproject.org/projects/torbrowser/design/.

[Unger et al. 2013] Unger, T., Mulazzani, M., Fruhwirt, D., Huber, M., Schrittwieser, S., and Weippl, E. (2013). Shpf: Enhancing http(s) session security with browser fingerprinting. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 255–261.

[W3C 2014] W3C (2014). Fingerprinting guidance for web specification authors. http://w3c.github.io/fingerprinting-guidance/.

[W3C 2014] W3C (2014). Html5: a vocabulary and associated apis for html and xhtml. http://www.w3.org/TR/html5/.

[W3C 2014] W3C (2014). Tracking preference expression (dnt). W3c working draft, W3C. http://www.w3.org/TR/tracking-dnt/.

[Weinberger 2011] Weinberger, A. (2011). The impact of cookie deletion on site-server and ad-server metrics in australia. http://www.comscore.com/Insights/Presentations-and-Whitepapers/2011/The-Impact-of-Cookie-Deletion-on-Site-Server-and-Ad-Server-Metrics-in-Australia-An-Empirical-comScore-Study.

# Capítulo 3

# Botnets: Características e Métodos de Detecção Através do Tráfego de Rede

Kaio R. S. Barbosa, Gilbert B. Martins, Eduardo Souto, Eduardo Feitosa

***Abstract***

*The passive monitoring and analysis enable the identification of suspected communication patterns of infected machines (bots) in network traffic. These hosts are often associated with malicious computers networks (botnets) used to spread malicious code (virus and worms) and denial of service (DDoS) attacks. Although researchers have been creating and evolving techniques to mitigate this type of application, attackers are also developing new mechanisms to hinder such an analysis. Thus, the technique of passive analysis is a viable alternative to find suspicious behaviors in unknown network traffic (zero-day) or when a worm was not fully explored by reverse engineering techniques. This chapter presents the strategies used to detect botnets through passive analysis of network traffic, separated in function of communication protocols commnly used to the implementation of command and control botnet channels, as well as a basic set of tools to support the process to detect this kind of malicious activity.*

***Resumo***

*O monitoramento e análise passiva permitem identificar padrões de comunicação suspeitos de máquinas infectadas (bots) no tráfego de rede. Tais hosts frequentemente estão associados a redes de computadores maliciosas (botnets) utilizadas para proliferação de códigos maliciosos (vírus e worms) e ataques de negação de serviço (DDoS). Embora os pesquisadores venham criando e evoluindo técnicas para mitigar esse tipo de aplicação, os atacantes também desenvolvem novos mecanismos para dificultar tal análise. Assim, a técnica de análise passiva é uma alternativa viável para encontrar comportamentos suspeitos no tráfego de rede desconhecidos (zero-day) ou quando um worm ainda não foi totalmente explorado pelas técnicas de engenharia reversa. Este Capítulo apresenta as estratégias utilizadas para detecção de botnets através da análise passiva do tráfego de rede, separadas em função dos protocolos de comunicação comumente utilizados para*

*implementação dos canais de comando e controle de botnets, bem como um conjunto básico de ferramentas para apoio ao processo de detecção deste tipo de atividade maliciosa.*

## 3.1. Introdução

Nos últimos anos, os códigos maliciosos (ou *malware*) deixaram de ter como função principal danificar ou tornar inoperantes os sistemas e máquinas alvos de seus ataques. Hoje em dia, os sistemas computacionais são infectados principalmente para o roubo de informações ou para o acesso aos recursos computacionais disponíveis nestes sistemas. Em geral, a meta primária dos atacantes é obter o controle total dos recursos sem afetar o comportamento usual do sistema até que estes desejem fazer uso dos recursos que agora estão sob seu controle. Neste modelo de atividade maliciosa, quanto maior for a distribuição do *malware*, maior será o poder computacional à disposição de seus criadores.

Formalmente, um *malware* é definido como um software que "deliberadamente", cumpre a intenção prejudicial de um atacante [Egele et al. 2008]. Contudo, o maior problema dos *malware*s tradicionais está relacionado ao procedimento de manutenção dos mesmos [Tiirmaa-Klaar et al. 2013]. Como os atacantes frequentemente não possuem acesso remoto aos hosts, é dificil corrigir o funcionamento do software malicioso. Por esse motivo, os *malwares* atuais são projetados para permitir que uma entidade externa mantenha o máximo de controle dos dispositivos infectados.

As máquinas contaminadas por *malware*, tipicamente passam a fazer parte de uma rede maliciosa, denominada de *botnet*, e são comumente conhecidas pela alcunha de zumbis ou apenas pelo nome de *bot*. O conceito de *botnets* está associado ao conjunto de máquinas comprometidas que permitem ao atacante controlar remotamente os recursos computacionais e realizar atividades fraudulentas ou ilícitas [McCarty 2003b, Freiling et al. 2005].

O *malware Storm*, que se espalhou por mais de dois milhões de computadores, deu ao seu proprietário um poder computacional agregado superior ao de um supercomputador [Colajanni et al. 2008]. Tal poder computacional desse e de outros *malwares* tem sido usado pelos atacantes, também chamados de *botmasters*, para promover ataques aos mais diversos provedores de serviço, instituições governamentais e outras empresas cujo modelo de negócios é fortemente atrelado ao uso da Internet.

Para reforçar a efetividade dos ataques contra alvos maiores, os *botmasters* necessitam aumentar o número de *bots* e, para isso, empregam diferentes técnicas para espalhar códigos maliciosos. No início, o procedimento para tornar uma máquina infectada não era completamente automatizado e necessitava de interação humana como clicar em um link, que conduzia ao download de códigos maliciosos, ou abrir um e-mail com um software malicioso anexado. Atualmente, este procedimento tornou-se cada vez mais automatizado. Por exemplo, `SDBot` [McFee 2003] pode se propagar usando técnicas como compartilhamento de arquivos, redes `P2P`, *backdoors* deixadas por *worms* ou ainda explorar vulnerabilidades comuns do sistema operacional *Windows*. Além disso, os ataques e a capacidade de comunicação do *bots* tem sido melhorados. Por exemplo, uma máquina infectada com o `Agabot` [Schiller and Binkley 2007] pode participar de

ataques de negação de serviços distribuídos (DDoS), servir como proxy para difusão de spam, criar túneis para outras redes de dados (GRE tunneling) e capturar senhas.

Este Capítulo tem como objetivo fornecer os subsídios para o entendimento das *botnets*. O foco é dado as técnicas de análise passiva do tráfego de rede e o seu emprego na detecção de *botnets*. Para tanto, o Capítulo empregará um enfoque teórico apoiado por exemplos práticos para garantir o entendimento do processo de identificação de *botnets*, desde a estratégia para coletar o tráfego até a confirmação de comportamento suspeito.

### 3.1.1. Organização do Capítulo

Este Capítulo está organizado da seguinte maneira: a Seção 3.2 apresenta um histórico e as definições sobre as *botnets*, incluindo métodos utilizados para propagação, ciclo de vida das *botnets*, como é feita a comunicação entre o operador da rede maliciosa e os *bots* e arquiteturas das *botnets*. A Seção 3.3 discute métodos e estratégias adotadas para detecção de *botnets* através da análise do tráfego de rede. Nessa Seção, os principais protocolos de rede utilizados para comunicação com os *bots* são descritos. A Seção 3.4 aborda algumas técnicas e ferramentas para coleta e análise do tráfego. Além disso, esta Seção apresenta exemplos de bibliotecas de programação de rede que podem auxiliar no processo de coleta e análise. Por fim, a Seção 3.5 apresenta as considerações finais e os problemas em aberto sobre o tema.

## 3.2. Botnets

### 3.2.1. Entendendo o problema

A preocupação com o correto funcionamento na comunicação em redes de computadores tem origem desde a concepção inicial da computação distribuída, na década de 80 [Shoch and Hupp 1982]. Uma aplicação com problema de funcionamento, mesmo que não intencional, poderia interromper a operação e a comunicação da rede, ocasionando problemas governamentais, financeiros e empresariais. Diante desses problemas, em 1987 o primeiro trabalho direcionado à pesquisa de aplicações maliciosas ou vírus de computador foi publicado [Cohen 1987].

Para Cohen, a definição inicial de um vírus de computador denota que um programa pode infectar outros programas com uma versão evoluída ou melhorada de si mesmo. A preocupação de Cohen era que se um vírus pudesse infectar outros computadores, através de redes de computadores ao redor do mundo, a comunicação da sociedade moderna teria inúmeros problemas. Tal fato pôde ser constatado quase 20 anos após a publicação de Cohen, quando Furnel e Ward [Furnell and Ward 2004] apresentaram uma visão geral da evolução dos vírus de computadores, demonstrando que o emprego de técnicas de persuasão para que usuários iniciassem um vírus, assim como previsto por Cohen, já não eram necessárias.

Assim, ao invés de esperar a intervenção do usuário, um software malicioso (*malware*) tornou-se capaz de infectar outras aplicações, explorando as vulnerabilidades do sistema operacional ou de softwares instalados. Além disso, alguns *malwares* já empregavam técnicas sofisticadas (metamorfismo de código, por exemplo) para mudar o seu código a medida que se propagavam ou deixavam portas escondidas abertas (*backdoor*) para que

o computador fosse utilizado para outros serviços, tais como encaminhador de e-mail, servidor de proxy e ataques de negação de serviço (DoS - *Denial of Service*).

Um marco nesse sentido foi o *malware Code Red*, que em 2001 infectou centenas de milhares de computadores ligados à Internet em um período de 24 horas [Berghel 2001]. Além da infecção, tal *malware* lançava ataques de negação de serviço, em datas específicas, contra o site da Casa Branca (*whitehouse.gov*) nos EUA. No entanto, para encontrar uma nova vítima, o *Code Red* utilizava a geração aleatória de IP, o que muitas vezes resultava em duas ou mais infecções de um mesmo host [Moore et al. 2002]. Outra característica relevante é que este *malware* instalava um *backdoor* que permitia que um host comprometido fosse reaproveitado em outros tipos de ataques.

Em 2003, McCarty [McCarty 2003b] apresentou um dos trabalhos pioneiros na detecção de computadores infectados (máquinas zumbis). O autor observou um conjunto de máquinas comprometidas ingressando em um servidor `IRC` bem como as atividades executas por esse conjunto. McCarty define tais máquinas como *bots* e o conjunto de *bots*, controlados por um ou mais atacantes, como uma *botnet*.

Inicialmente, como proposta, a função de um *bot* era o gerenciamento e controle de canais *IRC*. No entanto, devido à flexibilidade e ao crescente número de computadores conectados à Internet, tais máquinas passaram a ser utilizadas para atividades ilegais como ataques de negação de serviços [Peng et al. 2007], envio de spam em massa [John et al. 2009], *phishing* [Souza et al. 2013], fraudes em sistemas de propaganda [Daswani and Stoppelman 2007] ou aluguel de *botnets* [Studer 2011]. As seções seguintes definem o conceito, os componentes, arquiteturas e protocolos utilizados nas *botnets*.

### 3.2.2. Definições

O conceito de *botnets* está associado ao conjunto de máquinas comprometidas que permitem ao atacante o controle remoto dos recursos computacionais para realizar atividades fraudulentas ou ilícitas [McCarty 2003b, Freiling et al. 2005]. Tais máquinas utilizam um software chamado de *bot* (da palavra robô), o qual liga os computadores infectados à uma infraestrutura de Comando e Controle (C&C).

Esta infraestrutura permite que os *bots* se conectem à uma entidade de controle, que pode ser centralizada ou distribuída. Para comunicação com os *bots*, um ou mais protocolos de comunicação são utilizados pelos operadores da rede. Tais estratégias permitem que a *botnet* continue operando mesmo em situações de interrupções por via judicial [Sinha et al. 2010], sequestro do canal de Comando e Controle [Stone-Gross et al. 2009] ou contra-ataques de inundação dos *bots* [Davis et al. 2008].

Para controlar as operações de uma *botnet* é necessário uma entidade externa, definida como *botmaster* [Stone-Gross et al. 2009]. Um *botmaster* coordena as ações realizadas por cada *bot*, incluindo estratégias de ataques como negação de serviço e envio de spam em massa. A Figura 3.1 apresenta os principais componentes e interações do uso de *botnets*. Além dos componentes já citados, uma *botnet* deve possuir vetores de propagação (*malwares*) capazes de infectar novos dispositivos. Em geral, novos *bots* podem ser obtidos através de dois métodos: autopropagação (método ativo) ou propagação por indução (método passivo) de *malware* [Shin et al. 2011].

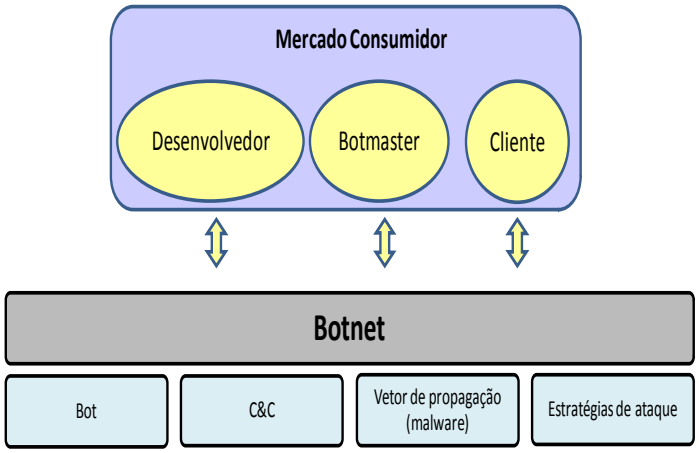



**Figura 3.1. Principais componentes e interações de uma *botnet*.**

Na autopropagação, o *bot* busca na rede outros dispositivos com vulnerabilidades que possam ser exploradas e que permitam acesso remoto. Por outro lado, na propagação por indução, técnicas de engenharia social (por exemplo, redirecionamento de URLs) tentam ludibriar o usuário para que o mesmo execute um *malware*. Em ambos os casos, após a infecção do dispositivo, o host busca o canal de C&C para notificar o *botmaster* e aguardar novas instruções.

Devido à importância do *malware* para operação da *botnet*, operadores da rede podem buscar novos métodos de infecção através de comunidades na Internet que pesquisam e vendem softwares maliciosos no mercado negro. Isso ocorre porque os desenvolvedores não são, necessariamente, os controladores da *botnet*. Nessas comunidades, *botmasters* e terceiros (clientes que desejam usar um *malware* como produto) podem comprar aplicações maliciosas pré-compiladas, pacotes de software para a criação personalizada de executáveis ou módulos de extensão. A *botnet Zeus* [Binsalleeh et al. 2010] é um típico exemplo desse tipo de comércio de *malware*.

Embora a definição de *botnets* possa atender as principais tecnologias atuais, o termo *botnet* pode ser refinado para abranger novas tendências e arquiteturas computacionais. Por exemplo, recentemente pesquisadores de uma empresa de segurança detectaram que um ataque de spam em massa tinha origem em roteadores, televisores e um refrigerador [ProofPoint 2014]. Tais dispositivos conectados fazem parte da Internet das Coisas (IoT - *Internet of Things*) e, portanto, uma vez infectados podem ser definidos como *bot* das coisas (ou *thingbots*). Além disso, a proliferação de aplicações maliciosas para ambiente móveis (por exemplo, *smartphones*) mostra que qualquer dispositivo conectado à Internet pode ser considerado um *bot* em potencial. Desta forma, esse trabalho define uma *botnet* como um conjunto de dispositivos eletrônicos comprometidos que são controlados remotamente por um ou mais operadores de *bots* (*botmasters*).

### 3.2.3. Evolução dos Ataques

A evolução de estratégias e protocolos usados nas *botnets* pode ser observada em diversos trabalhos [Peng et al. 2007, Feily et al. 2009, Rodríguez-Gómez et al. 2013]. Em 2003,

os *bots* utilizavam o protocolo `IRC` para entrar em contato com o C&C e os vetores de propagação englobavam o uso de mensagens spam, cavalos de Troia e vírus [McCarty 2003a]. Em meados de 2004, a exploração de vulnerabilidades nos navegadores Internet, especialmente o *Internet Explorer* da *Microsoft*, permitia que mais máquinas fossem comprometidas por *worms*. No ano seguinte, os ataques passaram a ser direcionados a redes de menor escala, permitindo aos atacantes ações como roubo de informações, fraudes e extorsões.

Em 2006, os ataques direcionados a usuários domésticos continuaram a crescer devido à exploração de vulnerabilidades nos navegadores *Internet Explorer* e *Firefox*. No ano de 2007, os EUA registraram os maiores números de servidores de comando e controle enquanto que a China o maior número de máquinas infectadas [Turner et al. 2007].

A mudança de estratégia dos ataques, em 2008, com foco para aplicações Web, permitiu que um número expressivo de computadores fossem atacados a partir de sites legítimos com problemas de vulnerabilidades como *cross-site scripts*, *phishing* e componentes de terceiros inseguros (*plugins*). Tal mudança partiu do princípio que esses ataques são mais difíceis de serem identificados por filtros da rede.

Os anos seguintes apresentaram um crescimento no número de máquinas comprometidas pertencentes a países emergentes. Enquanto em 2007 e 2008, o EUA possuía o maior número de máquinas infectadas, em 2010, o Brasil assumiu a terceira posição da lista dos países com maior número de atividades maliciosas. Uma razão para esse crescimento foi a melhoria da infraestrutura de banda larga de Internet no Brasil, permitindo que atacantes utilizem as máquinas remotamente por mais tempo [Fossi et al. 2010]. Outro fato relevante em 2010 foi a divulgação do kit de desenvolvimento do *malware* Zeus [Binsalleeh et al. 2010], o que permitiu que atacantes com pouca habilidade de programação pudessem criar novas *botnets*.

Em 2011 foi observada uma diversificação nos ataques em diferentes áreas [Fossi et al. 2011]. Em comparação com o ano de 2010, os ataques Web tiverem um aumento de 93%. Os ataques direcionados a dispositivos móveis também registraram um crescimento de 42% e mais de um milhão de *bots* foram identificados sob o controle da *botnet Rustock* [Fossi et al. 2011, Thonnard and Dacier 2011]. Além disso, também foram identificadas ameaças direcionadas a alvos industriais e corporativos, como *worm Stuxnet* [Rebane 2011] e *Hydraq* [Ferrer et al. 2010].

Apesar do número de spam ter reduzido em 2012 [CISCO 2013], o número de ataques e máquinas comprometidas não seguiu o mesmo comportamento devido ao uso de kits de *malware* que facilitaram a criação de novas *botnets*. Além disso, os ataques com objetivo de espionagem industrial ganharam mais evidência [Virvilis et al. 2013].

Finalmente, em 2013, a tendência em ataques à alvos pré-determinados continuou em alta, principalmente contra empresas de pequeno porte. Além disso, a tendência do aumento no número de ameaças para dispositivos móveis foi comprovada e o avanço de ameaças persistentes avançadas (APT - *Advanced Persistent Threat*) em ambientes corporativos [CISCO 2014].

### 3.2.4. Ciclo de vida dos bots

Para que uma *botnet* continue operando é importante que novos hosts sejam constantemente recrutados, uma vez que *bots* identificados por sistemas de detecção são comumente cadastrados em algum tipo de lista negra [Ishibashi et al. 2005]. Por isso, identificar hosts vulneráveis e comprometê-los é uma atividade vital para o sucesso de uma *botnet*.

O processo para encontrar hosts vulneráveis, explorá-los e torná-los membros da *botnet* é definido como ciclo de vida dos *bots*. Muitos trabalhos exploram o ciclo de vida das *botnets* para encontrar pontos de fraqueza e interromper as operações ilícitas da rede [Abu Rajab et al. 2006, Morales et al. 2009, Rodríguez-Gómez et al. 2013].

A Figura 3.2 ilustra o ciclo de vida de uma *botnet*. De maneira resumida, o ciclo de vida pode ser representado pelas seguintes etapas. No *Passo 1*, um membro da *botnet* identifica um host vulnerável na rede. Após a infecção desse host através dos vetores de propagação, consultas DNS são realizadas para encontrar o servidor que distribui software *bot* (*Passo 2*). No *Passo 3*, o host infectado baixa e instala o software *bot* para, finalmente, ingressar no canal de comando e controle (*Passo 4*).

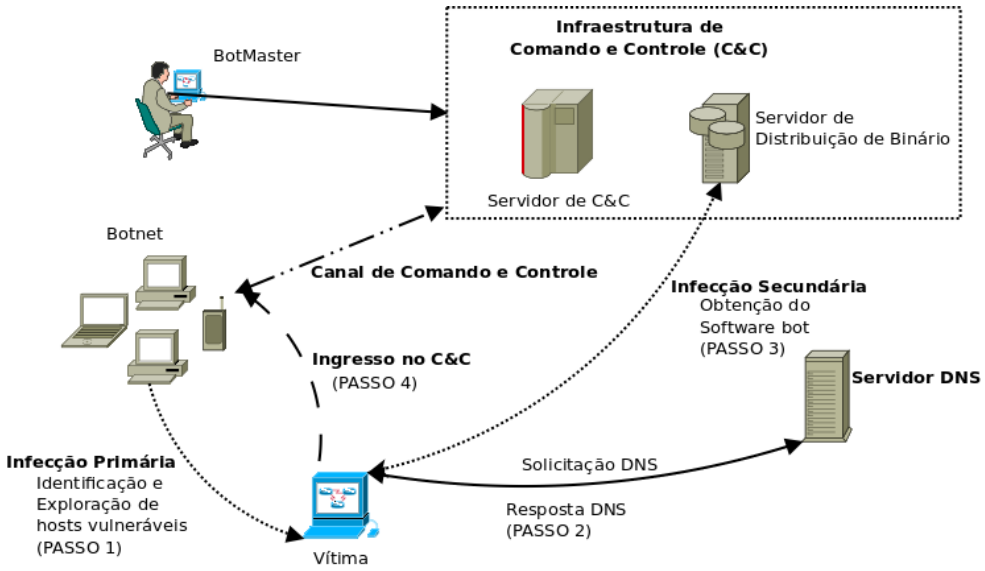



**Figura 3.2. Ciclo de vida da *botnet*.**

Para alguns trabalhos [Bazrafshan et al. 2013, Li et al. 2009a], o binário malicioso ou *backdoor* já faz parte do *malware* que infectou o dispositivo. Por outro lado, outros trabalhos observaram que o host comprometido realiza o download desse binário em um segundo instante (infecção secundária), a partir de um servidor que armazena o software malicioso [Abu Rajab et al. 2006, Feily et al. 2009]. Embora esses trabalhos divirjam quanto a presença do binário malicioso durante o estágio inicial de infecção, um dispositivo zumbi somente se torna útil à *botnet* a partir do momento em que o *botmaster* sabe da sua existência.

Como mostrado na Figura 3.2, a infecção do dispositivo pode ser dividida em duas etapas: primária e secundária. Na infecção primária, o host vulnerável é infectado por um

*malware* (vírus, *worms*, *trojans*, entre outros). Em seguida, na infecção secundária, o host infectado inicia o download do código malicioso para ingressar no C&C. A vantagem dessa estratégia é que a *botnet* pode suportar versões específicas do binário malicioso para arquiteturas de computadores distintas. Portanto, computadores e outros dispositivos com poderes limitados podem ingressar no C&C.

Para encontrar o canal de comando e controle e unir-se aos demais membros da *botnet*, o *bot* realiza um procedimento chamado *rallying*. O termo *rallying* refere-se ao momento em que um *bot* está se autenticando no servidor de comando e controle [Schiller and Binkley 2007]. Esse procedimento pode ser realizado através de uma abordagem estática ou dinâmica. Na abordagem estática, o host comprometido utiliza o endereço IP do servidor de C&C que se encontra codificado no próprio código que o infectou [Liu et al. 2009]. Embora tal estratégia seja simples, técnicas de engenharia reversa poderiam ser usadas para revelar o endereço IP do servidor de C&C, permitindo que a *botnet* fosse retirada de funcionamento [Egele et al. 2008].

Na abordagem dinâmica, o *bot* consulta servidores DNS (comprometidos ou não) para encontrar o endereço de nome de domínio que responde pelo C&C. Tal estratégia permite ao atacante realocar a sua rede de maneira rápida, caso servidores sejam sequestrados. Se a conexão falhar, o *bot* envia uma consulta DNS para receber o novo nome de domínio do servidor C&C. Existem alguns sites que fornecem esse serviço gratuitamente, como `dyndns.com`, onde é possível criar seu próprio domínio `yourname.dydns.com` e atribuir um IP dinâmico para este nome. Botnets recentes, por exemplo `Torpig` [Stone-Gross et al. 2009], usam domínios de fluxo rápido (*Fast Flux Domains*), onde cada *bot* independentemente usa um algoritmo de geração de nomes de domínios (DGA - *Domain Generatin Algorithm*) para computar uma lista de nomes de domínios. Quando nomes de domínios são usados, é necessário usar o sistema DNS para encontrar os endereços IP da máquinas a serem contactadas.

O sucesso de uma *botnet* está diretamente relacionado ao tempo que o *botmaster* mantém a rede em funcionamento. Por isso, diferentes arquiteturas tem sido utilizadas para tornar o canal de Comando e Controle mais resiliente.

### 3.2.5. Arquiteturas das Botnets

#### 3.2.5.1. Botnets baseadas em Arquiteturas Centralizadas

O funcionamento das *botnets* baseadas em arquiteturas centralizadas é semelhante ao clássico modelo cliente-servidor [Rodríguez-Gómez et al. 2013]. Neste modelo, uma infraestrutura de C&C centralizada é utilizada para estabelecer uma conexão entre o *botmaster* e os clientes da *botnet*, como ilustrado na Figura 3.3. Diferentes protocolos podem ser empregados para estabelecer a comunicação entre os *bot* e o canal de controle, como `IRC` [Schiller and Binkley 2007], `HTTP` [Perdisci et al. 2010] e `DNS` [Dietrich et al. 2011]. No caso do protocolo `IRC` , os *bots* se conectam diretamente no canal de comando e controle, permitindo ao *botmaster* avaliar os resultados de comandos enviados em um menor tempo. Em contra partida, *bots* que utilizam o tráfego `HTTP` para se comunicar com o C&C, devem solicitar em intervalos de tempo novas instruções de ataques. Tal modo de operação é conhecido como `PULL` [Binsalleeh et al. 2010].

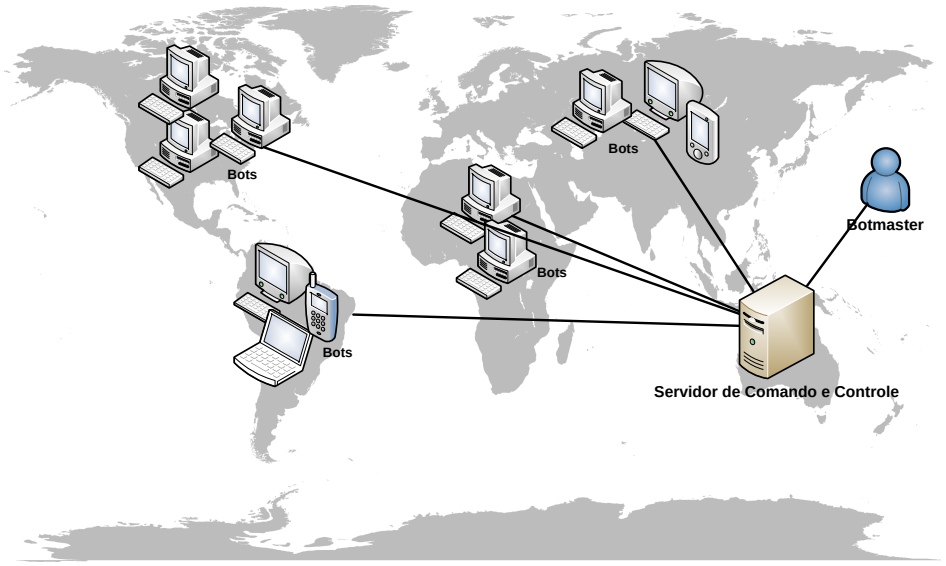



**Figura 3.3. Exemplo de uma arquitetura centralizada de *botnet*.**

Em geral, *botnets* baseadas em arquitetura centralizada oferecem menor latência de comunicação das mensagens trocadas entre o *botmaster* e os *bots* [Tyagi and G.Aghila 2011]. Por outro lado, o principal problema é que o servidor de C&C por si próprio é um ponto central de falhas, tornando-o não resiliente contra intervenções legais que visam interromper o seu funcionamento como sequestro e redirecionamento de domínio DNS [Stone-Gross et al. 2009] ou ataques de sessão `HTTP` ou `HTTPS` [Callegati et al. 2009].

### 3.2.5.2. Botnets baseadas em Arquiteturas Descentralizadas

Para oferecer robustez à infraestrutura da *botnet*, *botmasters* costumam empregar arquiteturas descentralizadas ou distribuídas [Tyagi and G.Aghila 2011]. Nesse tipo de arquitetura, protocolos `P2P` são frequentemente utilizados para fornecer resiliência, flexibilidade e escalabilidade a rede [Rodríguez-Gómez et al. 2013, Grizzard et al. 2007].

Nesse tipo de arquitetura, as informações sobre os membros ou da própria *botnet* estão distribuídas ao longo de toda rede maliciosa. Assim, *botnets* baseadas em arquiteturas descentralizadas são mais difíceis de serem desarticuladas, pois mesmo a descoberta de muitos *bots* não necessariamente significa a perda da rede da *botnet* inteira, visto que não existe um um servidor central de C&C [Rodríguez-Gómez et al. 2013]. A Figura 3.4 ilustra tal arquitetura de rede.

A união de um *bot* aos demais membros da *botnet* pode ser realizada através de consulta a uma lista estática de endereços, geralmente codificada no próprio software malicioso [Grizzard et al. 2007] ou por meio da varredura de endereços IP na Internet [Dagon et al. 2007]. Após a conexão com o C&C, um *bot* baseado em arquiteturas `P2P` busca, em servidores remotos, aplicações e instruções para atacar outras redes. Grizzard et al. (2007) apontam que, durante a fase de infecção do *bot* , se o conteúdo das informações foram cifradas usando criptografia de chave pública, a descoberta de outros componentes da infraestrutura é praticamente impossível. Além disso, a flexibilidade da

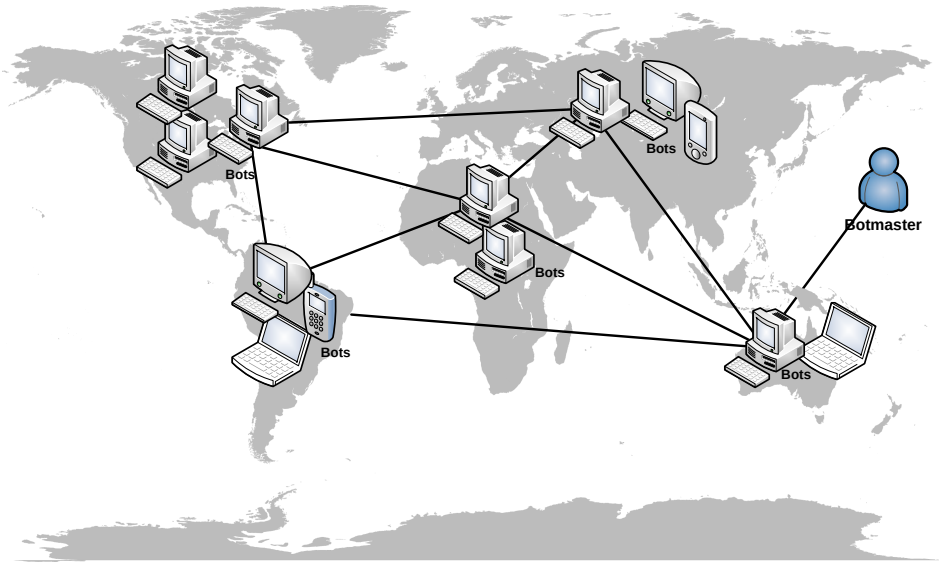



**Figura 3.4. Exemplo de uma arquitetura descentralizada de *botnet*.**

rede P2P dificulta a identificação dos membros e do canal de comando e controle. Em geral, as *botnets* descentralizadas operam em conjunto com outras redes P2P existentes como BitTorrent e Skype [Nappa et al. 2010], tornando difícil a distinção entre um host legítimo e um host malicioso [Yen and Reiter 2010].

As *botnets* baseadas em arquiteturas P2P podem operar tanto em modo estruturado quanto não-estruturado [Dagon et al. 2007]. No modo estruturado, existe um tipo de controle que pode ser utilizado para distribuir informações sobre a rede e armazenar arquivos. Um exemplo dessa arquitetura é o protocolo CHORD [Stoica et al. 2001]. Modelos não estruturados buscam pela descentralização completa da rede, onde cada nó participante opera tanto como cliente e servidor. A rede GNUTELA é um exemplo desse modelo [Chawathe et al. 2003].

Outra abordagem para manter o funcionamento da *botnet* é a organização dos nós em camadas e supernós [Chawathe et al. 2003]. Nessa estratégia, os clientes (*bots*) entram em contato com supernós, os quais fazem o roteamento das mensagens entre si até o destino final [Nappa et al. 2010]. A principal vantagem nessa arquitetura é que, caso um supernó seja retirado do ar, a *botnet* continua em operação devido outros supernós ainda receberem instruções do *botmaster*.

Para mitigar a proliferação de *botnets* baseadas em P2P, diversos trabalhos têm investigado técnicas para sobrescrever o índice da tabela de espalhamento distribuída com valores falsos [Wang et al. 2009]. Outra abordagem utilizada é infiltrar na *botnet* um grande número de nós falsos, visando interromper a comunicação entre os *bots* a partir da inserção do seu próprio endereço na lista de pares a servem comunicados [Holz et al. 2008b, Davis et al. 2008]. A análise do fluxo de rede também tem sido utilizada para identificar tráfego maliciosos em redes P2P [Rodríguez-Gómez et al. 2013, Peng et al. 2007].

#### 3.2.5.3. Botnets baseadas em Arquiteturas Híbridas

Uma forma de manter o equilíbrio entre a simplicidade do C&C centralizado e a escalabilidade de uma *botnet* descentralizada é unir ambos os conceitos em uma arquitetura híbrida. A *botnet* `Waledac` [Calvet et al. 2010] é um típico exemplo que explora essa arquitetura. A comunicação dos *bots* é realizada através do protocolo `P2P` e alguns membros das *botnets* funcionam como servidores de encaminhamento (*relay*), os quais escondem o verdadeiro endereço do canal de comando e controle [Tenebro 2008]. Outros exemplos de *botnets* híbridas são `Alureon/TDL4` e `Zeus-P2P` [Rodionov and Matrosov 2011]

Além dos protocolos mais usados como `IRC` e `HTTP`, a pluralização do acesso aos dispositivos móveis permite que novas topologias de *botnets* híbridas sejam propostas e identificadas. Por exemplo, as mensagens de texto de celular (mensagens `SMS`) podem ser utilizadas para operar o canal de comando e controle, enquanto o protocolo `P2P` pode permitir que os nós sejam controlados de maneira descentralizada [Mulliner and Seifert 2010]. Aplicações maliciosas para dispositivos móveis já são observadas desde a década passada, incluindo aplicações para envio de spam, controle remoto e disseminação de vírus [Dunham 2009].

### 3.2.6. O que uma Botnet Faz?

Independentemente da arquitetura utilizada, uma *botnet* é projetada para executar uma grande variedade de ataques. Esta Seção descreve algumas das principais atividades executadas através de *botnets*.

**Negação de serviço**

Ataques de negação de serviço procuram impedir que os usuários legítimos possam se utilizar de algum serviço ou recurso na rede, tornando-os indisponíveis [Kumar and Selvakumar 2011]. Sendo atualmente executados de forma distribuída (DDoS - *Distributed Denial of Service*), este tipo de ataque necessita de uma estrutura que pode ser provida inteiramente pela arquitetura padrão de uma *botnet*, onde os *bots* são utilizados para sobrecarregar o alvo de acordo com as instruções do responsável pela coordenação geral do ataque.

Para intensificar o volume de tráfego durante ataques de DDOS, atacantes podem manipular protocolos que não armazenam o estado da conexão como `DNS` e `SNMP`. Recentemente, o procolo `SNMP` foi demonstrado como vetor de ataques de reflexão de resposta [BITAG 2012], onde endereços IP de origem são forjados durante as consultas para que as respostas sejam redirecionadas para o alvo do ataque.

O tráfego DNS também é utilizado para amplificar o tamanho das respostas [Guo et al. 2006]. Nessa abordagem, o atacante formula solicitações para vários servidores de nomes recursivos (`rDNS`) na Internet, cujas respostas ultrapassam o limite 1500 bytes da unidade de transmissão máxima (`MTU`) da rede. Portanto, ao receber a quantidade de pacotes, o servidor aumenta a carga de processamento para atender todas as solicitações recebidas. Um exemplo deste tipo de ataque é apresentado na Figura 3.5.

Peng et al. [Peng et al. 2007] mostram que uma consulta DNS com 60 bytes

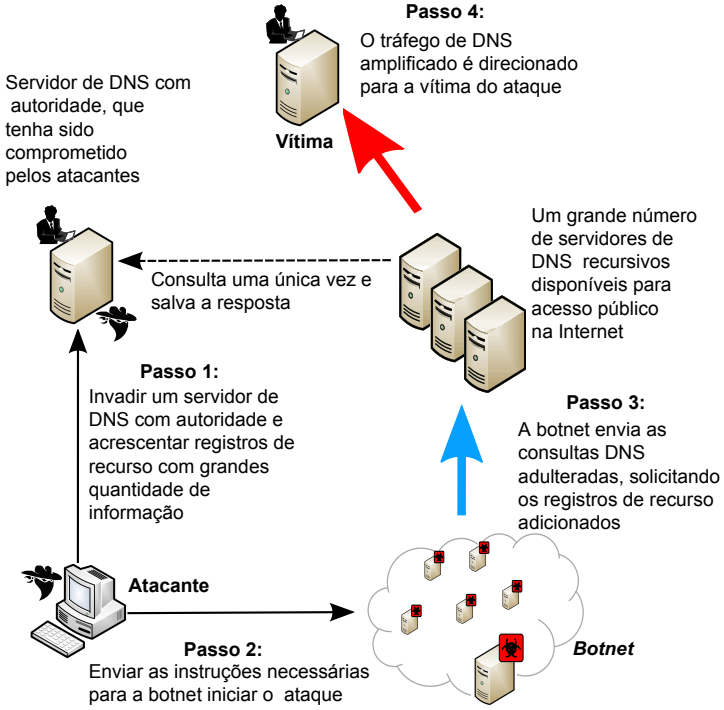



**Figura 3.5. Exemplo de ataque de negação de serviço do tipo Amplificação de DNS.**

de tamanho pode ser construída de tal forma a gerar uma resposta que possuirá mais de 4.380 bytes de tamanho, o que corresponde a um fator de ampliação superior a 73 vezes o tamanho da mensagem original. Para que este nível de amplificação seja alcançado, é necessário que o servidor de nomes, com a autoridade para responder a consulta, tenha sido previamente comprometido. Desta forma, o atacante pode manipular vários registros de recursos (RR) adicionais, os quais serão solicitados pelos membros da *botnet*.

Em contra partida, mesmo que os atacantes não tenham acesso ao servidor de DNS comprometido, as consultas DNS da *botnet* podem ser construídas para utilizar endereços públicos autênticos que já possuem tais registros.

### Ataques de reconhecimento de rede

Para identificar um IP válido, o atacante pode utilizar ferramentas que percorrem segmentos de endereçamento IP, como `nmap`[1]. No entanto, essa abordagem gera muito ruído (evidência) e pode ser utilizada para identificar o atacante.

Outra solução para encontrar endereços válidos é utilização do tráfego DNS. Nesse tipo de ataque, o tráfego DNS indica possíveis nós ativos e configurados na rede. Essa abordagem utiliza o registro de recurso do tipo `PTR`, o qual fornece o nome de um domínio a partir de um endereço IP. O registro reverso `PTR` é útil, pois cada IP acessível na Internet deve possuir um nome reverso [Barr 1996].

Além disso, programas de código malicioso também utilizam técnicas de reconhecimento de rede para encontrar possíveis nós comprometidos. Essa estratégia tem como

[1]http://nmap.org/

objetivo comprometer outros computadores localizados na rede. Dependendo da natureza do programa malicioso, o reconhecimento pode utilizar portas de serviço bem conhecidas ou comprometer outros computadores com base na confiança entre eles.

### Propagação de *spam* em massa

*Botnets* também são usadas com um meio eficiente para a propagação de mensagens *spam* em massa [John et al. 2009]. Antes de enviar uma mensagem não solicitada, o *bot* precisa percorrer endereços para encontrar servidores de e-mail abertos na Internet (*relay servers*). Para evadir de possíveis sistemas de detecção de intrusos (IDS) na rede, os *bots* solicitam essas informações diretamente aos servidores de nome raiz. A Figura 3.6 ilustra esse processo.

Para identificar ataques de *spam* em massa, uma alternativa é monitorar a frequência de utilização dos registros `A`, `PTR` e `MX` [Barbosa and Souto 2009]. Esses tipos de registros indicam fortes evidências de comportamento anômalo na rede. Por exemplo, uma variação do *worm Sobig* é observada a partir da análise dos registros `A` e `MX` [Levy 2003].

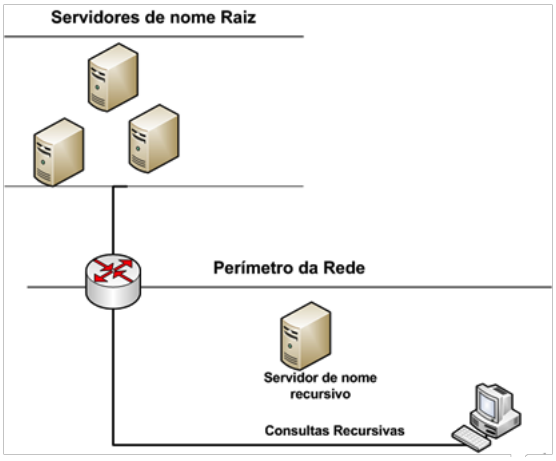



**Figura 3.6. Cliente infectado busca diretamente os servidores de raiz para não ser detectador por sistemas de intruso da rede.**

### Botnets como um modelo de negócios

Para Li et al. [Li et al. 2009b], antigamente os desenvolvedores de *malware* eram motivados por sentimentos de auto-realização, por diversão ou ainda pelo desejo de provar suas habilidades. Este comportamento mudou para motivações de ordem financeira, onde o modelo de negócio empregado envolve a criação, exploração e manutenção de *botnets*, que tem seus recursos postos à venda para aqueles interessados nas atividades ilegais executadas através destas redes. Como estas redes tendem a crescer de forma exponencial, isto se tornou um negócio da ordem de bilhões de dólares.

De acordo com Feily et al. [Feily et al. 2009], o controle de *botnets* se transformou num negócio ilícito muito atrativo, visto que tais redes são projetadas para proteger a identidade de seus criadores. Este anonimato é obtido por elementos como: *i*) um canal de comunicação de múltiplas camadas, o que dificulta a sua rastreabilidade; *ii*) pela distribuição física dos *bots* por diversos países, o que agrega diferenças de idiomas, zonas de tempo e leis; e *iii*) pelas próprias distâncias geográficas associadas aos elementos que

constituem a *botnet*, dificultando ainda mais o combate a este tipo de ameaça cibernética.

Um relatório técnico do *Communication Systems Group* (CSG), sobre os aspectos financeiros de diversas *botnets* [Studer 2011], afirma que existem duas formas de se ganhar dinheiro neste ramo de negócios escusos: *i*) venda de serviços, onde os *botmasters* são contratados para atacar alvos definidos pelos seus clientes; ou ainda *ii*) através do aluguel da *botnet*, onde os criadores originais cedem temporariamente o controle dos computadores infectados a terceiros. Os dados financeiros apresentados neste relatório são preocupantes:

- Apenas no ano de 2008, depois de executarem 190 mil ataques para seus clientes, os proprietários das *botnets* ganharam aproximadamente 20 milhões de dólares, o que torna este negócio muito rentável.

- Por apenas 67 dólares por hora, é possível alugar uma *botnet* com mais de mil computadores comprometidos à sua disposição, o que deixa o processo de aluguel de *botnets* atrativo.

- Uma análise dos dados de quatro empresas, para exemplificar o impacto destes ataques para os seus alvos, mostrou que o custo hora de cada ataque sofrido variou de 190 mil dólares até 19 milhões de dólares, com uma perda total média variando de 380 mil dólares até 114 milhões de dólares.

Levando estas características em consideração, Li et al. [Li et al. 2009b] propõem combater o fenômeno das *botnets* através de uma abordagem econômica, onde máquinas virtuais seriam propositadamente inseridas na rede para comprometer o desempenho da *botnet* e a efetividade de seu ataque, tornando seu uso inviável economicamente.

### 3.2.7. Abordagens usadas para detecção de *botnets*

Em função das características fundamentais das redes maliciosas, existem duas abordagens normalmente usadas para o combate às *botnets*: *i*) utilização de *honeypots*, e *ii*) monitoramento e análise do tráfego de rede [Feily et al. 2009].

Os mecanismos de *honeypots* permitem simular comportamentos de sistemas operacionais ou aplicações com problemas de vulnerabilidades. Nesse tipo de abordagem, o *honeypot* pode interagir (responder) aos ataques de maneira passiva ou ativa [Alata et al. 2007]. *Honeypots* passivos, ou de baixa interatividade, permitem detectar tentativas de varreduras de portas e servidores de redes falsos, enquanto os *honeypots* ativos, ou de alta interatividade, fornecem acesso ao atacante em um ambiente controlado. Tal ambiente é frequentemente monitorado para extração de registros de acesso e características de comportamentos maliciosos.

Vale ressaltar que *honeypots* são ferramentas que possibilitam a coleta de amostras de *malwares*, que serão analisadas com o objetivo de identificar as características e tecnologias associadas às redes maliciosas. Por outro lado, o monitoramento e análise passiva do tráfego de rede tem a responsabilidade de prevenir ou identificar as contaminações nas redes monitoradas.

O monitoramento e análise do tráfego podem ser desenvolvidos através do emprego de duas abordagens distintas: *a*) baseada em assinaturas e *b*) análise de fluxo de dados para detecção de anomalias [Feily et al. 2009].

As abordagens baseadas em assinaturas englobam sistemas de detecção de intrusão que monitoram continuamente o tráfego da rede, comparando elementos suspeitos com uma base de assinaturas ou com um conjunto de regras. Esta comparação permite identificar a presença de algum elemento malicioso previamente mapeado [Peng et al. 2007]. Por esta razão, esta técnica é eficiente e, em geral, resulta em baixas taxas de falso positivos. Entretanto, este modelo de identificação não é capaz de lidar com *malware* cuja descrição não tenha sido inserida previamente no sistema, o que deixa esta solução dependente de bases de dados atualizadas.

A detecção de *botnets* através da análise de fluxo de dados de rede permite a identificação de informações relevantes a respeito da estrutura de controle da *botnet*, tais como características e tecnologias empregadas para sustentar seu funcionamento. Esta abordagem é vital em situações onde o processo de engenharia reversa, que é a base para a construção de assinaturas de *malware*, não é suficiente para interromper a proliferação de ataques na rede [Stone-Gross et al. 2009]. O objetivo principal é inspecionar o fluxo para identificar comportamentos suspeitos como alta latência, grande volume de dados sendo transportados, presença de tráfego associado à portas de comunicação incomuns ou, ainda, comportamentos incomuns do sistema. Essas e outras características são utilizadas como indicadores da presença de componentes contaminados dentro da rede. Por exemplo, o monitoramento do tráfego `DNS` permite identificar o momento em que a máquina contaminada tenta se comunicar com o *botmaster*.

Para detecção de *botnets* através da análise de rede, diferentes abordagens podem ser empregadas. Agrupamento de padrões de comportamento semelhantes (conhecidos) ou comportamentos suspeitos (não conhecidos) em *botnets* [Gu et al. 2008a, Zeidanloo and Manaf 2010]. Técnicas da Teoria da Informação [Husna et al. 2008, Kang and Zhang 2009] e classificação automática através de aprendizagem de máquina [Zhao et al. 2013, Lin et al. 2009].

A Seção 3 apresenta alguns trabalhos que empregam diferentes abordagens e analisam diferentes aspectos do tráfego de rede para identificação de *botnets*.

## 3.3. Estado da Arte

Esta Seção descreve algumas das principais abordagens utilizadas para identificação e caracterização de redes *botnets*. Para um melhor entendimento, os trabalhos apresentados são categorizados de acordo com o protocolo utilizado pela *botnet*.

### 3.3.1. Botnets baseadas no Protocolo IRC

O protocolo IRC foi inicialmente projetado para estabelecer a comunicação entre usuários distribuídos geograficamente em um único servidor [Oikarinen and Reed 1993]. Devido ao crescimento da rede IRC, operadores desenvolveram ferramentas (*scripts*) para auxiliar na administração de usuários e canais de bate-papo. Essa possibilidade de comunicação em tempo real tem sido bastante explorada por atacantes para controlar máquinas infec-

tadas [Schiller and Binkley 2007].

No ciclo de vida de uma *botnet* IRC, cada *bot* tenta encontrar um servidor IRC através de uma consulta DNS. Uma vez que a comunicação entre *bots* e servidor IRC está estabelecida, o *bot* envia uma senha, através da mensagem `PASS`, para iniciar o processo de autenticação. Tal mecanismo pode ser mútuo: o *bot* se autentica para o servidor e o *botmaster* se identifica para o *bot* [Lu and Ghorbani 2008]. Para iniciar um ataque de negação de serviço, o *botmaster* envia um comando para a sala (por exemplo, `!ddos start IP`), como ilustrado na Figura 3.7.

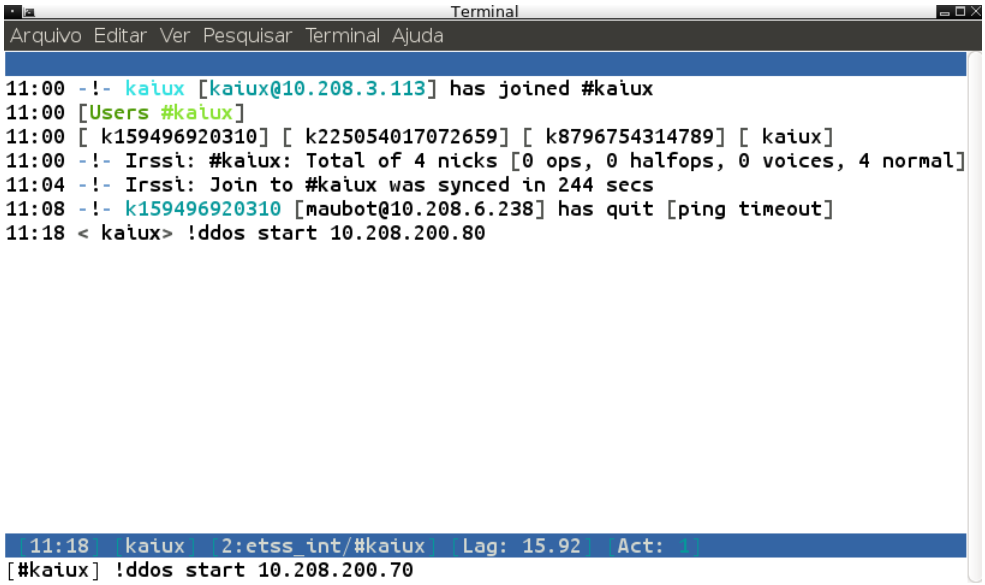



**Figura 3.7. Botmaster disparando ataque do tipo SYN**

A seguir são fornecidos alguns exemplos de *botnets* baseadas no protocolo IRC. Canavan apresentou uma análise da evolução histórica das *botnets* e do processo de comunicação entre o operador da *botnet* e os *bots* [Canavan 2005]. Essa análise demonstrou que o código fonte da *botnet* `GT-Bot` era utilizado para criar versões derivadas de outras *botnets* como `SpyBot`, `RBot` e `Agobot`. Em função disto, os *bots* criados possuíam semelhanças estruturais de código, o que permitiu o desenvolvimento de soluções de antivírus, baseadas em assinaturas e em padrões de comportamento, destinadas a identificar e interromper a proliferação de *botnets*. Entretanto, algumas *botnets* conseguiam enganar tais sistemas através de técnicas de polimorfismo de código como, por exemplo, a `PolyBot` [Hachem et al. 2011].

Mazzariello apresentou uma abordagem capaz de diferenciar padrões normais (emitidos por usuários legítimos) e maliciosos (emitidos por máquinas infectadas) em canais IRC [Mazzariello 2008]. O autor parte do princípio que as sequências de caracteres e estruturas gramaticais utilizadas por humanos mudam com mais frequência e não estão limitadas a um conjunto restrito de palavras ou dicionários, diferentemente de máquinas infectadas. O procedimento de detecção é baseado em um conjunto de características obtidas a partir de um canal IRC, como número de usuários, quantidade de palavras numa sentença, frequência de palavras, apelidos (*nicknames*) diferentes, quantidade de respostas semelhantes a partir de uma pergunta e número de comandos emitidos como `ping` e

join. Tais características são extraídas a partir de uma base de dados contendo amostras benignas e maliciosas, classificadas através de algoritmos de aprendizagem de máquina como SVM e J48.

Os autores em [Livadas et al. 2006] utilizaram as características do tráfego IRC como tamanho da carga útil, endereços IP de origem e destino, *flags* do cabeçalho TCP e total de bytes para detecção de anomalias. Tais características são extraídas de bases de dados rotuladas como legítimas e maliciosas. Livadas et al. compararam o desempenho de classificação usando Redes Bayesianas, Naive Bayes e árvore de decisão (J48). Os resultados mostraram que o classificador Naive Bayes detecta 97.51% de pacotes com tráfego malicioso e os algoritmos de árvore de decisão (J48) e redes bayesianas identificam 92.11% e 90% de tráfego de *botnet*, respectivamente. No entanto, para classificar comportamento malicioso na rede, a proposta utiliza características do tráfego quando o serviço de IRC é utilizado na sua configuração padrão e comunicação entre *bots* em texto claro.

Já em [Strayer et al. 2008], os autores apresentaram uma arquitetura para identificar *botnets* através da análise de características do fluxo de rede, como largura de banda, temporização dos pacotes e tempo de comunicação, em busca de indícios da presença de mensagens de C&C. A metodologia proposta está dividida em quatro etapas: (a) filtragem, (b) classificação, (c) correlação e (d) análise topológica. Na primeira etapa, os pacotes de rede são filtrados por listas de reputação, tipo de protocolo (TCP), flags do cabeçalho (SYN−RST), remoção de pacotes com mais de 300 bytes e fluxos de curta duração (tamanho superior a um pacote e com duração máxima de apenas 60 segundos). Isto remove pacotes que não são considerados suspeitos, como mensagens de IRC normais ou ataques de varredura, por exemplo.

Na etapa seguinte, os fluxos remanescentes são classificados através do algoritmo de aprendizagem de máquina Naïve Bayes, para determinar se existe alguma evidência de comunicação entre clientes e servidores IRC. A partir da identificação de comunicação IRC, os fluxos são correlacionados para rastrear hosts que possuam o mesmo tipo de comportamento, o que permite identificar outros elementos participando da mesma *botnet*. O agrupamento de fluxo é feito através de um algoritmo que calcula a distância entre eles. Na última etapa da metodologia, pares de fluxo que possuem maior afinidade são investigados através da análise topológica, a qual permite identificar membros da *botnet* e servidores que controlam o canal de C&C. Os experimentos demonstraram que esta metodologia é capaz de identificar a presença de *botnets*, mesmo em tráfegos contendo mais de um 1.3 milhões de fluxos IRC.

Ma et al. [Ma et al. 2010] apresentaram um algoritmo de detecção de botnets através da análise de sequência dos tamanhos dos pacotes IRC. Este trabalho parte do princípio que os padrões de comportamento apresentado por humanos e *bots*, quando estão interagindo com servidores de IRC, possuem características distintas. Enquanto o primeiro apresenta um padrão estocástico, o segundo demonstra possuir ciclos de periodicidade. Para identificar estas diferenças, os autores propuseram a construção de uma estrutura de dados baseada na sequência de conteúdo da conversa (CCS - *conversation content sequence*). Tal estrutura registra o tamanho dos primeiros 120 pacotes transmitidos entre o cliente e o servidor IRC. A partir daí, a metodologia de identificação é dividida

em 4 etapas.

Inicialmente, é feito um cálculo da média dos tamanhos de CCS, o qual é útil para identificar sequências de conteúdo de conversa produzidas seres humanos. Na segunda etapa, o conteúdo de CSS é convertido em uma *string S*, a qual é inserida em uma árvore de *substring* utilizando o algoritmo *Ukkonen*. Na etapa seguinte, esta árvore é usada para encontrar *substrings* que são mais frequentes e não se sobrepõem, sendo assim identificadas como instruções de ataques. Na quarta e última etapa, é feito um cálculo para determinar se *S* é ou não uma *string* de `botnet`, a partir da média dos tamanhos dos pacotes e do seu nível de periodicidade. Os autores validaram a proposta a partir de base de dados do projeto *HoneyNet* de fevereiro a abril de 2009[2] e tráfego benigno do provedor de IRC *FreeNode*[3]. Os experimentos demonstraram a eficiência deste método na identificação de *botnets*, mas os autores deixam claro que a inclusão de mensagens randômicas na comunicação de C&C pode dificultar a detecção por este método.

Os autores em [Carpine et al. 2013] apresentaram uma abordagem de detecção de *botnets* baseadas no monitoramento e construção de modelos comportamentais, para o tráfego IRC na rede e para as salas de bate-papo específicas. Este modelo foi construído com o objetivo de atingir três metas básicas: (a) ser capaz de efetuar detecção em tempo real, com um monitoramento on-line da rede; (b) possuir desempenho satisfatório, sem exigências elevadas de poder de processamento; e (c) fazer uso de modelos de classificação baseados em técnicas de aprendizagem de máquina.

Para atingir todos estes objetivos, os autores construíram um sistema que aprende de maneira gradativa o conceito atual, sem perder o que já fora aprendido. Este classificador foi baseado nos algoritmos SOINN [Shen and Hasegawa 2010] e KNN, com o segundo sendo utilizado para fornecer dados de entrada para que o primeiro aprenda novos conceitos. Esta metodologia utiliza ainda uma função de maximização, que busca encontrar o mínimo de instâncias e que ofereça o máximo de precisão para a base de dados utilizada. A detecção acontece em tempo real e possui o desempenho desejado, pois o classificador é capaz de processar 10.000 amostras por segundo, em um computador CPU *Intel Celeron* 530 com 1.5GB de memória RAM.

A partir das características identificadas nos trabalhos apresentados nesta Seção, é possível observar que a detecção de *botnets* baseadas no tráfego IRC pode ser dividida em duas etapas: identificação de tráfego IRC na rede e classificação de comportamento malicioso. A primeira tem como objetivo identificar e filtrar apenas o tráfego IRC dos demais protocolos na rede, pois algumas *botnets* não utilizam a porta padrão do serviço de IRC [Strayer et al. 2008]. Na etapa seguinte, as características da carga útil do IRC são utilizadas para fazer a distinção entre comportamentos suspeitos e benignos.

A Tabela 3.1 sumariza as principais características utilizadas em ambas às etapas.

Além das características citadas, alguns trabalhos incluem o uso de lista negra e expressões regulares para identificar apelidos de *bots*.

[2]http://www.edu.cn/HomePage/english/cernet/index.shtml

[3]http://www.freenode.org/

**Tabela 3.1. Características utilizadas para classificação de botnets baseadas no tráfego IRC.**

| | | |
|---|---|---|
| Características de Rede | Tupla Básica | IP de origem e destino |
| | | Porta de origem e destino |
| | | Flags do cabeçalho TCP (syn, syn-rst, ack) |
| | | Tempo de chegada do pacote. |
| | Estatística do Pacote ou Fluxo | Tamanho |
| | | Média do tamanho do pacote |
| | | Round-Trip-Time (RTT) |
| | | Total de pacotes e de bytes |
| | | Total de bytes |
| | | Total de pacotes enviados |
| | | Média de bytes e de bits por pacote no fluxo |
| | | Média de pacotes enviados por segundo |
| | | Porcentagem de pacotes enviados |
| | | Variância do RTT |
| | | Variância de bytes por pacote no fluxo |
| | | Intervalo de tempo de chegada de pacotes |
| Classificação do Comportamento | Comandos IRC | nick, user, privmsg, join, ping, pong, who, kick |
| | Estatísticas do Canal IRC | Total de usuários |
| | | Média de palavras em uma sentença |
| | | Distribuição de palavras do dicionário |
| | | Número de respostas iguais |
| | | Taxa de ingresso no canal |
| | | Número de mudanças de apelidos |
| | | Média e variância de mensagens privadas |
| | | Média de vogais, consoantes e caracteres especiais em uma sentença e no tópico do canal |
| | Estatísticas do Apelido | Tamanho do nickname |
| | | Quantidade de letras |
| | | Quantidade de números |
| | | Média de vogais, consoantes e caracteres especiais no apelido |

### 3.3.2. Tráfego HTTP

*Botnets* baseadas no tráfego HTTP apresentam algumas vantagens em comparação com as que utilizam o tráfego IRC. Uma das mais óbvias é que o tráfego IRC não é considerado comum em redes corporativas. Portanto, a utilização desse protocolo é sempre sinalizada como suspeita ou é simplesmente bloqueada. Por outro lado, o bloqueio do tráfego HTTP é inviável, pois restringe o acesso e a navegação na Internet. Em função disto, uma *botnet* baseada no tráfego HTTP é capaz de passar por filtros tradicionais de pacotes. Tal característica é explorada por *botmasters* para hospedar mecanismos de C&C [Perdisci et al. 2010] ou para utilizar redes sociais visando propagar aplicações de código malicioso [Souza et al. 2013, Freitas et al. 2014]

Outras *botnets* utilizam aplicações baseadas no tráfego HTTP para buscar e armazenar informações privilegiadas e disseminar códigos maliciosos. Por exemplo, [Boshmaf et al. 2011] simularam o comportamento de usuários e se infiltraram na rede social *Facebook*. Os autores conseguiram coletar, durante 8 semanas, informações úteis na identificação de uma pessoa, como número do telefone, idade, sexo, endereço residencial e email. Outros exemplos demonstram a existência de *bots* publicando informações fraudulentas sobre eleição norte americana na rede social *Twitter* [Ratkiewicz et al] ou utilizando os tópicos de tendência (*trending topics*) para divulgar sites maliciosos e ataques

de *phishing* através de URLs encurtadas [Souza et al. 2013, Freitas et al. 2014].

As *botnets* baseadas no tráfego HTTP precisam frequentemente entrar em contato com o canal de comando e controle para obter instruções do *botmaster*, estratégia conhecida como `PULL` (solicitação) [Gu et al. 2008b]. Devido a isto, alguns trabalhos identificaram essas rede a partir do grau de repetição de acesso [Lee et al. 2008]. No entanto, para evadir tal mecanismo de detecção, os *botmasters* adicionaram comportamentos aleatórios nas solicitações de comandos ao C&C, resultando no aumento da quantidade de falsos alarmes na rede [Eslahi et al. 2013]. Por isso, diversos trabalhos buscam identificar *botnets* baseadas no tráfego HTTP através da construção de modelos comportamentais para o tráfego da rede.

A combinação do tráfego HTTP com outros mecanismos para garantir maior resiliência da *botnet* é proposta em [Xiang et al. 2011]. A proposta tem como objetivo permitir que dispositivos de pequeno porte (*smarthones* e PDAs, por exemplo) acessem o C&C através de uma arquitetura híbrida. Nesta arquitetura, o tráfego HTTP é o protocolo de comunicação e as instruções de ataques são obtidas através de URLs criadas dinamicamente por algoritmos.

Por exemplo, se o canal de C&C é uma URL de um usuário em uma rede social, as máquinas infectadas gerariam nomes de usuários dinamicamente (UGA - *user generation algorithm*) para obterem as instruções de ataque. Tal estratégia é semelhante as redes de domínios de fluxo rápido, baseadas em algoritmos DGA. Desta forma, a proposta de Xiang et al. também pode ser detectada a partir de consultas com erros (`404` - página não encontrada).

Perdisci et al. [Perdisci et al. 2010] apresentaram um sistema de agrupamento de *malware* em nível de rede. Os resultados mostraram que é possível extrair automaticamente assinaturas de rede a partir de máquinas infectadas utilizando o tráfego HTTP como meio de disseminação. Para extração das características da rede, é usado um sistema de detecção de intrusos no roteador de borda para monitorar solicitações HTTP de saída. Este processo de detecção monitora a quantidade consultas, o tamanho e semelhanças estruturais entre URLs, a duração e o tipo de solicitação enviada. Inicialmente, os `malwares` são agrupados com base no comportamento para encontrar semelhanças estruturais entre as sequências de solicitações HTTP.

Este processo é dividido em duas etapas. Primeiro, são agrupados os *malwares* com base em elementos estatísticos (e.g.: número de solicitações HTTP e tamanho da URL). Na segunda etapa, após este agrupamento inicial, cada grupo é dividido em subgrupos classificados por características estatísticas de tráfego e com diferentes estruturas do HTTP. Para validação da metodologia proposta, os autores compararam os seus resultados com aqueles obtidos por três produtos de antivírus comerciais: *McAfee*[4], *Avira*[5] e *TrendMicro*[6]. Ao final deste processo de avaliação, nenhuma das soluções de antivírus foi capaz de detectar os *malwares* identificados pela metodologia proposta. No entanto, como a abordagem proposta monitora apenas o tráfego HTTP, uma *botnet* baseada em HTTPS seria capaz de burlar este processo de identificação.

[4]http://www.mcafee.com

[5]http://www.avira.com

[6]http://www.trendmicro.com

Haddadi et al. [Haddadi et al. 2014] detectaram *botnets* baseadas no tráfego HTTP através da análise de fluxo de rede ao invés da carga útil do pacote. Esta abordagem permite identificar comportamento malicioso mesmo que o tráfego entre o cliente e o canal de comando e controle esteja encriptado. Cada componente do fluxo da rede é representado através de uma tupla contendo cinco elementos extraídos a partir do cabeçalho do pacote: endereços IP de origem e destino, portas de origem e destino e protocolo de comunicação. Além disso, os autores extraem 21 características (como duração do fluxo, tamanho do fluxo e o tipo de serviço) que são utilizadas por dois classificadores: Naive Bayes e C4.5 (árvore de decisão). Para validação, Haddadi et al. utilizam uma base benigna[7] e outra maliciosa[8] e comparam os desempenho dos classificadores. Os resultados obtidos atestaram que, apesar do algoritmo C4.5 exigir mais tempo computacional para classificação, os resultados foram superiores aos do Naive Bayes, atingindo uma taxa precisão de 88% identificação das *botnets* avaliadas.

Cai e Zou [Cai and Zou 2012] investigaram como as características do tráfego HTTP podem ser agrupadas para identificar *botnets*. Algumas das características estudadas incluem campos do cabeçalho HTTP como `user agent`, tipo de conteúdo e tamanho do conteúdo. Como resultado deste estudo, foi identificado que *bots* não criam as solicitações HTTP contendo as informações completas ou corretas. Além disso, para que a *botnet* seja escalável, o tamanho do conteúdo da página, que fornece as instruções do comando e controle, deve ser pequeno. A partir da análise do tipo de conteúdo foi demonstrado que os *bots* fazem download de binários maliciosos com extensões de arquivos falsos, como .mp3 e .mp4, com o objetivo de evadir filtros de rede.

De maneira semelhante aos trabalhos de Perdisci et al. e Haddadi et al., a proposta de Cai e Zou mostra que a frequência de consultas entre máquinas infectadas e o servidor de C&C pode ser utilizada na detecção de *bots* na rede. No entanto, como apontado por Haddadi et al., caso o *botmaster* utilize algum tipo de criptografia entre o *bot* e o canal de comando e controle, esta proposta não seria mais viável como ferramenta de identificação.

A Tabela 3.2 sumariza as características utilizadas do tráfego de rede e os métodos estatísticos para detecção de *botnets* baseadas no tráfego HTTP.

### 3.3.3. Tráfego DNS

O sistema de nomes de domínio (*Domain Name System* - DNS) [Mockapetris 1987] é utilizado por muitas aplicações como navegadores Internet, aplicações de correio eletrônico e softwares de mensagens instantâneas. Este serviço associa nomes simbólicos aos respectivos endereços IP numéricos, permitindo aos usuários utilizar, com maior comodidade, recursos compartilhados em um ambiente de rede conectado.

Devido à importância do tráfego DNS para aplicações de rede, atacantes fazem uso das consultas como um dos principais pontos de partida para possíveis ataques de rede. As vulnerabilidades do DNS podem ser divididas em duas categorias: ataques que exploram falhas de segurança no serviço de tradução de nomes e ataques de rede que utilizam o tráfego DNS como ponto de partida para outros ataques de rede.

[7] http://www.alexa.com

[8] https://zeustracker.abuse.ch

**Tabela 3.2. Características utilizadas para classificação de botnets baseadas no tráfego HTTP.**

| | |
|---|---|
| Características de Rede | Métodos HTTP GET, POST, HEAD |
| | Nome do USER Agent |
| | Porta de destinos (80, 8080, 800*, 443) |
| | Campos do cabeçalho HTTP como tamanho do conteúdo (content-length) |
| | Tipo do conteúdo (content-type) |
| Classificação do Comportamento | Tamanho médio da URL |
| | Total de requisições |
| | Total de GET, POST e HEAD |
| | Média de argumentos passados na URL |
| | Média de bytes enviados no POST |
| | Média do tamanho das respostas |
| | Similaridade entre os valores dos argumentos passados |
| | Frequência de solicitações GET/POST |
| | Total de pacotes e bytes |
| | Total de pacotes ebits por segundo |
| | Total de bytes por pacote |

Na primeira situação, atacantes subvertem o funcionamento do protocolo para envenenar o resolvedor de um cliente ou servidor de nomes a fim de direcioná-lo a um site comprometido, como o ataque Kaminsky [Musashi et al. 2011]. Na segunda categoria, atacantes utilizam o tráfego DNS para identificar possíveis alvos configurados em redes remotas, encontrar servidores de correio eletrônico abertos ou para encontrar o endereço do servidor do canal de comando e controle durante a fase de rallying.

Por esses motivos, o tráfego DNS tem sido investigado para identificar tanto anomalias de rede quanto para entender o ciclo de vida de *botnets*. Por exemplo, no trabalho de Choi e Lee [Choi and Lee 2012], *botnets* são detectadas a partir do agrupamento de atividades similares no tráfego DNS, tais como consultas repentinas, quantidade de consultas únicas, nome de domínio requisitado por múltiplos endereços IP de origem em um determinado intervalo de tempo e consultas para domínios dinâmicos DNS (`DDNS`). O processo de detecção utiliza similaridades entre consultas, listas negras para diferenciar entre domínios legítimos e maliciosos e aplicações de terceiros ( máquinas de busca e reputação web, por exemplo).

Para encontrar o canal de C&C, os *bots* podem utilizar consultas DNS estáticas ou dinâmicas. Na consulta estática, o host comprometido utiliza o endereço IP do servidor de C&C codificado no próprio software *bot* [Jakobsson and Ramzan 2008]. No entanto, apesar da estratégia ser simples, tal abordagem apresenta diversas fraquezas [Tiirmaa-Klaar et al. 2013]. Por exemplo, técnicas de engenharia reversa poderiam revelar o endereço IP da rede permitindo que a *botnet* fosse retirada de funcionamento. Outro problema ocorre na migração de uma *botnet* para outro canal de comando e controle, pois os membros da *botnet* devem atualizar o software bot e tal comportamento em massa pode indicar atividades suspeitas na rede [Choi et al. 2009].

Uma alternativa para contornar os problemas das listas estáticas é o uso da resolução dinâmica de nomes de domínios, a qual pode ser dividia em duas estratégias. Na primeira, um nome de domínio é gerado automaticamente por um algoritmo, chamado DGA (*Do-*

*main Generation Algorithm*) [Yadav et al. 2012]. Tais nomes de domínios também são conhecidos como domínios de fluxo rápido ou *domain flux*. Para ilustrar como nomes de domínios são gerados, a Listagem 3.1 apresenta um algoritmo que gera um nome a partir das sementes de entrada: ano, mês e dia. Por exemplo, os valores fornecidos como entrada (`2014, 9, 25`) resultam na sequência de caracteres `odnslofgimonbruy`, os quais podem ser utilizados como prefixo em domínios como `odnslofgimonbruy.br` e `odnslofgimonbruy.net`.

**Listagem 3.1. ]Exemplo de algorítimo que gera nome de domínio. [Autor desconhecido]**

```
1  def generate_domain(year, month, day):
2      """Generates a domain by the current date"""
3      domain = ""
4      for i in range(16):
5          year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) <<
               17)
6          month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0
               xFFFFFFF8)
7          day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) <<
               12)
8          domain += chr(((year ^ month ^ day) % 25) + 97)
9          print domain
10
11     return domain
12
13 print generate_domain(2014, 9, 25)
```

Na segunda estratégia para encontrar o endereço do C&C, o nome de domínio pode ser estático e as mensagens direcionadas ao servidor de comando são processadas por *bots* , que operam como intermediadores (*proxy*) entre a vítima e o verdadeiro servidor. Vale ressaltar que essa abordagem é frequentemente utilizada para esconder páginas de conteúdo malicioso ou ilícito [Holz et al. 2008a].

A Figura 3.8 ilustra o mecanismo conhecido como redes de serviço de fluxo rápido (FFSN - *fast-flux service network*). Antes de acessar à página maliciosamente forjada pelo *botmaster* , a vítima deve consultar o endereço do servidor que hospeda tal conteúdo. O objetivo das redes de fluxo rápido é manter o maior número de endereços IP associados a um nome de domínio. Por exemplo, uma vítima consulta o domínio `www.example.com` e obtém como resposta o endereço IP `192.168.0.8`. A cada consulta direcionada ao nome de domínio malicioso, novos endereços IPs, que fazem parte da *botnet* , são retornados. Estes endereços IPs são trocados com frequência, usando uma combinação de técnicas de *Round-Robin* e o tempo de vida (TTL) consideravelmente curto. Portanto, a vítima ao acessar o conteúdo malicioso, na verdade, está se comunicando com *bots* ao invés do servidor central (*mothership*).

Salusky e Danford [Salusky and Danford 2008] apresentaram um dos primeiros trabalhos dedicados à identificação e caracterização do tráfego de redes de fluxo rápido, incluindo as principais diferenças entre as redes *single-flux* e *double-flux*. Redes *single-flux* mudam apenas o registro de recurso (RR) do tipo `A`, enquanto as redes *double-flux* alteram os RRs `A` e `NS`, o qual contém o endereço IP do servidor de nomes que responde pelo domínio malicioso.

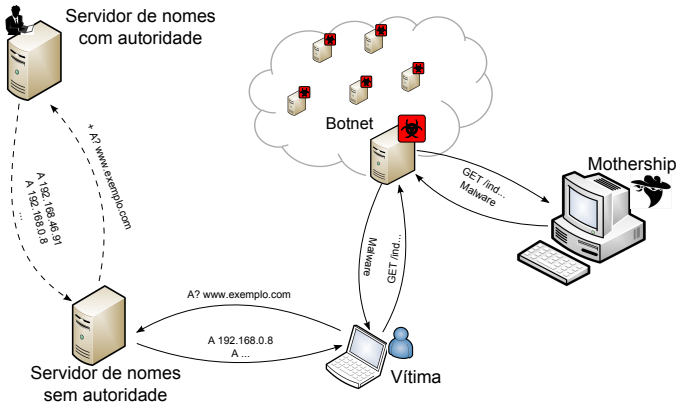



**Figura 3.8. Exemplo de rede de fluxo rápido.**

A proposta de Salusky e Danford para mitigar redirecionamentos de fluxo de rede é baseada no monitoramento do tempo de vida (`TTL`) das consultas DNS. Um domínio benigno possui um número estável de endereços IPs, enquanto um domínio de fluxo rápido emprega o `TTL` curto para que o nome de domínio seja atendido por um número maior de endereços IP distintos de máquinas infectadas.

No entanto, Holz et. al. [Holz et al. 2008a] mostraram que a utilização do `TTL`, apesar de ser um indicador relevante em redes de fluxo rápido, não é um fator definitivo na detecção de redes dessa natureza. Os autores apresentaram uma heurística que utiliza um conjunto de recursos de rede para distinguir entre tráfego de rede de fluxo rápido, redes de distribuição de conteúdo (CDN) e redes que usam o tráfego DNS para distribuição de carga em diferentes servidores. Para isso, foi empregado o número de endereços IP dos registros de recurso `A`e `NS`, a quantidade de números de sistemas autônomos (ASN) por domínio, os valores do `TTL` e a localização geográfica do IP.

Para distinguir entre redes CDN e FFSN, Holtz et al. dividem a quantidade total de endereços IP encontrados no registro `A`, pela quantidade de endereços retornados em uma única consulta do tipo `A`. Caso o valor encontrado seja próximo a 1, o domínio é benigno, caso contrário o domínio apresenta comportamento de CDN ou FFSN. Para classificação de domínios de fluxo rápido, os autores compararam as características de base de dados de domínios benignos (e.g.: `alexa.com` e Projeto Dmoz[9]) com base de dados de domínios maliciosos, a qual foi criada a partir de `URLs` presentes em emails categorizados como *spam*. Em comparação com as CDNs, os resultados mostram que máquinas com software de *fast-flux* possuem baixa disponibilidade de acesso, visto que podem ser desligadas a qualquer momento. Além disso, foi possível identificar que 30% dos domínios encontrados em SPAM são hospedados em redes de fluxo rápido.

Nazario e Holz [Nazario and Holz 2008] também utilizaram os domínios de `URLs` para encontrar redes de fluxo rápido. Para apoiar esse processo, os autores empregaram engenharia reversa de *malware*, listas negras e análise manual dos domínios. Além das características observadas por Holz et al., a proposta de Nazario e Holz extrai informações da zona autorizativa do domínio (`SOA`) e a distribuição dos endereços IP encontrados.

[9]Projeto Dmoz disponível em:http://www.dmoz.org/

Caso um domínio seja confirmado como suspeito, esse endereço é adicionado em uma base de dados que, posteriormente será utilizada como referência de reputação de IPs. Os resultados mostraram que 76% do total de 928 endereços de domínios `.com` são redes de fluxo rápido e que 83% dos endereços IP são utilizados em múltiplos domínios, demonstrando, consequentemente, uma sobreposição de endereços IP por nomes de domínios.

De uma maneira geral, as soluções propostas dependem de fontes externas para identificar tráfego suspeito, tal como listas negras, engenharia reversa de código e base de dados de *spam* [Holz et al. 2008a, Nazario and Holz 2008, Passerini et al. 2008]. Por esses motivos, Perdisci et al. propuseram um modelo passivo para detecção e monitoramento de redes de fluxo rápido a partir da análise de comportamento histórico do tráfego DNS [Perdisci et al. 2009]. Os autores observaram as características do domínio como o tempo de vida (TTL); quantidade de endereços IP que atendem um domínio; a proporção de ASN por domínio; tempo da última consulta do domínio; o total de consultas novas destinadas a um domínio dentro de um espaço de tempo; e total de endereços IP que buscaram por um domínio. Perdisci et al. classificaram os domínios maliciosos através do algoritmo de árvore de decisão (C4.5). Os resultados mostraram que o modelo proposto acerta em 99.7% e erra 0.002%. No entanto, apesar do percentual de precisão ser elevado, o modelo utiliza o comportamento histórico do DNS e uma janela de monitoramento extensa; 24 horas.

Huang et al. [Huang et al. 2010] apresentaram um mecanismo de detecção de *fast-flux* em tempo real, baseado em localização geográfica dos hosts. De uma maneira geral, enquanto os trabalhos anteriores utilizavam a relação temporal (`TTL`), Huang et al. utilizaram a relação espacial (latitude, longitude) entre os hosts para detecção de domínios maliciosos. Os autores partem do princípio que tais redes possuem máquinas infectadas em todo globo. Por tanto, o fuso horário é utilizado para calcular a dispersão entre os hosts da rede. Isto é, redes benignas possuem concentração de hosts na mesma zona de fuso horário, enquanto as redes maliciosas a distribuição é mais dispersa.

Os resultados, em comparação com [Holz et al. 2008a], mostraram que a solução espacial é mais rápida na detecção de *botnets* de fluxo rápido (0,5 segundos). A solução proposta consegue identificar 98.16% dos domínios maliciosos, enquanto Holz et. al. 90.80%. Huang et al. utilizaram uma base de dados de terceiros para obter a localização geográfica de endereços IP. Isso significa que, caso um endereço IP não esteja cadastrado na base de dados, o sistema proposto não atingirá o seu objetivo.

Mecanismos como *single-flux* e *double-flux* mostram que *botmasters* buscam estratégias para dificultar o interrompimento da *botnet* . Ao combinar redes de fluxo rápido com algoritmos que geram nomes de domínios dinamicamente, o problema torna-se ainda mais crítico. Antonakakis et al. [Antonakakis et al. 2010] apresentaram um sistema de reputação dinâmica para nomes de domínios novos ou desconhecidos no ccTLD do Canadá (`.ca`). O sistema, denominado `Notos`, analisa o comportamento do tráfego DNS e atribui uma pontuação de acordo com as atividades relacionadas com o domínio investigado. O comportamento histórico do DNS é obtido a partir de bases legítimas e maliciosas. O comportamento legítimo é coletado em servidores recursivos DNS, enquanto o tráfego malicioso é obtido através de sensores de rede como *honeypots*, *spamtraps* e *sandboxes*. Para facilitar a análise do tráfego, os autores agruparam domínios

de comportamentos semelhantes observando características léxicas do nome de domínio (e.g.: frequência de caracteres e tamanho do nome) e de rede (e.g.: número AS e total de endereços IP associados ao domínio). A partir dessas características, nomes de domínios que apresentem os comportamentos conhecidos podem ser classificados como benignos ou suspeitos.

Os autores em [Shin and Gu 2010] apresentaram uma visão global do número de máquinas infectadas pelo *malware Conficker* (25 milhões de vítimas). Os resultados mostraram que 99% de vítimas de um domínio específico são clientes de banda larga e responsáveis pela maioria dos emails de *spam* enviados. Esse resultado é semelhante ao trabalho de Barbosa e Souto (2009), o qual aponta o uso de clientes de banda larga do Brasil (.br) varrendo por endereços de servidores de email. Para Shin e Gu, os domínios `.br`, `.net` e `.cn` representam 24% do total de vítimas do *Conficker*, onde o continente Asiático e da América Sul são os principais focos de infecção. Além disso, os autores mostraram que apenas 17,18% do total de 24.912.492 vítimas foram corretamente identificadas por listas negras, o que indica outros tipos de identificação são necessários.

Stone-Gross et al. [Stone-Gross et al. 2009] apresentaram uma visão geral de como funciona um canal de comando e controle de *botnets*. Durante o período de 10 dias, o canal de C&C da *botnet Torpig* foi sequestrado permitindo que pesquisadores pudessem entender seu funcionamento, aplicando técnicas de engenharia reversa ao algoritmo de DGA do bot. Stone-Gross et al. observaram que durante a comunicação com o C&C, cada *bot* possuía um identificador único, o qual foi utilizado para estimar com precisão o tamanho da *botnet* (aproximadamente 180 mil máquinas). Em 10 dias, mais de 49 mil novas máquinas foram infectadas e buscaram o C&C monitorado pelos pesquisadores.

Já Yadav et al. [Yadav et al. 2012] partem do princípio que domínios gerados por algoritmos são diferentes em termos de padrões de nomes quando comparados aos domínios registrados por seres humanos. Para entender a relação entre domínios, Yadav et al. observaram a distribuição alfanumérica e o bigrama dos caracteres utilizados no nome de domínio. Além disso, os autores assumem que os nomes gerados por algoritmos tendem a ser impronunciáveis, apesar da *botnet* `Kraken` [Royal 2008] ser capaz de produzir nomes mais próximos da língua inglesa.

Para validar a proposta, os autores utilizaram domínios legítimos, obtidos através da varredura do espaço de endereçamento do IPv4, enquanto os domínios maliciosos foram coletados por domínios gerados pelas *botnets* `Conficker` e `Kraken`. O modelo proposto foi validado em um tráfego DNS real de grandes servidores de Internet da América do Sul e Ásia. Os domínios foram agrupados através de um conjunto de características, tais como nível do nome de domínio, relação entre endereços IP e a relação entre domínio e endereço IP. Durante os experimentos foi possível observar que a partir da distribuição de frequência dos caracteres, as letras 'm', 'o' e 's' em domínios legítimos não eram uniformes, enquanto as vogais 'a', 'e', e 'i' possuíam distribuição uniforme para domínios maliciosos.

A Tabela 3.3 sumariza as principais características utilizadas pelos trabalhos citados nesta Seção para detecção de redes e domínios de fluxo rápido. São apresentadas características comuns aos dois tipos de anomalia (*fast-flux* e *domain-flux*), onde as redes de fluxo rápido são identificadas através da correlação dos recursos de rede, enquanto os

**Tabela 3.3. Características para classificação de botnets baseadas no tráfego DNS.**

| | | |
|---|---|---|
| Fast-Flux | Recursos de Rede | Tempo de vida do pacote (TTL) |
| | | Endereço IP obtido através dos registros A, NS e SOA |
| | | Número do sistema autônomo (ASN) |
| | | Prefixo do endereço do roteador de borda (BGP) |
| | | Retorno da consulta reversa de um domínio |
| | | Data do registro do domínio (whois) |
| | | Nome da operadora que controla os registros de domínio |
| | Estatística de Rede | Total de endereços IP que respondem por um registro do tipo A durante um intervalo de tempo (t) |
| | | Distância entre os endereços que respondem por um registro do tipo A, NS e SOA |
| | | Total de endereços IP únicos obtidos em uma consulta do tipo A, NS e SOA |
| | | Distância entre os endereços IP de nameservers |
| | | Nomes dos roteadores obtidos através do traceroute |
| | | Desvio padrão do RTT entre o cliente e todos os endereços IP obtidos através dos registros A, NS e SOA |
| | | Total de endereços IP que se sobrepõem |
| | | Número único de endereços IP encontrados nos registros A e NS |
| | | Porcentagem de distribuição geográfica dos endereços IP |
| | | Média do TTL durante um intervalo de tempo (t) |
| Domain-Flux | Recursos de Rede | Consultas de domínios com erro (NX-Domain, SRVFail) |
| | Estatísticas do Nome de Domínio | Distribuição de frequência de caracteres de [a-z] e [0-9] |
| | | Total de caracteres |
| | | Total de níveis de sub-domínio |
| | | Frequência de caracteres por nível de domínio |
| | | Número de domínios que retornam o mesmo IP |
| | | N-grama do nome |
| | | Entropia dos sub-domínios |
| | | Média, mediana, desvio padrão e variância para o nome e sub-domínios especiais em uma sentença e no tópico do canal |
| | | Total de domínios de primeiro nível |
| | | Distribuição do total de níveis de sub-domínios |
| | | Similaridade entre domínios e n-gramas |

domínios de fluxo rápido são investigados pelas características do nome do domínio.

Vale ressaltar que além das características ilustradas na Tabela 3.3, alguns trabalhos também utilizam fontes externas como `URL` em *spam*, listas negras, listas brancas, *honeypots*, engenharia reversa, base de dados geográficas e o fuso horário.

### 3.3.4. Protocolos P2P

A computação ponto-a-ponto (`P2P`) tem promovido uma grande modificação nos padrões de uso da Internet nos últimos anos. Sua grande vantagem, em relação à computação cliente/servidor, é possibilitar a colaboração direta entre os usuários, sem depender de servidores administrados por terceiros [Kamienski et al. 2005]. Contudo, a falta de controle sobre a troca de conteúdo compartilhado nessas redes, tornou as aplicações `P2P` a principal fonte de compartilhamento de conteúdo ilegal na Internet.

Para detectar estes tipos de *botnets*, a maioria dos trabalhos propostos são base-

ados em técnicas de análise do tráfego P2P. Características como o par endereço IP e porta, protocolos utilizados, *strings* especificas contidas no pacote e a taxa de sucesso de comunicação do nó são comumente empregadas na análise do tráfego.

Para entender o funcionamento das redes P2P, Grizzard et al. (2007) investigaram o comportamento de computadores infectados com *worm Trojan.Peacomm* que utiliza o tráfego P2P para comunicação. Após a infecção de uma vítima, o *malware* utiliza uma URL para baixar os binários da injeção secundária, conforme descrito no ciclo de vida das *botnets* 3.2.4. Tal binário permite ao *bot* encontrar outros pares da rede P2P e oferecer serviços como *relay SMTP* ou coleta de informações sigilosas como o número de cartão de crédito e credenciais de banco.

Douceur [Douceur 2002] apresentou um ataque, denominado de Sybil, que explora a falta de uma autoridade de certificação em redes P2P para assumir múltiplas identidades e assim controlar a rede. Este tipo de ataque se baseia no fato de que é praticamente impossível, em sistemas computacionais distribuídos, que nós que não se conhecem apresentem identidades distintas convincentes. Um ataque Sybil é aquele em que um atacante subverte o sistema de reputação de uma rede P2P, criando muitas identidades e usando-as para obter vantagens. Baseado neste conceito, Davis et al. (2008) investigaram a viabilidade de um ataque *Sybil* para interromper a proliferação da *botnet Storm*. Ataques do tipo *sybil* simulam falsos pares (nós) na rede para sobrescrever o índice da tabela distribuída dos membros verdadeiros. Ao enviar dados falsos aos membros da *botnet*, um *bot* legítimo não consegue estabelecer a comunicação com outros *bots* da rede, pois as informações repassadas para envenenamento sobrescrevem os endereços dos *bots* válidos.

Nagaraja et al. [Nagaraja et al. 2010] desenvolveram o BotGrep, uma ferramenta para detectar *botnets* P2P baseadas nas trocas de mensagens entre pares de nós da rede, também conhecido como grafos de comunicação. Esta abordagem se baseia em grafos gerados a partir da estrutura de C&C da *botnet* P2P. O algoritmo BotGrep particiona de forma iterativa o grafo de comunicação, estreitando-o para destacar apenas os componente da *botnet*. Esta análise pressupõe que os hosts pertencente as *botnets* P2P tendem a ser mais conectados do que outros hospedeiros, o que permite sua separação de outros hosts benignos. Os resultados experimentais indicaram que esta técnica pode localizar a maioria dos *bots* com baixa taxa de falsos positivos.

Outro modelo de comunicação ponto-a-ponto é o serviço de mensagens curtas (SMS) utilizado por aparelhos celulares [Hua and Sakurai 2013]. Hua e Sakurai apresentam uma arquitetura de botnet baseada no protocolo *Bluetooth* e em mensagens de texto curtas utilizadas para contactar o servidor de comando e controle. Os autores demonstraram que tal *bonet* é possível simulando três cenários de comportamentos distintos, onde 100 *bots* estão distribuídos aleatoriamente em uma área de $100m^2$. O principal problema dessa arquitetura é que a mesma não é escalável, pois está limitada ao raio de alcance da tecnologia *Bluetooth*.

## 3.4. Ferramentas e Técnicas para Detecção de Botnets

Esta Seção apresenta as ferramentas e técnicas frequentemente utilizadas para a detecção de *botnets*. A literatura sobre o tema é extensa e, por isso, este trabalho são descritas

apenas algumas ideias e ferramentas comumente usadas para identificar comportamentos suspeitos na rede. Contudo, o material apresentado é suficiente para oferecer um visão geral a respeito deste assunto.

O processo de coleta do tráfego de rede deve seguir um roteiro predeterminado, respeitando os principais objetivos da pesquisa, para que dados irrelevantes ou desnecessários não sejam processados nessa etapa. Vale ressaltar que questões legais sobre monitoramento e armazenamento do tráfego devem ser consideradas. Portanto este trabalho assume que qualquer processo de coleta e armazenamento do tráfego, executado com base nas ferramentas e técnicas aqui apresentadas, possui algum amparo ético e legal.

Primeiro, será demonstrado como o tráfego de rede pode ser coletado através de ferramentas de captura (*sniffers*). Em seguida, são apresentados exemplos de bibliotecas de programação de rede que auxiliam no desenvolvimento de ferramentas específicas para coleta do tráfego. Finalmente será apresentado um estudo de caso que demonstra como a análise dos dados coletados pode ser utiliza para o processo de identificação de *botnets*.

### 3.4.1. Coleta e Armazenamento do Tráfego de Rede

Para coletar o tráfego de rede é preciso identificar possíveis pontos de coleta na rede. Em redes de computadores que utilizam comutadores (*switch*), o tráfego de rede deve ser replicado em um porta secundária (*span port*), a qual terá um *sniffer* capaz de processar e armazenar os pacotes de rede.

Entre as aplicações de *sniffers* mais conhecidas, podem ser citadas as ferramentas `TCPDump`[10], `Wireshark`[11] e `Snort`[12]. Para ilustrar como o tráfego pode ser coletado por tais ferramentas, considere o exemplo da Listagem 3.2. Por definição do protocolo `IRC`, os clientes da rede devem estabelecer uma conexão com o servidor através da porta `6667` [Oikarinen and Reed 1993]. Entretanto, como descrito na seção 3.3.1, os *botmasters* podem utilizar faixas de endereçamento de portas para evadir sistemas de intrusão. No exemplo em questão, o filtro do `TCPDump` coleta o tráfego `IRC` nas portas `TCP 6660-6669` e `7000` através da interface de rede `eth0`. Portanto, essa regra permite identificar e capturar o tráfego de qualquer aplicação que utilize estas portas.

**Listagem 3.2. Exemplo de coleta de tráfego `IRC` através do `TCPDump`**

```
1 tcpdump -n -l -i eth0 tcp portrange 6660-6669 or tcp port 7000
```

Semelhante às operações ilustradas pelo `TCPDump`, a ferramenta `Wireshark` oferece ao usuário da aplicação mecanismos de monitoramento do tráfego de rede através de uma interface gráfica mais amigável. Por exemplo, para filtrar as requisições `GET` de uma *botnet* baseada no tráfego `HTTP` (e.g.: `Zeus`) um usuário deve adicionar a seguinte opção `http.request.method == "GET"` no campo de filtros da ferramenta `Wireshark`, como ilustrado na Figura 3.9. Vale ressaltar que tal filtro também pode ser utilizado para criar estatísticas dos sites que tiveram o maior número de requisições originadas pelos *bots*.

[10]http://www.tcpdump.org
[11]https://www.wireshark.org
[12]https://www.snort.org

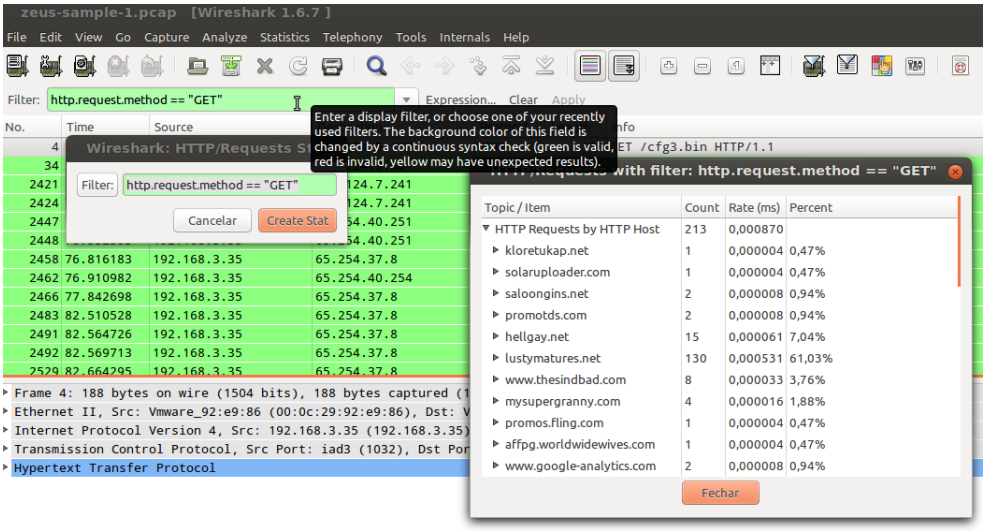



**Figura 3.9. Exemplo do total de solicitações do método `GET` enviados por um membro da botnet `Zeus`.**

Embora a versão em modo gráfico da ferramenta `Wireshark` permita analisar o tráfego de maneira mais simples, tal solução é inviável em ambientes onde o acesso é remoto e lento. Para superar este problema, o usuário pode utilizar uma outra versão desta ferramenta que opera em modo linha de comando, conhecida como `tshark`, a qual inclui todas opções que a versão em modo gráfico possui, porém em modo texto. Para ilustrar o emprego do `tshark` na coleta do tráfego, a Listagem 3.3 apresenta um exemplo da coleta do tráfego `DNS` durante 5 minutos na interface de rede `eth0` de um servidor Web. Esse filtro realiza uma comparação estatística dos registros de recursos (RR) `A` (`dns.qry.type==1`), `PTR` (`dns.qry.type==12`) e `MX` (`dns.qry.type==15`) considerando uma janela de tempo de 1 minuto (`-qz io,stat,60`). Após 5 minutos ou 300 segundos de atividade (`-a duration:300`), a execução do *sniffer* é interrompida automaticamente.

**Listagem 3.3. Exemplo de coleta de tráfego `DNS` através do `tshark`**

```
1  tshark -n -l -qz "io,stat,60,dns.qry.type==1,dns.qry.type==12,
       dns.qry.type==15" -a duration:300 -R dns -i eth0
```

Vale ressaltar que os métodos ilustrados acima são utilizados para monitorar o tráfego em tempo real. Portanto, antes de coletar e armazenar o tráfego, é importante planejar uma estratégia que defina os protocolos a serem filtrados, o tempo de coleta e o tamanho dos arquivos armazenados. A Listagem 3.4 ilustra dois exemplos de coleta de tráfego: `HTTP` e `DNS`. Nas linhas de 1-4, o tráfego `HTTP` (`tcp port 80`) é coletado a partir da interface de rede (`eth0`). As opções `-b duration:300` e `-w http.pcap` permitem ao `tshark` armazenar este tráfego em arquivos distintos, formados pelo prefixo *http*, quando o limite de tempo (`300`) for alcançado. De maneira semelhante, as linhas de 6-9 ilustram o processo de coleta do tráfego `DNS` (`udp port 53`), onde as opções do temporizador `-G 300` e o formato do nome do arquivo `-w "dns_%Y%m%d.%H%M%S.pcap"` permitem ao `TCPDump` criar múltiplos arquivos com prefixo *dns*.

**Listagem 3.4. Exemplo de coleta de tráfego HTTP e DNS através do TCPDump**

```
1 tshark  -n -l -i eth0 \
2         -b duration:300 \
3         -w http.pcap \
4         tcp port 80
5
6 tcpdump -n -l -i eth0 \
7         -G 300 \
8         -w "dns_%Y%m%d.%H%M%S.pcap" \
9         udp port 53
```

Apesar de poderosas, nem todos os elementos de redes (roteadores e *switches*) oferecem suporte a execução do Wireshark e TCPDump. Para essa situações, existem duas alternativas: *i*) o fabricante do equipamento deve fornecer uma ferramenta proprietária para suprir as operações de coleta e gerenciamento do tráfego da rede; e *ii*) o equipamento deve oferecer suporte a algum padrão de coleta de tráfego IP. Por exemplo, o protocolo Netflow [Claise 2004] é um recurso, que foi introduzido em roteadores Cisco, cuja função é coletar o tráfego de redes IP, tanto na saída quanto na entrada de uma interface.

Um fluxo de dados Netflow consiste numa tupla contendo o endereço IP de origem e destino, portas de origem e destino, protocolo, campos do cabeçalho do protocolo TCP como SYN, RST e FIN, tipo de serviço, total de pacotes, total de bytes, hora inicial e final que o fluxo foi visto na rede. O formato Netflow é frequentemente utilizado devido à maneira em que os dados do fluxo são organizados. Os fluxos são coletados por meio de algum tipo de técnica de amostragem, uma vez que a coleta de todas as informações do fluxo podem elevar o consumo de CPU de um roteador [Schiller and Binkley 2007].

Para leitura e processamento do formato específico do Netflow é necessário utilizar um conjunto de ferramentas do projeto NFDUMP[13]. A Figura 3.10 apresenta o resultado da execução do comando nfdump em um tráfego de rede previamente armazenado no formato NetFlow. É possível observar a hora inicial que o pacote foi visto, o tempo de duração do total, endereço IP de origem e destino, campos do cabeçalho, tipo de serviço (Tos), total de pacotes, total de bytes, total de pacotes por segundo (pps) e total de bytes por segundo (bps).

```
Date flow start         Duration Proto    Src IP Addr:Port         Dst IP Addr:Port   Flags Tos  Packets    Bytes      pps      bps
2005-08-30 06:53:53.370   63.545 TCP   113.138.32.152:25    ->   222.33.70.124:3575  .AP.SF   0       62     3512        0      442
2005-08-30 06:53:53.370   63.545 TCP    222.33.70.124:3575  ->  113.138.32.152:25    .AP.SF   0       58     3300        0      415
Time window: Aug 30 2005 06:53:53 - Aug 30 2005 06:54:56
```

**Figura 3.10. Exemplo do resultado do comando nfdump com a opção da saída estendida ativada.**

Apesar dos exemplos ilustrados nesta Seção serem favoráveis para o monitoramento e análise de tráfego, existem situações onde o controle da porta espelhada do roteador ou do *switch* não é possível. Para superar esta limitação, algumas técnicas de redirecionamento de tráfego na camada 2 podem ser aplicadas como *arp spoofing*, ataques de inundação da tabela MAC e ataques de saltos de redes virtual VLAN (*vlan hopping*).

[13]http://nfdump.sourceforge.net

O ataque de *arp spoofing* consiste na substituição das informações armazenadas nas tabelas `arp` dos computadores da rede com os dados forjados pelo atacante através de mensagens de *broadcast* [Schiller and Binkley 2007]. Nesse ataque, o computador alvo recebe uma mensagem `arp` informando que o endereço IP do roteador (legítimo) mudou para um novo endereço (malicioso). Desta forma, os hosts que tiveram esse dado alterado, mandarão suas informações para o endereço IP do atacante, o qual será capaz de coletar o tráfego.

No caso de um *switch*, os atacantes se aproveitam do fato de que, para manter a comunicação ponto a ponto, este equipamento deve manter uma tabela que contenha todos os endereços `MAC` das máquinas e suas respectivas portas [Bhaiji 2009]. Assim, se um grande número de endereços `MAC` forem injetados em uma única porta, a tabela mantida pelo *switch* será inundada. Nessa situação, o *switch* não saberá qual endereço `MAC` pertence a vítima, permitindo que os hosts ligados a esse equipamento recebam todos os pacotes transmitidos na rede.

Finalmente, o ataque de saltos de `VLAN` acontece quando um atacante se conecta a uma `VLAN` para ter acesso ao tráfego de outras `VLANs` [Schiller and Binkley 2007]. Para este ataque funcionar, o usuário pode simular o comportamento de uma porta *trunk* em seu computador. Uma porta *trunk* corresponde é uma porta que é atribuída para transportar o tráfego de todas as `VLANs` que são acessíveis através de um *switch* específico. O protocolo DTP (*Dynamic Trunk Protocol*) permite a interligação e a configuração automática entre `VLANs` através da porta *trunk* [Cisco 2014]. Desta forma, o atacante consegue ter acesso a todas as `VLANs` da rede para coleta de tráfego.

### 3.4.2. Bibliotecas de Programação de Rede

As ferramentas ilustradas na Seção anterior apenas coletam o tráfego de rede, cabendo ao pesquisador, desenvolver soluções próprias para analisar e interpretar os resultados obtidos, tais como *parsers* e *scripts*. Para agilizar tal processo de investigação, pesquisadores podem utilizar bibliotecas de programação de rede para automatizar a leitura e análise do tráfego.

Entre as bibliotecas de programação de rede mais populares, encontra-se a `libpcap` [Jacobson et al. 2004]. Desenvolvida em 2003, a `libpcap` oferece uma interface de programação que permite capturar pacotes que passam através das interfaces de rede. Esta biblioteca pode ser instalada em sistemas baseados em `UNIX` e `Windows`. Ao iniciar uma sessão de captura, filtros semelhantes ao `TCPDump` podem ser especificados para capturar pacotes relevantes para análise.

De modo resumido, para coletar tráfego de rede a partir da `libpcap`, os seguintes passos podem ser seguidos:

- **Leitura do tráfego**: deve ser definido qual a fonte de coleta do tráfego. A `libpcap` oferece basicamente dois métodos: leitura em tempo real e leitura *off-line*. A leitura em tempo real coleta o tráfego através das interfaces de rede do computador, enquanto o último pode ler tráfego a partir de arquivos salvos no formato da `libpcap`.
- **Filtros**: são sequências de caracteres que seguem o formato *BSD Packet Filter*

[McCanne and Jacobson 1993]. Por exemplo, a sequência `tcp port 80` pode ser utilizada para filtrar o tráfego `HTTP` na porta 80.

- **Encapsulamento de dados**: alguns sistemas operacionais e formatos de arquivo rede organizam os dados da camada de enlace de maneira distinta. Por exemplo, o modelo `Ethernet` (IEEE 802.11) difere do modelo `RAW IP`, visto que o último não possui informações da camada de enlace de rede.
- **Leitura em camadas**: após a coleta de um pacote, é possível obter informações das camadas de rede individualmente. No entanto, cabe ao desenvolvedor implementar os métodos necessários para acessar tais dados.

A Listagem 3.5 apresenta um exemplo de código fonte na linguagem `C` capaz de monitorar o tráfego a partir de uma interface rede. Por questões de espaço, os trechos de códigos que validam retornos de funções e ponteiros estão omitidos. Os comandos apresentados nas linhas 21 a 26 são responsáveis pela captura do tráfego.

**Listagem 3.5. Código básico para coleta de pacotes usando a `libpcap`.**

```
1  #include <stdio.h>
2  #include <pcap.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>
5  #include <arpa/inet.h>
6
7  void dumpPkt(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);
8
9  int main(int argc, char **argv)
10 {
11         char *device = argv[1];              // interface de rede
12         char errBuf[PCAP_ERRBUF_SIZE];       // armazena erros
13         pcap_t *handler = NULL;              // tratador
14         struct pcap_pkthdr header;           // info. pacote capturado
15         struct bpf_program fp;               // filtro
16         char filter_rule[] = "tcp port 80"; // regras do filtro - apenas trafego http
17         bpf_u_int32 mask;                    // mascara ip da rede
18         bpf_u_int32 netip;                   // ip da rede
19         const u_char *pacote;                // pacote em si
20
21         handler = pcap_open_live(device, 65535, 1, 0, errBuf);
22         pcap_lookupnet(device, &netip, &mask, errBuf);
23         pcap_compile(handler, &fp, filter_rule, 0, netip);
24         pcap_setfilter(handler, &fp);
25         pcap_loop(handler, -1 , dumpPkt, NULL);
26         pcap_close(handler);
27         return 0;
28 }
```

Vale destacar a instrução descrita na linha 25, a qual apresenta o método de *callback* `dumpPtk` que sempre é executado quando um pacote de rede for lido pela função `pcap_loop`. O `dumpPtk` recebe um ponteiro (`*packet`) que pode ser utilizado para acessar às camadas do pacote capturado. Esta função de *callback* deve conter rotinas que façam o tratamento de dados por camadas tais como enlace, rede e transporte. Desta forma, o acesso à carga útil (*payload*) do protocolo `HTTP` pode ser obtido através das operações de ponteiros como `payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_tcp)`. No caso do tráfego `HTTP` , antes de extrair os dados para análise, é importante que o desenvolvedor reorganize os fluxos `TCP` conforme foram enviados na rede.

Embora a `libpcap` seja utilizada por diversas aplicações e seja portada como extensão para outras linguagens como `Perl`, `Ruby`, `Python` e `Java`, o acesso a carga útil do pacote não é trivial. Nesse sentido, algumas soluções mais recentes buscam agilizar este acesso como a `libtrace` e `scapy`.

A biblioteca `libtrace`, desenvolvida em [Alcock et al. 2012], possui inúmeras vantagens em relação à `libpcap`, incluindo capacidade de leitura e escrita de arquivos compactados, menor consumo de memória e acesso direto às camadas de rede através de métodos prontos. O acesso nativo de leitura aos formatos compactados permite que análise do tráfego de rede seja mais rápida, através de paralelismo computacional. Para extrair as informações do cabeçalho `TCP`, o desenvolvedor pode utilizar a seguinte função `trace_get_tcp(pacote)`, a qual retorna um ponteiro para tal posição do cabeçalho. No entanto, mesmo oferecendo algumas facilidades na manipulação de camadas, ainda é necessário que o desenvolvedor implemente suas rotinas para tratar a carga útil do pacote.

A `scapy` tem como objetivo abordar os problemas demonstrados acima de maneira mais simples [Biondi 2010]. Desenvolvida em `Python`, a biblioteca permite ao usuário enviar, coletar, dissecar e forjar pacotes de rede. No cenário de coleta de tráfego, o desenvolvedor pode definir uma função de *callback*, a qual funciona de maneira semelhante a `libtrace`.

Para ilustrar o processo de coleta usando a `scapy`, a Listagem 3.6 apresenta um código que captura o tráfego `TCP` destinado à porta 80. A função `http_callback` (linhas 4-16) recebe os pacotes fornecidos pela função `sniff` (linha 18) e é capaz de imprimir o endereço IP de destino do servidor `HTTP` (linha 13) e o cabeçalho da solicitação (linha 14).

**Listagem 3.6. Exemplo de *sniffer* `HTTP` usando a biblioteca `scapy`.**

```
1  #!/usr/bin/env python
2  from scapy.all import *
3
4  def http_callback(pkt):
5          httpHead = False
6          try:
7                  httpHead = pkt[Raw].load
8
9                  if httpHead:
10                         dstIP = pkt[IP].dst
11
12                         print "-----------------"
13                         print dstIP
14                         print httpHead
15         except:
16                 pass
17
18 sniff(prn=http_callback, filter="tcp dst port 80", store=0)
```

Essa Seção demonstrou como as ferramentas e bibliotecas de programação de rede podem ser utilizadas no processo de coleta de tráfego. A seguir, será demonstrado um estudo de caso de detecção de tráfego malicioso.

### 3.4.3. Estudo de Caso

Esta Seção apresenta um estudo de caso de comportamentos suspeitos na rede. Para investigação, um ambiente virtualizado composto por quatro com o sistema operacional `Debian GNU/Linux` na versão estável. A arquitetura possui um servidor e três clientes, onde cada máquina virtual possui 1GB de memória RAM e softwares de linguagem de programação como `C`, `Perl` e `Python`. Além disso, softwares extras foram adicionados no servidor para auxiliar na análise de tráfego incluindo os descritos na seção 3.4.1 e ferramentas como `RRDTool`[14], `MRTG`[15] e `NFSen`[16].

Inicialmente, o servidor é habilitado com o software `nfdump` e `MRTG`, o qual será utilizado para capturar fluxos de tráfego. Em seguida, os clientes iniciarão um ataque de negação de serviço a partir da ferramenta `T50`[17]. O objetivo é lançar ataques do tipo `SYN` contra o serviço web do servidor e coletar fluxos de dados para análise posterior.

Cada máquina executou, em dois momento distintos, o comando `t50` fornecendo os argumentos para estabelecimento de conexão (`-S`) na porta de destino 80 (`--dport 80`) do endereço IP do servidor `10.208.8.81`. Além disso, foram adicionados os comandos para inundação (`--flood`), que continua enviando pacotes à vítima até o cancelamento do `t50` e a opção `--turbo`, que aumenta o número de `threads` durante o ataque. É possível observar dois picos de ingresso de tráfego na Figura 3.11 durante o ataque. Apesar de gerar certo volume e deixar o acesso mais lento, tal ataque não interrompeu o serviço web da vítima. No entanto, em casos reais, como a `botnet Torpig`, o volume gerado por milhares de máquinas pode interromper a operação de grandes provedores de Internet [Stone-Gross et al. 2009].

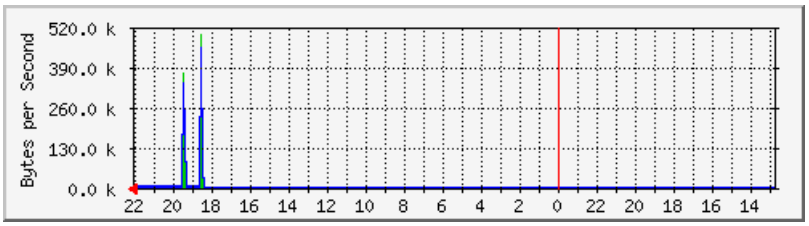



**Figura 3.11. Exemplo de tráfego coletado durante um ataque de negação de serviço do tipo `SYN`.**

Outra maneira de observar o ataque descrito na Figura 3.11 é a utilização dos métodos estatísticos que o `tshark` oferece. A Listagem 3.7 apresenta um resumo comparativo entre comportamentos de tráfego legítimo e `DDoS`. As linhas 6 e 18 representam o filtro de pacotes do tipo `SYN`, enquanto as linhas 7 e 19 apenas as consultas do tipo `ACK`, ou seja, a confirmação que o pacote foi recebido.

O comportamento do tráfego legítimo foi obtido através de consultas randômicas, utilizando o comando `wget`. É possível observar que existe uma proporção entre as consultas (*frames*) do tipo `SYN` (*Column #0*) e do tipo `ACK` (*Column #1*) nesse tráfego. Por outro lado, a quantidade de consultas do tipo `SYN` no tráfego `DDoS` é desproporcional

[14] http://oss.oetiker.ch/rrdtool/
[15] http://oss.oetiker.ch/mrtg/index.en.html
[16] http://nfsen.sourceforge.net/
[17] http://sourceforge.net/projects/t50/

em relação às consultas do tipo ACK. Vale ressaltar que em ataques de DDoS do tipo SYN, o endereço IP de origem é forjado. Portanto, a vítima desse ataque não recebe a confirmação, pois o endereço IP verdadeiro não enviou o pacote [Eddy 2007].

Para interromper os ataques de negação de serviço, existem algumas abordagens que podem ser utilizadas como filtragem, redução do temporizador que sinaliza um pacote SYN recebido, o emprego de cache de pacotes SYN, filtragem por estado de pacote e proteção através de proxy na rede. A descrição de tais técnicas podem ser observadas na RFC 4987 [Eddy 2007].

**Listagem 3.7. Comparação entre comportamento de acesso legítimo e um DDoS**

```
1
2  # Tráfego Legítimo
3  ===================================================================
4  IO Statistics
5  Interval: 60.000 secs
6  Column #0: tcp.flags.syn==1 && tcp.flags.ack==0
7  Column #1: tcp.flags == 0x0010
8                  |   Column #0    |   Column #1
9  Time            |frames|  bytes  |frames|  bytes
10 000.000-060.000      23      1702     92      6072
11 060.000-120.000      30      2220    120      7920
12 120.000-180.000      35      2590    137      9042
13
14 # Tráfego Malicioso (DDoS)
15 ===================================================================
16 IO Statistics
17 Interval: 60.000 secs
18 Column #0: tcp.flags.syn==1 && tcp.flags.ack==0
19 Column #1: tcp.flags == 0x0010
20                 |   Column #0    |   Column #1
21 Time            |frames|  bytes  |frames|  bytes
22 000.000-060.000   27422   1645320      0         0
23 060.000-120.000    8504    510240      0         0
24 120.000-180.000    1746    104802      6       396
```

## 3.5. Considerações Finais

Diferente dos outros tipo de programas maliciosos, cujo único objetivo é inutilizar os sistemas computacionais afetados, as *botnets* fazem parte de uma categoria muito mais complexa, onde o objetivo é, na maioria dos casos, atingir algum ganho financeiro. E seus desenvolvedores e mantenedores tem se esforçado muito para manter suas criações indetectáveis, garantindo a continuidade de suas atividades maliciosas.

Em suas primeiras encarnações, o ganho financeiro derivado das atividades de *botnets* era restrito à interceptação de dados, como números de cartões de crédito e informações bancárias, que poderiam ser utilizados em operações comerciais fraudulentas. Deste lá, este quadro evoluiu para o momento a partir do qual os proprietários das *botnets* passaram a vender ou alugar seus serviços, o que criou todo um mercado negro de recursos computacionais sequestrados, que são continuamente empregados como base de ações maliciosas. Seus ataques agora não são mais indiscriminados e aleatórios. Seus alvos são bem definidos e tem um preço que, infelizmente, alguém está disposto a pagar.

Assim, para proteger o seu "modo de vida", os desenvolvedores de *botnets* não têm medido esforços para ludibriar todas as tentativas de interromper suas atividades. Técnicas de polimorfismo e encriptação de mensagens, para ludibriar sistemas baseados em assinaturas; uso de protocolos de comunicação considerados benignos, para enganar os tradicionais filtros de pacotes; e a dissimulação de suas atividades, para impedir a aparição de anomalias que dariam pistas de sua presença dentro dos sistemas contaminados, são alguns dos exemplos. Isoladas ou em conjunto, todas estas técnicas tem sido amplamente empregadas para evitar cada novo processo de detecção que foram sendo desenvolvidos durante os últimos anos. Além disso, a inclusão dos dispositivos móveis na infraestrutura das *botnets* cria toda uma nova gama de desafios, exigindo ainda mais criatividade dos pesquisadores dedicados ao combate da ameaça que as *botnets* representam.

Dada a complexidade deste problema, dificilmente será encontrada uma solução única que seja capaz de lidar com todas as técnicas empregadas para proteger as ações das *botnets*. Entretanto, a combinação de soluções de detecção e a pesquisa e o desenvolvimento de novas ferramentas de combate a esta ameaça promete não dar trégua nesta contínua batalha contra estes programas maliciosos e seus desenvolvedores.

Este Capítulo apresentou informações relevantes para o entendimento do problema das *botnets*, incluindo um conjunto de definições básicas, a evolução histórica, o ciclo de vida, as principais arquiteturas utilizadas na construção, os principais tipos de ataques suportados, as abordagens básicas. Também apresentou uma visão geral do estado da arte sobre a detecção de *botnets*, bem como um apanhado de ferramentas comumente empregadas para coletar os indícios da presença de uma *botnet* em sistemas computacionais. Os autores esperam que este conteúdo sirva de ponto de partida para futuros trabalhos e possa despertar o interesse de pesquisadores para esta área.

## Referências

[Abu Rajab et al. 2006] Abu Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A. (2006). A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 41–52, New York, NY, USA. ACM.

[Alata et al. 2007] Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., and Herrb, M. (2007). Lessons learned from the deployment of a high-interaction honeypot. *CoRR*, abs/0704.0858.

[Alcock et al. 2012] Alcock, S., Lorier, P., and Nelson, R. (2012). Libtrace: a packet capture and analysis library. *SIGCOMM Comput. Commun. Rev.*, 42(2):42–48.

[Antonakakis et al. 2010] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., and Feamster, N. (2010). Building a dynamic reputation system for dns. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 18–18, Berkeley, CA, USA. USENIX Association.

[Barbosa and Souto 2009] Barbosa, K. R. S. and Souto, E. (2009). Análise passiva do tráfego dns da internet brasileira. In *IX Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg 2009)*, pages 203–216, Campinas.

[Barr 1996] Barr, D. (1996). RFC 1912: Common DNS operational and configuration errors. http://www.ietf.org/rfc/rfc1912.txt.

[Bazrafshan et al. 2013] Bazrafshan, Z., Hashemi, H., Fard, S., and Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 113–120.

[Berghel 2001] Berghel, H. (2001). The code red worm. *Commun. ACM*, 44(12):15–19.

[Bhaiji 2009] Bhaiji, Y. (2009). Understanding, preventing, and defending against layer 2 attacks.

[Binsalleeh et al. 2010] Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., and Wang, L. (2010). On the analysis of the zeus botnet crimeware toolkit. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 31–38.

[Biondi 2010] Biondi, P. (2010). Welcome to scapyś documentation! Acessado em 12/09/2013.

[BITAG 2012] BITAG (2012). Snmp reflected amplification ddos attack mitigation. Technical report, Broadband Internet Technical Advisory Group.

[Boshmaf et al. 2011] Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2011). The socialbot network: When bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 93–102, New York, NY, USA. ACM.

[Cai and Zou 2012] Cai, T. and Zou, F. (2012). Detecting http botnet with clustering network traffic. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–7.

[Callegati et al. 2009] Callegati, F., Cerroni, W., and Ramilli, M. (2009). Man-in-the-middle attack to the https protocol. *Security Privacy, IEEE*, 7(1):78–81.

[Calvet et al. 2010] Calvet, J., Davis, C. R., Fernandez, J. M., Marion, J.-Y., St-Onge, P.-L., Guizani, W., Bureau, P.-M., and Somayaji, A. (2010). The case for in-the-lab botnet experimentation: Creating and taking down a 3000-node botnet. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 141–150, New York, NY, USA. ACM.

[Canavan 2005] Canavan, J. (2005). The evolution of malicious irc bots.

[Carpine et al. 2013] Carpine, F., Mazzariello, C., and Sansone, C. (2013). Online irc botnet detection using a soinn classifier. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 1351–1356.

[Chawathe et al. 2003] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. (2003). Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 407–418, New York, NY, USA. ACM.

[Choi and Lee 2012] Choi, H. and Lee, H. (2012). Identifying botnets by capturing group activities in dns traffic. *Computer Networks*, 56(1):20–33.

[Choi et al. 2009] Choi, H., Lee, H., and Kim, H. (2009). Botgad: Detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 2:1–2:8, New York, NY, USA. ACM.

[CISCO 2013] CISCO (2013). Cisco annual security report. Technical report.

[CISCO 2014] CISCO (2014). Cisco annual security report. Technical report.

[Cisco 2014] Cisco (2014). Layer 2 lan port configuration. Acesssado em 10/09/2014.

[Claise 2004] Claise, B. (2004). RFC 3954 - Cisco Systems NetFlow Services Export Version 9.

[Cohen 1987] Cohen, F. (1987). Computer viruses: Theory and experiments. *Computers & Security*, 6(1):22 – 35.

[Colajanni et al. 2008] Colajanni, M., Gozzi, D., and Marchetti, M. (2008). Collaborative architecture for malware detection and analysis. In Jajodia, S., Samarati, P., and Cimato, S., editors, *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, volume 278 of *IFIP - The International Federation for Information Processing*, pages 79–93. Springer US.

[Dagon et al. 2007] Dagon, D., Gu, G., Lee, C., and Lee, W. (2007). A taxonomy of botnet structures. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 325 –339.

[Daswani and Stoppelman 2007] Daswani, N. and Stoppelman, M. (2007). The anatomy of clickbot.a. In *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 11–11, Berkeley, CA, USA. USENIX Association.

[Davis et al. 2008] Davis, C., Fernandez, J., Neville, S., and McHugh, J. (2008). Sybil attacks as a mitigation strategy against the storm botnet. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 32–40.

[Dietrich et al. 2011] Dietrich, C., Rossow, C., Freiling, F., Bos, H., van Steen, M., and Pohlmann, N. (2011). On botnets that use dns for command and control. In *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, pages 9 –16.

[Douceur 2002] Douceur, J. (2002). The sybil attack. In Druschel, P., Kaashoek, F., and Rowstron, A., editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg.

[Dunham 2009] Dunham, K. (2009). *Mobile Malware Attacks and Defense*. Syngress Publishing.

[Eddy 2007] Eddy, W. (2007). TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (Informational).

[Egele et al. 2008] Egele, M., Scholte, T., Kirda, E., and Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, 44(2):6:1–6:42.

[Eslahi et al. 2013] Eslahi, M., Hashim, H., and Tahir, N. (2013). An efficient false alarm reduction approach in http-based botnet detection. In *Computers Informatics (ISCI), 2013 IEEE Symposium on*, pages 201–205.

[Feily et al. 2009] Feily, M., Shahrestani, A., and Ramadass, S. (2009). A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, pages 268 –273.

[Ferrer et al. 2010] Ferrer, Z., Ferrer, M. C., and Researchers, C. I. S. (2010). In-depth analysis of hydraq. *The face of cyberwar enemies unfolds. ca isbu-isi white paper*, 37.

[Fossi et al. 2011] Fossi, M., Egan, G., Haley, K., Turner, D., Johnson, E., Johnson, E., and Adams, T. (2011). Symantec global internet security threat report. trends for 2010. Technical Report 17, Symantec.

[Fossi et al. 2010] Fossi, M., Turner, D., Johnson, E., Johnson, E., and Adams, T. (2010). Symantec global internet security threat report. Technical Report 17, Symantec.

[Freiling et al. 2005] Freiling, F., Holz, T., and Wicherski, G. (2005). Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In Vimercati, S., Syverson, P., and Gollmann, D., editors, *Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 319–335. Springer Berlin Heidelberg.

[Freitas et al. 2014] Freitas, C., Benevenuto, F., and Veloso, A. (2014). Socialbots: Implicações na segurança e na credibilidade de serviços baseados no twitter. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos- SBRC 2014*, pages 603–616.

[Furnell and Ward 2004] Furnell, S. and Ward, J. (2004). Malware evolution: Malware comes of age: The arrival of the true computer parasite. *Netw. Secur.*, 2004(10):11–15.

[Grizzard et al. 2007] Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., and Dagon, D. (2007). Peer-to-peer botnets: Overview and case study. In *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 1–1, Berkeley, CA, USA. USENIX Association.

[Gu et al. 2008a] Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008a). Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 139–154, Berkeley, CA, USA. USENIX Association.

[Gu et al. 2008b] Gu, G., Zhang, J., and Lee, W. (2008b). Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium.*

[Guo et al. 2006] Guo, F., Chen, J., and cker Chiueh, T. (2006). Spoof detection for preventing dos attacks against dns servers. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 37–37.

[Hachem et al. 2011] Hachem, N., Ben Mustapha, Y., Granadillo, G., and Debar, H. (2011). Botnets: Lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8.

[Haddadi et al. 2014] Haddadi, F., Morgan, J., Filho, E., and Zincir-Heywood, A. (2014). Botnet behaviour analysis using ip flows: With http filters using classifiers. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 7–12.

[Holz et al. 2008a] Holz, T., Gorecki, C., Rieck, K., and Freiling, F. C. (2008a). Measuring and detecting fast-flux service networks. In *NDSS*.

[Holz et al. 2008b] Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. (2008b). Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08, pages 9:1–9:9, Berkeley, CA, USA. USENIX Association.

[Hua and Sakurai 2013] Hua, J. and Sakurai, K. (2013). Botnet command and control based on short message service and human mobility. *Computer Networks*, 57(2):579 – 597. Botnet Activity: Analysis, Detection and Shutdown.

[Huang et al. 2010] Huang, S.-Y., Mao, C.-H., and Lee, H.-M. (2010). Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 101–111, New York, NY, USA. ACM.

[Husna et al. 2008] Husna, H., Phithakkitnukoon, S., and Dantu, R. (2008). Traffic shaping of spam botnets. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 786–787.

[Ishibashi et al. 2005] Ishibashi, K., Toyono, T., Toyama, K., Ishino, M., Ohshima, H., and Mizukoshi, I. (2005). Detecting mass-mailing worm infected hosts by mining dns traffic data. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, MineNet '05, pages 159–164, New York, NY, USA. ACM.

[Jacobson et al. 2004] Jacobson, V., Leres, C., and McCanne, S. (2004). pcap - packet capture library.
urlhttp://www.tcpdump.org/pcap3_man.html (acessado em 01/08/2014).

[Jakobsson and Ramzan 2008] Jakobsson, M. and Ramzan, Z. (2008). *Crimeware: Understanding New Attacks and Defenses (Symantec Press)*. Addison-Wesley Professional, 1 edition.

[John et al. 2009] John, J. P., Moshchuk, A., Gribble, S. D., and Krishnamurthy, A. (2009). Studying spamming botnets using botlab. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 291–306, Berkeley, CA, USA. USENIX Association.

[Kamienski et al. 2005] Kamienski, C., Souto, E., Rocha, J., Domingues, M., Callado, A., and Sadok, D. (2005). Colaboração na internet ea tecnologia peer-to-peer. In *XXV Congresso da Sociedade Brasileira de Computação*, pages 1407–1454.

[Kang and Zhang 2009] Kang, J. and Zhang, J.-Y. (2009). Application entropy theory to detect new peer-to-peer botnet with multi-chart cusum. In *Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on*, volume 1, pages 470–474.

[Kumar and Selvakumar 2011] Kumar, P. A. R. and Selvakumar, S. (2011). Distributed denial of service attack detection using an ensemble of neural classifier. *Computer Communications*, 34(11):1328 – 1341.

[Lee et al. 2008] Lee, J.-S., Jeong, H., Park, J.-H., Kim, M., and Noh, B.-N. (2008). The activity analysis of malicious http-based botnets using degree of periodic repeatability. In *Security Technology, 2008. SECTECH '08. International Conference on*, pages 83–86.

[Levy 2003] Levy, E. (2003). The making of a spam zombie army. dissecting the sobig worms. *Security Privacy, IEEE*, 1(4):58–59.

[Li et al. 2009a] Li, C., Jiang, W., and Zou, X. (2009a). Botnet: Survey and case study. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 1184–1187.

[Li et al. 2009b] Li, Z., Liao, Q., and Striegel, A. (2009b). Botnet economics: Uncertainty matters. In Johnson, M., editor, *Managing Information Risk and the Economics of Security*, pages 245–267. Springer US.

[Lin et al. 2009] Lin, H.-C., Chen, C.-M., and Tzeng, J.-Y. (2009). Flow based botnet detection. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 1538–1541.

[Liu et al. 2009] Liu, J., Xiao, Y., Ghaboosi, K., Deng, H., and Zhang, J. (2009). Botnet: Classification, attacks, detection, tracing, and preventive measures. In *Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control*, ICICIC '09, pages 1184–1187, Washington, DC, USA. IEEE Computer Society.

[Livadas et al. 2006] Livadas, C., Walsh, R., Lapsley, D., and Strayer, W. (2006). Using machine learning technliques to identify botnet traffic. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 967–974.

[Lu and Ghorbani 2008] Lu, W. and Ghorbani, A. A. (2008). Botnets detection based on irc-community. In *Proceedings of the Global Communications Conference, 2008. GLOBECOM 2008, New Orleans, LA, USA, 30 November - 4 December 2008*, pages 2067–2071.

[Ma et al. 2010] Ma, X., Guan, X., Tao, J., Zheng, Q., Guo, Y., Liu, L., and Zhao, S. (2010). A novel irc botnet detection method based on packet size sequence. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5.

[Mazzariello 2008] Mazzariello, C. (2008). Irc traffic analysis for botnet detection. In *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on*, pages 318–323.

[McCanne and Jacobson 1993] McCanne, S. and Jacobson, V. (1993). The bsd packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, USENIX'93, pages 2–2, Berkeley, CA, USA. USENIX Association.

[McCarty 2003a] McCarty, B. (2003a). Automated identity theft. *Security Privacy, IEEE*, 1(5):89–92.

[McCarty 2003b] McCarty, B. (2003b). Botnets: Big and bigger. *IEEE Security and Privacy*, 1(4):87–90. cited By (since 1996)41.

[McFee 2003] McFee (2003). Virus profile: W32/sdbot.worm. Acessado em 12/09/2014.

[Mockapetris 1987] Mockapetris, P. (1987). RFC 1034: Domain names - concepts and facilities. http://www.ietf.org/rfc/rfc1034.txt.

[Moore et al. 2002] Moore, D., Shannon, C., and claffy, k. (2002). Code-red: A case study on the spread and victims of an internet worm. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 273–284, New York, NY, USA. ACM.

[Morales et al. 2009] Morales, J., Al-Bataineh, A., Xu, S., and Sandhu, R. (2009). Analyzing dns activities of bot processes. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 98–103.

[Mulliner and Seifert 2010] Mulliner, C. and Seifert, J.-P. (2010). Rise of the ibots: Owning a telco network. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 71–80.

[Musashi et al. 2011] Musashi, Y., Kumagai, M., Kubota, S., and Sugitani, K. (2011). Detection of kaminsky dns cache poisoning attack. In *Intelligent Networks and Intelligent Systems (ICINIS), 2011 4th International Conference on*, pages 121–124.

[Nagaraja et al. 2010] Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., and Borisov, N. (2010). Botgrep: finding p2p bots with structured graph analysis. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 7–7, Berkeley, CA, USA. USENIX Association.

[Nappa et al. 2010] Nappa, A., Fattori, A., Balduzzi, M., DellAmico, M., and Cavallaro, L. (2010). Take a deep breath: A stealthy, resilient and cost-effective botnet using skype. In Kreibich, C. and Jahnke, M., editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 6201 of *Lecture Notes in Computer Science*, pages 81–100. Springer Berlin Heidelberg.

[Nazario and Holz 2008] Nazario, J. and Holz, T. (2008). As the net churns: Fast-flux botnet observations. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 24–31.

[Oikarinen and Reed 1993] Oikarinen, J. and Reed, D. (1993). Internet relay chat protocol.

[Passerini et al. 2008] Passerini, E., Paleari, R., Martignoni, L., and Bruschi, D. (2008). Fluxor: Detecting and monitoring fast-flux service networks. In Zamboni, D., editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5137 of *Lecture Notes in Computer Science*, pages 186–206. Springer Berlin Heidelberg.

[Peng et al. 2007] Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1).

[Perdisci et al. 2009] Perdisci, R., Corona, I., Dagon, D., and Lee, W. (2009). Detecting malicious flux service networks through passive analysis of recursive dns traces. *Computer Security Applications Conference, Annual*, 0:311–320.

[Perdisci et al. 2010] Perdisci, R., Lee, W., and Feamster, N. (2010). Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 26–26, Berkeley, CA, USA. USENIX Association.

[ProofPoint 2014] ProofPoint (2014). Proofpoint uncovers internet of things (iot) cyberattack. Acessado: 10/04/2014.

[Rebane 2011] Rebane, J. C. (2011). *The Stuxnet Computer Worm and Industrial Control System Security*. Nova Science Publishers, Inc., Commack, NY, USA.

[Rodionov and Matrosov 2011] Rodionov, E. and Matrosov, A. (2011). The evolution of tdl: Conquering x64. *ESET, June*.

[Rodríguez-Gómez et al. 2013] Rodríguez-Gómez, R. A., Maciá-Fernández, G., and García-Teodoro, P. (2013). Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.*, 45(4):45:1–45:33.

[Royal 2008] Royal, P. (2008). Analysis of the kraken botnet. *Damballa*, 9.

[Salusky and Danford 2008] Salusky, W. and Danford, R. (2008). Know your enemy: Fast-flux service networks. an ever changing enemy. *The Honeynet Project*.

[Schiller and Binkley 2007] Schiller, C. and Binkley, J. (2007). *Botnets: The Killer Web Applications*. Syngress Publishing.

[Shen and Hasegawa 2010] Shen, F. and Hasegawa, O. (2010). Self-organizing incremental neural network and its application. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 535–540, Berlin, Heidelberg. Springer-Verlag.

[Shin and Gu 2010] Shin, S. and Gu, G. (2010). Conficker and beyond: A large-scale empirical study. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 151–160, New York, NY, USA. ACM.

[Shin et al. 2011] Shin, S., Lin, R., and Gu, G. (2011). Cross-analysis of botnet victims: New insights and implications. In Sommer, R., Balzarotti, D., and Maier, G., editors, *Recent Advances in Intrusion Detection*, volume 6961 of *Lecture Notes in Computer Science*, pages 242–261. Springer Berlin Heidelberg.

[Shoch and Hupp 1982] Shoch, J. F. and Hupp, J. A. (1982). The “worm” programs—early experience with a distributed computation. *Commun. ACM*, 25(3):172–180.

[Sinha et al. 2010] Sinha, P., Boukhtouta, A., Belarde, V., and Debbabi, M. (2010). Insights from the analysis of the mariposa botnet. In *Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*, pages 1–9.

[Souza et al. 2013] Souza, A., Barbosa, K. R. S., and Feitosa, E. (2013). Identificando spam no twitter através da anáise empírica dos trending topics brasil. In *SBSeg 2013 - WTICG (2013)*, pages 394–403.

[Stoica et al. 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA. ACM.

[Stone-Gross et al. 2009] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., and Vigna, G. (2009). Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 635–647, New York, NY, USA. ACM.

[Strayer et al. 2008] Strayer, W., Lapsely, D., Walsh, R., and Livadas, C. (2008). Botnet detection based on network behavior. In Lee, W., Wang, C., and Dagon, D., editors, *Botnet Detection*, volume 36 of *Advances in Information Security*, pages 1–24. Springer US.

[Studer 2011] Studer, R. (2011). Economic and technical analysis of botnets and denial-of-service attacks. *Communication systems IV*, page 19.

[Tenebro 2008] Tenebro, G. (2008). W32.waledac - threat analysis. Technical report, Symantec.

[Thonnard and Dacier 2011] Thonnard, O. and Dacier, M. (2011). A strategic analysis of spam botnets operations. In *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 162–171, New York, NY, USA. ACM.

[Tiirmaa-Klaar et al. 2013] Tiirmaa-Klaar, H., Gassen, J., Gerhards-Padilla, E., and Martini, P. (2013). Botnets: How to fight the ever-growing threat on a technical level. In *Botnets*, SpringerBriefs in Cybersecurity, pages 41–97. Springer London.

[Turner et al. 2007] Turner, D., Entwisle, S., Denesiuk, M., Fossi, M., Blackbird, J., and McKinney, D. (2007). Internet security threat report. 2007 trends. Technical Report 17, Symantec.

[Tyagi and G.Aghila 2011] Tyagi, A. K. and G.Aghila (2011). A wide scale survey on botnet. *International Journal of Computer Applications*, 34(9):10–23. Published by Foundation of Computer Science, New York, USA.

[Virvilis et al. 2013] Virvilis, N., Gritzalis, D., and Apostolopoulos, T. (2013). Trusted computing vs. advanced persistent threats: Can a defender win this game? In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 396–403.

[Wang et al. 2009] Wang, P., Wu, L., Aslam, B., and Zou, C. (2009). A systematic study on peer-to-peer botnets. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–8.

[Xiang et al. 2011] Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., and Tianning, Z. (2011). Andbot: towards advanced mobile botnets. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, LEET'11, pages 11–11, Berkeley, CA, USA. USENIX Association.

[Yadav et al. 2012] Yadav, S., Reddy, A. K. K., Reddy, A. L. N., and Ranjan, S. (2012). Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/ACM Trans. Netw.*, 20(5):1663–1677.

[Yen and Reiter 2010] Yen, T.-F. and Reiter, M. (2010). Are your hosts trading or plotting? telling p2p file-sharing and bots apart. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 241–252.

[Zeidanloo and Manaf 2010] Zeidanloo, H. R. and Manaf, A. B. A. (2010). Botnet detection by monitoring similar communication patterns. In *International Journal of Computer Science and Information Security (IJCSIS)*, volume Vol. 7.

[Zhao et al. 2013] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., and Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39, Part A(0):2 – 16. 27th {IFIP} International Information Security Conference.

# Capítulo 4

# Segurança em Redes Veiculares: Inovações e Direções Futuras

**Michelle S. Wangham**[◇], **Michele Nogueira**[†], **Cláudio P. Fernandes**[◇], **Osmarildo Paviani**[◇], **Benevid F. da Silva**[†]

[◇] Universidade do Vale do Itajaí (UNIVALI)

[†] Universidade Federal do Paraná (UFPR)

***Abstract***

*Over the past years, many research efforts have focused on enhancing vehicular networks (VANETs - Vehicular Adhoc Networks), as they are considered the foremost technology to provide safety and convenience to drivers as well as to support new applications to passengers' entertainment. Despite the benefits offered by VANETs, these networks introduce security challenges, such as the difficulty in keeping data privacy and the effortlessness in generating malicious behaviors, which although they have been gradually addressed in the literature, they result in yet open issues for research and innovation. In this chapter, we revisit* $(i)$ *the main concepts of vehicular networks,* $(ii)$ *the major vulnerabilities and attacks against these networks, and* $(iii)$ *the most prominent existing countermeasures. The open issues related to the existing countermeasures and the impacts of using those countermeasures are critically discussed, pointing out future directions for research and innovation.*

***Resumo***

*Ao longo dos últimos anos, muitos esforços de pesquisa têm se concentrado no avanço das redes veiculares (VANETs - Vehicular Adhoc Networks), consideradas hoje uma das principais tecnologias para fornecer segurança e conveniência aos motoristas, bem como possibilitar novas aplicações voltadas ao entretenimento dos passageiros. Apesar dos vários benefícios oferecidos pelas redes veiculares, estas introduzem desafios de segurança, tais como a dificuldade em garantir a privacidade dos dados e a facilidade em gerar comportamentos maliciosos na rede, os quais, ainda geram questões abertas em termos de pesquisa e inovação. Neste capítulo, são revisitados* $(i)$ *os principais conceitos de VANETS,* $(ii)$ *as principais vulnerabilidades e ataques de segurança, e* $(iii)$ *as contramedidas mais proeminentes. As questões em aberto relacionadas às contramedidas de segurança e os impactos decorrentes do uso destas são discutidos de forma crítica, apontando direções futuras para pesquisa e inovação.*

## 4.1. Introdução

Os noticiários mostram diariamente acidentes e congestionamentos que poderiam ser evitados com a utilização de Sistemas de Transportes Inteligentes (ITS- *Intelligent Transportation Systems*). Estes sistemas se apoiam nos avanços tecnológicos da comunicação sem fio e da computação móvel e na criação de aplicações que facilitem o cotidiano de motoristas e pedestres, assim como a gestão de trânsito e tráfego urbano realizada por empresas e pelo governo, visando o desenvolvimento de cidades inteligentes [Avelar et al. 2014]. Por exemplo, o uso de técnicas e equipamentos específicos que possibilitem a comunicação entre veículos [Faezipour et al. 2012]. Tendo em vista o potencial de revolucionar a experiência ao dirigir e a segurança, a comunicação entre veículos (V2V - *Vehicle-to-Vehicle*) está se tornando cada vez mais popular e tem atraído a atenção da indústria automobilística e da academia, além de permitir a existência de redes veiculares [Karagiannis et al. 2011].

A idéia básica das VANETs (*Vehicular Adhoc Networks*) é fazer uso, com alguns ajustes, das tecnologias amplamente adotadas e baratas das redes sem fio e das redes de sensores, e instalá-las em veículos. Os veículos são então capazes de coletar, gerar e analisar uma grande quantidade de dados, porém, para de fato proverem assistência a passageiros e motoristas, esses dados precisam ser compartilhados. Nas VANETs, o compartilhamento desses dados pode ser realizado de forma colaborativa diretamente entre os veículos em "um salto" de comunicação ou ainda através de "múltiplos saltos" de comunicação (V2V), em que veículos intermediários auxiliam na comunicação repassando as mensagens a outros veículos dentro da área de cobertura do sinal de comunicação sem fio [Hussain et al. 2012]. A comunicação através de múltiplos saltos possui limitações de alcance de propagação dos dados e está sujeita a intempéries. Desta forma, a fim de aumentar a cobertura da comunicação, pontos fixos na estrada (unidades de acostamento ou *Road Side Unit* - RSUs) podem também ser implantados, suportando comunicações V2I (*Vehicle-to-Infraestructure*), aquelas entre o veículo e uma RSU.

As aplicações e os serviços emergentes em redes veiculares exigem grandes mudanças nos modelos de computação e de comunicação, devido às suas características próprias como alta dinamicidade dos veículos, densidade variável, desconexões frequentes e outras. Tais aspectos motivam o desenvolvimento de modelos específicos para as redes veiculares [Lee et al. 2014]. Trabalhos recentes descrevem diversas possibilidades para combinação das redes veiculares com o modelo de computação em nuvem (V2C - *Vehicle-to-cloud*), com as redes sociais e com os modelos de redes da Internet do Futuro, tais como Redes Orientadas a Conteúdos e Redes Tolerantes a Atrasos e Desconexões [Gerla e Kleinrock 2011, Karagiannis et al. 2011, Hussain et al. 2012, Wang et al. 2012, Lee et al. 2014, Grassi et al. 2014, Mezghani et al. 2014a]. Além disso, as aplicações de tecnologias de comunicação mais avançadas, tais como rádio cognitivo e *Long-Term Evolution* (LTE), são cada vez mais utilizadas para aperfeiçoar o desempenho das comunicações sem fio em VANETs e compartilhar os dados de forma mais eficiente e robusta [Silva et al. 2014a].

A segurança em redes veiculares é um fator imprescindível que precisa ser observado, pois a falta desta pode afetar a vida de pessoas. Como quaisquer redes de computadores sem fio e redes *ad hoc* estas estão suscetíveis a ataques por usuários ou nós maliciosos, tais como ataques de negação de serviço, modificação de mensagens e

análise de tráfego [Raya et al. 2006a, Nogueira et al. 2009, Hartenstein e Laberteaux 2010]. Porém, as características das VANETs, tais como a alta mobilidade dos nós, desconexões frequentes, densidade variável e escala da rede, trazem novos desafios à segurança. Como as VANETs suportam aplicações de emergência em tempo real e lidam com informações críticas de segurança no trânsito, estas devem satisfazer os seguintes requisitos de segurança: confidencialidade, integridade, disponibilidade, autenticidade, privacidade e não repudiação para prover segurança na comunicação dos dados [Samara et al. 2010]. Os desafios de segurança aumentam quando as redes veiculares são combinadas com outras tecnologias que compõem os plataformas em nuvem e orientadas a conteúdos. As aplicações nesses novos e promissores ambientes encorajam o compartilhamento de recursos e informações. Entretanto, essas plataformas também passam a ser alvo de ataques, como por exemplo, de negação de serviço ou de injeção de aplicativos maliciosos [Lee et al. 2014], sendo a prevenção desses mais complexa.

Muitas pesquisas estão sendo realizadas para garantir a segurança em VANETs. De acordo com [Isaac et al. 2010], o principal desafio em proporcionar segurança nessas redes consiste no fato de que em nenhum momento, durante a comunicação V2V a verdadeira identidade dos motoristas deve ser exposta, uma vez que adversários podem usar esta informação para atacar, aplicando identidades falsas para não serem descobertos. No entanto, os veículos e os condutores, muitas vezes, precisam divulgar suas identidades às RSUs em uma comunicação V2I, para poderem se comunicar com essas. Assim, garantir a autenticidade e o não repúdio sem afetar a privacidade é um grande desafio em ambientes veiculares. Outro desafio é a sensibilidade aos atrasos, uma vez que atrasos significantes proíbem o uso de protocolos e mecanismos de segurança que têm grande sobrecarga ou que dependem de múltiplos estágios de comunicação entre os nós, como por exemplo, os sistemas de reputação [Isaac et al. 2010].

Na tentativa de atender aos requisitos das redes veiculares, em julho de 2010, o IEEE 802.11p foi definido por um grupo de trabalho do IEEE para facilitar a implantação dessas redes em ambientes de alta velocidade (comunicação V2V e V2I). O padrão de acesso sem fio em ambientes veiculares - WAVE (*Wireless Access in the Vehicular Environment*), além de ser um modo de operação do IEEE 802.11p, é composto por um conjunto de normas, dentre estas, destaca-se a IEEE 1609.2 (serviços de segurança WAVE). Esta norma define os mecanismos de segurança a serem usados, o formato dos pacotes, os protocolos para tratamento das mensagens e um certificado digital compacto especial para comunicações veiculares. No entanto, o padrão IEEE 1609.2 não define como deve ser a identificação do veículo e como proteger a privacidade, deixando essas questões em aberto [Lin et al. 2008, Sukuvaara et al. 2013].

Diante das recentes pesquisas a cerca das inovações das redes veiculares, os objetivos deste capítulo consistem em revisitar os conceitos e as vulnerabilidades de segurança existentes em VANETs, e analisar de forma crítica os desafios de segurança e as principais contramedidas propostas em trabalhos acadêmicos. As questões-chave analisadas neste capítulo focam na autenticação de usuários e veículos versus a privacidade dos dados, nos ataques gerados por nós maliciosos e nos novos ataques decorrentes do uso de tecnologias como computação em nuvem e da integração dessas redes com modelos visionados para a Internet do Futuro.

Este capítulo está dividido em cinco seções. Nesta primeira seção, foi apresentada uma contextualização, destacando os objetivos e a motivação para a escolha do tema. A Seção 4.2 apresenta uma visão geral sobre redes veiculares, abordando as principais características e restrições, os principais padrões de comunicação e alguns domínios de aplicações de redes veiculares. Na Seção 4.3, são elencados os principais requisitos de segurança em redes veiculares, o perfil dos atacantes e os principais ataques descritos na literatura. A Seção 4.4 apresenta o estado da arte em termos das principais contramedidas aos ataques analisados, dentre estas destacam-se: os serviços de segurança WAVE; os mecanismos de autenticação; os controles criptográficos e os sistemas de reputação. Por fim, a Seção 4.5 traz uma síntese dos principais aspectos da segurança em redes veiculares analisados e as tendências de pesquisa nesta área.

## 4.2. Visão Geral sobre Redes Veiculares

As redes veiculares são um tipo de rede sem fio em ascensão devido aos avanços na indústria automobilística e nas tecnologias de comunicação sem fio. Os avanços na indústria automobilística estão possibilitando agregar novas tecnologias aos veículos, provendo aos usuários acesso a novos serviços, como informações sobre o ambiente e as condições do trânsito, alertas de perigos nas vias, conexão com a Internet, entre muitos outros [Macedo et al. 2013]. Além disso, a evolução das tecnologias de comunicação sem fio, a reserva de uma banda dedicada para comunicação de curto alcance (DSRC do Inglês *Dedicate Short-range Communications*) e a definição de um conjunto de padrões específicos para as redes veiculares (IEEE 1609) tem contribuído para o amadurecimento destas redes [Al-Sultan et al. 2014].

Os veículos são os principais componentes de uma rede veicular, os chamados nós. Estes possuem uma interface de comunicação sem fio para enviar, receber ou trocar informações. A Figura 4.1 ilustra o processo de comunicação em uma rede veicular. Os nós $A$, $B$ e $C$ representam um cenário em que os veículos estão circulando por uma via e trocando informações entre si e com uma infraestrutura fixa no acostamento. A comunicação entre os nós $A$, $B$ e $C$ caracteriza uma comunicação de veículo para veículo (V2V). Neste modelo, para que uma informação de alerta seja encaminhada do nó $A$ para o $C$, é preciso que o nó $B$, que se encontra no alcance de ambos, faça o encaminhamento da mensagem da origem ($A$) para o destino ($C$). A comunicação representada pelo nó $A$ e a infraestrutura $I1$ (V2I) corresponde à troca de informações entre os nós de uma rede veicular e equipamentos fixos ou unidades de acostamento (RSUs − *Road Side Unit*) ao longo da via. Neste modo, um nó RSU pode estabelecer a comunicação com outras redes de serviços ou até mesmo com a Internet.

As VANETs são geralmente comparadas com as redes *ad hoc* tradicionais, porém existem alguns aspectos que são favoráveis às redes veiculares, como a baixa restrição do consumo de energia, o poder computacional elevado e a mobilidade previsível dos nós. O consumo de energia de um nó veicular não é considerado crítico pois este possui uma fonte de energia superior a de um nó *ad hoc* tradicional. O poder computacional também é superior, podendo ter maior capacidade de processamento, armazenamento e largura de banda, e ainda, agregar outros recursos, como antenas com tecnologias avançadas, sistemas de posicionamento global (GPS), sensores, entre muitos outros. Por fim, a mobilidade dos nós em um ambiente urbano pode ser previsível, pois os veículos tendem a seguir o trajeto

estabelecido pelas vias e sinalizações de trânsito [Sichitiu e Kihl 2008a].

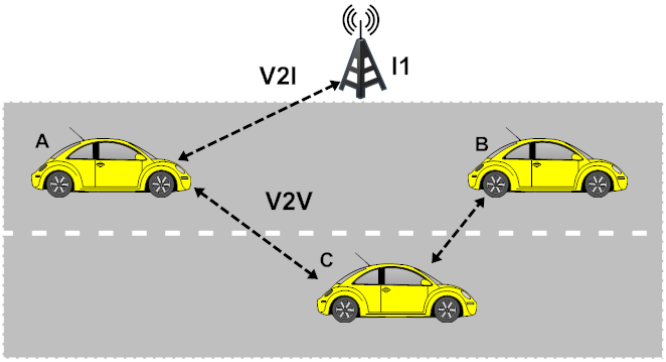



**Figura 4.1. Esquema de comunicação em uma rede veicular**

### 4.2.1. Características das Redes Veiculares

Os aspectos citados anteriormente são favoráveis às redes veiculares, contudo, existem outras características específicas do ambiente e precisam ser consideradas no desenvolvimento de seus protocolos e serviços [Al-Sultan et al. 2014, Hartenstein e Laberteaux 2008, Mejri et al. 2014]. Entre estas, citam-se:

- **Densidade variável da rede**: densidade pode variar de muito alta, a exemplo de um engarrafamento, ou muito baixa, como em horários noturnos ou nos subúrbios;
- **Escala da rede**: pode abranger áreas de grande densidade, como o centro de cidades ou rodovias de grande circulação;
- **Alta mobilidade**: é considerada uma das mais importantes, pois os nós estão sempre em movimento e se deslocando em velocidades elevadas e com direções variadas;
- **Topologia dinâmica da rede**: devido à alta mobilidade, a topologia muda rapidamente. O tempo de comunicação entre os nós geralmente é muito curto, principalmente, se estiverem se movendo em direções contrárias, fazendo com que o alcance da comunicação (diâmetro efetivo da rede) mude constantemente;
- **Desconexões frequentes**: a topologia dinâmica, a alta mobilidade e outros fatores como condições climáticas e densidade do tráfego podem causar desconexões frequentes dos nós.

Estas especificidades dificultam ou impossibilitam a aplicação de padrões e protocolos já consolidados para as redes *ad hoc* em redes veiculares, determinando novos desafios para o processo de comunicação neste ambiente. Em [Al-Sultan et al. 2014] são apresentados alguns dos principais desafios em VANETs:

- **Encobrimento de sinal**: objetos localizados entre dois veículos que estejam se comunicando é um dos desafios que afeta a eficiência das VANET. Estes objetos podem ser outros veículos ou construções distribuídos em torno das rodovias ou das cidades;

- **Limitações de banda**: deve-se realizar um gerenciamento eficiente de banda para que não ocorra um congestionamento do canal. Isto ocorre porque neste modelo de rede não se tem um coordenador central que controla a comunicação entre os nós, e que teria a responsabilidade de gerenciar a banda e as operações de contenção;

- **Conectividade**: devido à alta mobilidade e às mudanças rápidas da topologia, que levam a uma fragmentação frequente nas redes, o tempo de vida de uma ligação precisa ser alongado o quanto for possível. Esta tarefa pode ser feita através do aumento da potência de transmissão, contudo, isto pode levar à degradação do rendimento;

- **Diâmetro pequeno**: o diâmetro efetivo de uma VANET é pequeno, o que leva a uma conectividade fraca entre os nodos. Deste modo, é impraticável para um nó manter de forma completa uma topologia global da rede. Esta restrição do diâmetro da rede resulta em problemas, quando se aplica, algoritmos de roteamento existentes em redes *ad hoc*;

- **Privacidade e segurança**: manter um equilíbrio entre segurança e privacidade é um dos principais desafios das VANETs. Um receptor quer ter certeza que pode confiar na fonte da informação. Entretanto, ter acesso a identidade do emissor pode contradizer os requisitos de privacidade deste emissor;

- **Protocolos de roteamento**: devido às características da rede, desenvolver um protocolo de roteamento eficiente que possa entregar um pacote com o período mínimo de tempo e com pouca perda de pacotes é considerado um desafio crítico nas VANETs.

### 4.2.2. Padrões de Comunicação usados em Redes Veiculares

A reserva de uma faixa de comunicação de curto alcance dedicada (DSRC) pode ser considerada uma das primeiras iniciativas de padronização das tecnologias, específicas para as comunicações veiculares de curto alcance, V2V e V2I. A iniciativa de alocação desta faixa partiu dos Estados Unidos, que, em 1999, através da FCC (*Federal Communications Commission*), alocaram a banda de 5.9 GHz (5.850 – 5.925 GHz) para este fim. Posteriormente, surgiram outras iniciativas de alocar esta faixa de comunicação com o mesmo objetivo, como a Europa e o Japão, que alocaram a banda de 5.8 GHz [Li e Wang 2007, Al-Sultan et al. 2014, Faezipour et al. 2012]. Em 2002, a FCC sofreu grande influência da ITSA (*Intelligent Transportation Society of America*) em relação ao licenciamento, regras de serviços, e as possíveis tecnologias para a banda DSRC. As recomendações sugeriam a adoção pela FCC de um padrão único de arquitetura para as camadas físicas (PHY) e de controle de acesso ao meio (MAC) e que esta fosse desenvolvida pela ASTM (*American Society for Testing and Materials*), utilizando como base o padrão IEEE 802.11.

Em 2004, um grupo de trabalho da IEEE (grupo de trabalho p ou TGp do grupo de trabalho IEEE 802.11) assumiu o papel iniciado pela ASTM e começou o desenvolvimento de uma emenda ao padrão 802.11 para incluir as demandas das redes veiculares. Esta emenda ficou conhecida como 802.11p. Outro grupo de trabalho da IEEE (grupo de trabalho 1609) ficou responsável por desenvolver as especificações para cobrir as camadas adicionais através de um conjunto de protocolos. Atualmente, o conjunto de padrões IEEE

1609 consiste em um conjunto de documentos que, coletivamente com o IEEE 802.11p, são chamados de WAVE, que tem como objetivo facilitar o acesso sem fio às redes veiculares. Em termos de nomenclatura, comumente refere-se ao projeto conceitual como arquitetura WAVE e aos sistemas que o utiliza como sistemas WAVE [Toor et al. 2008, Al-Sultan et al. 2014, Faezipour et al. 2012].

A família de padrões IEEE 1609 é dividida em uma série de documentos que descrevem o funcionamento e os protocolos da arquitetura WAVE. A Figura 4.2 demonstra a organização da arquitetura em relação ao modelo ISO/OSI, e também uma referência entre as camadas e os documentos que formam a família de padrões IEEE 1609, conjuntamente com o protocolo IEEE 802.11p.

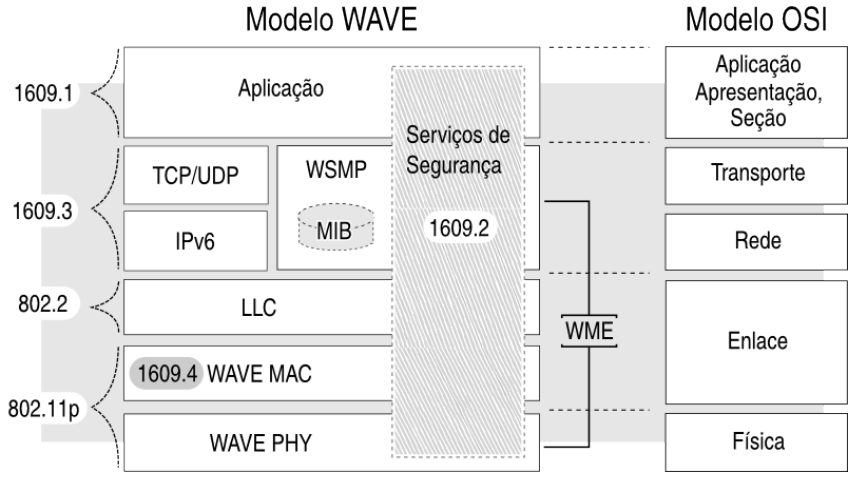



**Figura 4.2. Arquitetura WAVE [Uzcategui e Acosta-Marum 2009]**

Em síntese, os principais documentos da família IEEE 1609 são:

- **IEEE P1609.0**: descreve a arquitetura WAVE. Este define o funcionamento dos padrões e os serviços necessários para que os dispositivos possam se comunicar, utilizando os múltiplos canais DSRC em um ambiente de alta mobilidade.

- **IEEE P1609.1**: está relacionado com o gerenciamento de recursos (define os fluxos de dados e os recursos). Este descreve também os componentes básicos da arquitetura WAVE e define as mensagens de comando, os formatos para armazenamento dos dados e especifica os tipos de dispositivos que podem ser suportados por uma unidade de bordo (*On-Board Unit* − OBU)[1].

- **IEEE 1609.2**: refere-se aos serviços de segurança para as aplicações e o gerenciamento de mensagens. Este documento define os métodos de processamento e os formatos das mensagens de segurança utilizados pelos sistemas WAVE e DSRC. Este descreve como prover segurança para as mensagens de aplicações e de gerenciamento, bem como as funções necessárias para suportar mensagens seguras e a privacidade dos veículos (ver Seção 4.4.1).

[1]Unidade embarcada nos veículos que permite a comunicação entre os nós da rede.

- **IEEE 1609.3**: descreve os serviços para as camadas de rede e de transporte, como o roteamento e endereçamento com suporte a troca de mensagens seguras. O WAVE suporta duas pilhas de protocolo: o IP na versão 6 (IPv6) e o WSMP (*WAVE Short-Message Protocol*). A razão para ter duas pilhas de protocolo é a necessidade de suportar comunicações de alta prioridade e sensíveis ao tempo, como também aplicações não tão exigentes, como as transações que utilizam as transmissões TCP/UDP. Este padrão também define um conjunto de funções de gerenciamento, denominado WME (*WAVE management entity*), que deve ser utilizado para prover serviços de rede, e uma base de informações de gerenciamento (MIB, do Inglês *Management Information Base*).

- **IEEE 1609.4**: descreve as operações em múltiplos canais que utilizam o protocolo 802.11p (controle de acesso ao meio e camada física) para a arquitetura WAVE.

- **IEEE 1609.11**: define os serviços e o formato das mensagens necessárias para utilização de sistemas de pagamentos eletrônicos seguros. O IEEE 1609.12 especifica os valores dos identificadores que foram alocados para os sistemas WAVE.

As camadas PHY e MAC, especificadas pelo IEEE 802.11p, são baseadas no IEEE 802.11a para definição da camada física e no IEEE 802.11e para a camada de controle de acesso ao meio, respectivamente. A camada IEEE 802.11p PHY é baseada na OFDM (do inglês *Orthogonal frequency-division multiplexing*), que utiliza uma técnica de modulação que faz a multiplexação por divisão de frequência para encaminhar os sinais através de diferentes canais/frequências. As taxas de fluxo estão entre 3-27 Mbps, utilizando canais com largura de 10MHZ, e podendo alcançar cerca de 1000m [Mejri et al. 2014]. A camada IEEE 802.11p MAC utiliza o EDCA (do inglês *Enhanced Distributed Channel Access*), que é um melhoramento do DCF (do inglês *Distributed Coordination Function*), um protocolo que utiliza uma função de coordenação distribuída, em que a decisão de qual estação pode transmitir é realizada individualmente pelos pontos da rede, a fim de evitar colisões. O EDCA introduz, entre outras características, o conceito de gerenciamento com qualidade de serviço, utilizando categorias de acesso (divididas em tráfego de fundo, de melhor esforço, de vídeo e de voz), para garantir que as mensagens de segurança sejam transmitidas dentro de um tempo razoável [Uzcategui e Acosta-Marum 2009].

Um sistema WAVE é composto basicamente por entidades que são denominadas unidades de bordo (OBUs) e unidades de acostamento (RSUs). Uma OBU é um componente embarcado nos veículos que funciona enquanto estes transitam. Estas unidades realizam diversas funções, dentre estas, o roteamento entre os nós da rede, o controle de congestionamento e a transferência de mensagens confiáveis. As entidades RSUs são componentes instalados em postes de luz, semáforos, em margens das vias, entre outros.

Atualmente, existem outros padrões de tecnologias sem fio que podem ser utilizadas para prover serviços e informações para uma rede veicular. Estas tecnologias incluem as redes de celular (LTE, GSM, entre outras) e as redes locais sem fio (WLAN 802.11 a/b/g/n). De acordo com [Hill e Garrett 2011], as redes de telefonia celular não são apropriadas para troca de dados em tempo real utilizadas nas comunicações V2V e V2I, devido à latência da rede e às possíveis quedas de conexão, contudo, estas podem prover serviços valiosos para estes ambientes, principalmente, para aplicações que necessitam de

conectividade. Os aparelhos celulares (*smartphones*) são capazes de oferecer informações e entretenimento durante a viagem muito além do que os meios de transmissão por banda AM/FM comuns nos veículos. O uso destas tecnologias não está limitado a receber apenas dados da viagem, de modo que as conexões com a rede de telefonia celular estão sendo gradativamente utilizadas para outros fins, como coletar dados para investigação ou para aplicações telemáticas, especialmente, em frotas de veículos (táxi, caminhões, etc).

Em relação às redes sem fio locais, que utilizam a tecnologia Wi-Fi (*Wireless Fidelity*), foram inicialmente desenvolvidas para prover comunicação sem fio no lugar das redes locais cabeadas. Os equipamentos são amplamente utilizados em residências, no comércio e ambientes industriais. Os protocolos desta tecnologia são descritos pelo padrão IEEE 802.11 original, e define as especificações das camadas PHY e MAC para prover a conectividade sem fio para dispositivos fixos e móveis em uma rede local. O protocolo de controle de acesso ao meio (MAC) é o principal elemento de uma WLAN, responsável por gerenciar as situações de congestionamentos que podem ocorrer na rede. As redes que seguem o padrão podem implementar um dos dois métodos de acesso, o DCF (*Distributed Coordination Function*), para acesso assíncrono, distribuído e orientado a contenções, e o PCF (*Point Coordination Function*), para acesso centralizado e livre de contenções [Bononi et al. 2004]. O objetivo do PCF é dar suporte para serviços de tempo real, dando mais prioridade de acesso ao meio sem fio para as estações (nós) em relação ao DCF. O DCF é considerado a base para as redes sem fio *ad hoc* e utiliza o protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) para prevenção de colisões e acesso ao meio [Bononi et al. 2004, Mangold et al. 2002]. As redes WLANs e o padrão IEEE 802.11 original não foram projetados para suportar as aplicações veiculares que utilizam a comunicação V2V ou V2I, contudo, segundo [Hill e Garrett 2011] têm sido utilizadas eficientemente em aplicações que envolvem a comunicação entre veículos e estações fixas, como em estacionamentos e pátios de manutenção.

Outras tecnologias de comunicação sem fio podem ser utilizadas em aplicações de redes veiculares, contudo, estas são limitadas pelas características do ambiente. Como exemplo, tem-se o *Bluetooth* (padrão IEEE 802.15.1), utilizado para comunicações sem fio de curto alcance para conectar dispositivos de uma rede PAN (*Personal Area Network*). Esta tecnologia é considerada de baixo custo, de fácil uso e é amplamente utilizada nos veículos através dos dispositivos multimídias. Contudo, no contexto da comunicação veicular, o *Bluetooth* tem diversas desvantagens. O número limitado de nós, a baixa taxa de transferência e o curto alcance da comunicação são exemplos dos limites que esta tecnologia possui diante das redes veiculares [Sichitiu e Kihl 2008b].

### 4.2.3. Redes Veiculares, Nuvem e a Internet do Futuro

Os recentes avanços tecnológicos, principalmente nas áreas da computação móvel, redes sem fio e sensoriamento remoto, estão impulsionando o desenvolvimento dos sistemas de transporte inteligente (ITS). Os veículos estão se tornando sistemas computacionais sofisticados, embarcados com diversos sensores e computadores dedicados a funções específicas. Com o advento da comunicação sem fio, estes podem trocar as informações coletadas de seus sensores e do ambiente com os demais veículos que estejam por perto [Papadimitratos et al. 2009].

Os veículos podem ser considerados uma plataforma de observação ideal do ambiente, contudo, se as informações coletadas permanecerem localmente podem não ter muita relevância em uma rede veicular. Tendo como base este princípio, os autores em [Gerla 2012] propuseram um modelo de computação em nuvem veicular (VCC do Inglês *Vehicular Cloud Computing*), baseado na computação em nuvem móvel (MCC do Inglês *Mobile Cloud Computing*). Uma MCC utiliza o poder computacional dos dispositivos móveis a fim de permitir que os dados sejam processados e armazenados localmente, minimizando os custos de fazer o *upload* ou *download* dos dados da Internet. Deste modo, alguns dos benefícios em relação a uma nuvem na Internet é a redução dos atrasos na comunicação, redução da área de alcance e a ampliação das possíveis aplicações [Gerla 2012].

Uma VCC aproveita a capacidade de processamento e armazenamento de uma coleção de veículos para formar uma nuvem na qual serviços são produzidos, mantidos e consumidos [Lee et al. 2014]. Por exemplo, em uma situação em que um motorista busca por um restaurante em uma cidade, é mais útil se ele conseguir as recomendações diretamente com os veículos próximos, que possuem maior conhecimento da vizinhança, do que recorrer à Internet. Em [Lee et al. 2014], os autores introduzem o conceito de rede em nuvem veicular (VCN do inglês *Vehicular Cloud Networking*). Este conceito é baseado na integração das redes orientadas a informações (ICN do inglês *Information-centric networking*) ou analogamente, redes orientadas a conteúdo (CCN do inglês *Content Centric Network*) com a VCC. O ICN é um conceito que resulta das atividades de pesquisa relacionadas com a Internet do Futuro e baseia-se em uma arquitetura de comunicação para distribuição eficiente de conteúdo na Internet. Neste conceito, o paradigma principal para distribuição não é a comunicação fim a fim, mas sim a distribuição de conteúdo. Este paradigma utiliza atributos dos dados ou do nó como identificação do conteúdo, geolocalização ou contexto, ao invés de um endereço específico, como por exemplo o endereço IP (*Internet Protocol*) [Ahlgren et al. 2012].

O conceito de VCN [Lee et al. 2014] tem como base a arquitetura NDN (*Named Data Networking*), a qual segue a arquitetura de ICN e foi estendida para as VANETs. Esta arquitetura trabalha com dois tipos de pacotes: pacote de interesse e pacote de dados (conteúdo), que se referem ao cliente e ao distribuidor de conteúdos, respectivamente. Basicamente, esta arquitetura funciona do seguinte modo: um cliente faz uma requisição de um conteúdo transmitindo o seu pacote de interesse para os potenciais distribuidores de conteúdo. Quando um determinado distribuidor recebe este pacote de interesse, este confirma que seus dados coincidem com a busca e responde encaminhando o pacote de dados para o cliente através do caminho reverso do pacote de interesse. A arquitetura NDN permite que os roteadores façam o armazenamento dos dados para que agilizem as próximas consultas recorrentes, tornando a distribuição de conteúdo mais eficiente.

O objetivo de uma VCN é criar uma nuvem veicular temporária, com a participação de veículos e RSUs. Os membros desta nuvem colaboram para produzir serviços que não poderiam ser realizados de modo individual. A Figura 4.3 ilustra o funcionamento de uma VCN [Lee et al. 2014]. Quando um determinado veículo, no caso $V_1$, executa uma aplicação, este se torna o líder da nuvem e, então, recruta membros que podem prover recursos para compor a nuvem veicular. O alcance desta nuvem depende da aplicação, mas pode ser uma distância pré-definida, como parte de uma via ou um cruzamento, etc. O líder transmite uma mensagem em *broadcast* para os veículos de acordo com o

alcance determinado, conforme o tipo de recursos necessários. Os veículos que quiserem compartilhar os recursos, no exemplo representados por $V_2$ e $C_1$, respondem a mensagem enviada pelo líder com a informação sobre a capacidade de seus recursos. Após receber as mensagens de retorno, o líder seleciona os membros com os quais organiza a nuvem.

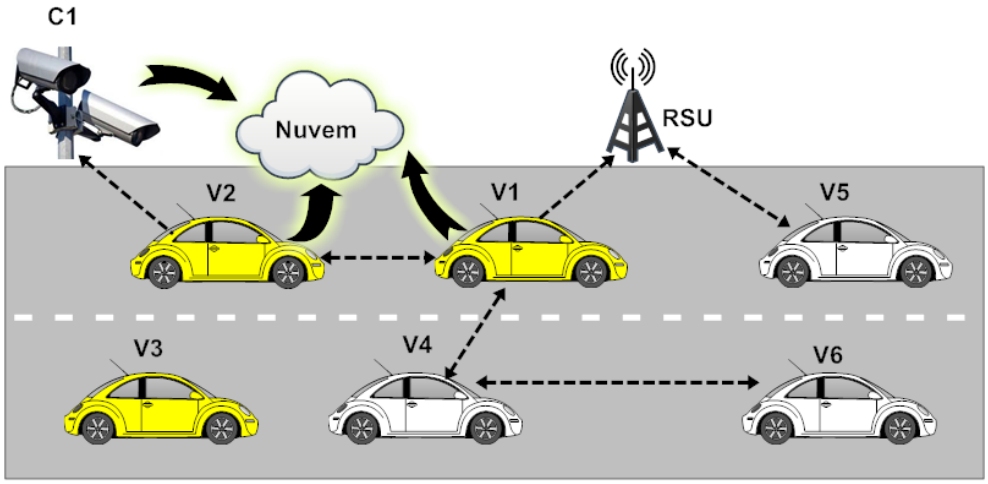



**Figura 4.3. Funcionamento de uma VCN [Lee et al. 2014]**

Com a nuvem formada, o líder divide a aplicação em diversas tarefas e distribui para os membros, com base na acessibilidade e disponibilidade dos recursos. Após completar a execução das tarefas, os membros retornam os resultados para o líder. Após coletar os resultados de todas as tarefas, o líder processa e salva a saída obtida. Caso não seja possível processar ou armazenar o conteúdo em um único nó, este pode selecionar outros nós para fazê-lo, representado pelo $V_4$. Por fim, este conteúdo fica acessível para todos os veículos da rede, na Figura, representados por $V_5$ e $V_6$. Caso um veículo deixe a nuvem, o líder pode selecionar a substituição do mesmo, encaminhando uma mensagem para os veículos dentro do alcance delimitado, a fim de completar o conjunto de recursos necessários. E, caso o líder deixe a nuvem ou ela não seja mais necessária, o líder encaminha uma mensagem para os membros, removendo-os da lista de membros e liberando-os para que possam fazer parte de outras nuvens.

De acordo com [Faezipour et al. 2012], algumas companhias automotivas têm dado alguns passos em direção à conectividade entre veículos e a nuvem. Em particular, a Toyota tem anunciado uma parceria com a Microsoft para oferecer esta conectividade. Os planos são para, a partir de 2015, conectar os carros na plataforma de nuvens Microsoft Azure para prover uma solução de telemática em nuvem. Ainda segundo os autores, este tipo de comunicação é útil para prover uma assistência ativa ao motorista e para o rastreamento de veículos em redes de gerenciamento de frotas.

Em relação a tecnologias emergentes, especificamente em relação a IoT (do inglês *Internet of Things*), [He et al. 2014] consideram que esta pode trazer soluções que irão transformar os sistemas de transporte e os serviços da indústria automotiva. Considerando que os veículos trazem sensores cada vez mais poderosos, conectividade, capacidade de processar dados, as tecnologias da IoT podem ser utilizadas para juntar estas capacidades e compartilhar os recursos subutilizados. Com a IoT, é possível rastrear a localização de cada veículo, monitorar seu movimento e prever sua localização futura. Ainda segundo

os autores, na IoT, pode haver a integração da computação em nuvem, das redes de sensores sem fio, das redes de sensores RFID, das redes de satélites e de outras tecnologias. Esta união vai possibilitar a criação de uma nova geração de nuvens de dados veiculares baseadas na IoT, para prover serviços que, entre outras coisas, aumentem a segurança nas estradas, reduzam os engarrafamentos, gerenciem o tráfego e recomendem a manutenção ou reparo do veículo [He et al. 2014].

### 4.2.4. Domínios de Aplicações em Redes Veiculares

A maioria das aplicações emergentes suportadas por redes veiculares estão relacionadas às questões de segurança necessárias a todos os veículos. Contudo, o acesso às redes infraestruturadas permite que inúmeras aplicações e serviços sejam providos para estes ambientes [Karagiannis et al. 2011, Silva et al. 2014b]. Com base nos trabalhos de [Papadimitratos et al. 2009, Hossain et al. 2010, Karagiannis et al. 2011], as aplicações podem ser classificadas em três tipos: aplicações de segurança (*safety*) no transporte, aplicações de gerenciamento e eficiência de tráfego e aplicações de *Infotainment*.

As aplicações do primeiro tipo buscam melhorar a segurança dos usuários nas vias, notificando os veículos sobre qualquer situação de perigo na vizinhança com intuito de reduzir a probabilidade de acidentes [Hossain et al. 2010]. Os exemplos mais comuns deste tipo são os alertas de colisão, assistência para mudança de faixa, aviso de ultrapassagem, e avisos de veículos de emergência. As aplicações de gerenciamento e eficiência de tráfego focam em melhorar o fluxo, a coordenação e a assistência do tráfego, e também, em prover a atualização das informações locais, como mapas e mensagens de relevância delimitadas no espaço ou no tempo. As aplicações de gerenciamento de velocidade e de navegação cooperativa são dois grupos típicos deste tipo de aplicações, a saber:

- **Aplicações de gerenciamento de velocidade**: têm como objetivo auxiliar o motorista a controlar a velocidade para que tenha uma direção estável e evitar que o mesmo tenha que parar sem necessidade. Notificadores ou reguladores de limite de velocidade ou a indicação no painel de uma luz verde que indique quanto esteja em uma velocidade ideal são dois exemplos deste grupo;
- **Navegação cooperativa**: utilizada para aumentar a eficiência do tráfego pelo gerenciamento da navegação dos veículos através da cooperação entre estes e entre as unidades de acostamento das vias. Alguns exemplos deste tipo de aplicações são: informações de tráfego e recomendação de um itinerário, cooperação para o controle de uma navegação adaptativa e em grupos.

O terceiro tipo são as aplicações que oferecem, entre outras coisas, informação e entretenimento para o conforto dos motoristas e passageiros, e são comumente citadas como aplicações de *Infotainment*. Este tipo de aplicação pode oferecer uma grande variedade de serviços, como disseminação de conteúdos multimídia em tempo real ou não, jogos interativos, busca de informações locais, como restaurantes ou postos de combustível, informações climáticas ou até mesmo acesso a Internet, para *download* de informações, músicas, etc [Hossain et al. 2010]. Alguns trabalhos recentes apresentam abordagens específicas para distribuição de conteúdo aos usuários das redes veiculares [Mezghani et al. 2014b].

A Tabela 4.1 apresenta uma lista de aplicações e alguns de seus requisitos. Cada aplicação está relacionada ao tipo de comunicação, ao tipo da mensagem, ao tempo da mensagem, à latência (tempo de atraso máximo requerido pela aplicação) e a outros requisitos (prioridade, alcance, etc) [Papadimitratos et al. 2009].

**Tabela 4.1. Características de Aplicações Veiculares [Papadimitratos et al. 2009]**

| # | Aplicações | Comunicação | Tipo | Tempo | Latência | Outros |
|---|---|---|---|---|---|---|
| 1 | Alerta de veículo lento | *ad hoc*, V2V | *broadcast* permanente | 500ms | 100ms | Alcance: 300m, alta prioridade |
| 2 | Alerta de colisão em cruzamento | *ad hoc*, infraestrutura, V2V, V2I | *broadcast* permanente | 100ms | 100ms | Posicionamento preciso em um mapa digital, alta prioridade |
| 3 | Pré-colisão | *ad hoc*, V2V | *broadcast* periódico, *unicast* | 100ms | 50ms | Alcance 50m, prioridade alta/média |
| 4 | Gerenciamento de Cruzamento | infraestrutura, *ad hoc*, V2I, V2V | *broadcast* periódico, *unicast* | 1000ms | 500ms | Precisão do posicionamento: < 5m |
| 5 | *Download* de Mídia | infraestrutura, rede de telefonia celular, etc | *unicast*, *broadcast*, sob-demanda | n/d | 500ms | Acesso à internet, Gerência dos direitos reservados |
| 6 | Assistência para direção ecológica | infraestrutura, *ad hoc*, V2I, V2V e rede de telefonia celular | *unicast*, *broadcast*, sob-demanda | 1000ms | 500ms | Acesso à Internet disponibilidade do serviço |

As aplicações 1, 2 e 3 estão relacionadas com a categoria de segurança no transporte. Estas utilizam em sua maioria a comunicação *ad hoc*, possuem uma restrição de tempo rigorosa e tem uma prioridade elevada no enlace dos dados. Entre estas, a aplicação de pré-colisão é a mais rigorosa com o requisito de latência. Este tipo de aplicação pode ser mais relevante para comunicações de curto alcance, que corresponde a pequenas distâncias entre os veículos. A aplicação 4 está relacionada com a categoria gerenciamento e eficiência do tráfego e caracteriza-se pela comunicação *ad hoc* com a infraestrutura (RSUs, pedágios, etc). A prioridade determinada é baixa em relação as aplicações de segurança e os requisitos de latência são maiores. As demais aplicações, 5 e 6, estão relacionadas com a categoria *infotainment*. Este tipo de aplicação depende mais da comunicação com redes infraestruturadas (redes de telefonia celular, Internet, etc) do que a comunicação V2V. A preocupação com a latência é menos que as outras categorias de aplicação, contudo, existem outras questões a serem observadas, como o gerenciamento dos direitos do conteúdo a ser obtido e a disponibilidade de determinados serviços, bem como o acesso à Internet.

## 4.3. Ameaças e Ataques de Segurança em Redes Veiculares

Ao abordar as questões de segurança em redes veiculares é interessante especificar os requisitos de segurança que devem ser garantidos para o correto funcionamento dessas

redes (Seção 4.3.1), os perfis dos atacantes na rede (Seção 4.3.2), bem como os tipos de ataques que podem ser executados por estes (Seção 4.3.3). As próximas subseções detalham esses três aspectos.

### 4.3.1. Requisitos de Segurança

As VANETs devem satisfazer os seguintes requisitos de segurança no suporte aos diferentes tipo de aplicações [Raya et al. 2006a, Samara et al. 2010, Tangade e Manvi 2013]:

- **Confidencialidade**: o sigilo deve ser fornecido ao dado sensível que está sendo enviado pela VANET, sobretudo em aplicações financeiras e comerciais.
- **Integridade**: as mensagens enviadas através da rede não podem ser corrompidas. Possíveis ataques que possam comprometer a integridade das mensagens são os ataques de nós maliciosos ou falhas de sinal que produzem erros na transmissão.
- **Autenticidade**: a identidade dos nós na rede deve ser assegurada. Caso contrário, será possível a um atacante simular um nó legítimo, a fim de enviar e receber mensagens em seu nome.
- **Disponibilidade**: a rede deve estar disponível e funcional em todos os momentos, a fim de possibilitar o envio e recebimento de mensagens. Duas possíveis ameaças à disponibilidade são: os ataques de negação de serviço (DoS, *Denial of Service*) e ataques *jamming* [Xu et al. 2006]. Outro problema de disponibilidade pode ser causado por nós egoístas que não oferecem os seus serviços para o benefício de outros nós, a fim de salvar os seus próprios recursos como a energia da bateria. A disponibilidade também pode ser comprometida por interferências nos sinais de transmissão, pelo problema do terminal escondido, pelo devanecimento do sinal e outros.
- **Não repúdio**: um nó remetente pode tentar negar o envio de uma mensagem, a fim de evitar a sua responsabilidade pelo seu conteúdo. Garantir o não-repúdio é particularmente útil para detectar a ação de nós maliciosos.
- **Privacidade**: o sigilo das informações dos motoristas (tais como, identidade, velocidade, caminho percorrido) contra observadores não autorizados deve ser garantido.

### 4.3.2. Perfil dos Atacantes

O objetivo do atacante é criar problemas para outros usuários da rede, por exemplo alterando o conteúdo das mensagens ou invadindo sua privacidade. Antes de descrever os diferentes ataques em VANETs, os perfis dos ataques são categorizados. Um nó é considerado adversário ou malicioso se este tenta injetar qualquer tipo de mau comportamento na rede que pode fazer com que outros nós e/ou rede funcionem de forma inadequada [Tangade e Manvi 2013]. Os atacantes em redes veiculares podem ter perfis variados. Para classificar a capacidade destes atacantes, os autores em [Raya e Hubaux 2007] definiram quatro dimensões:

- **Interno versus externo**: o atacante interno é aquele que está autenticado, que tem conhecimento detalhado da rede e que pode se comunicar com os outros nós. Os ataques mais sofisticados podem ser elaborados quando este atacante tem todas as informações sobre a configuração da rede. O atacante externo é considerado pelos membros da rede como um intruso, sendo que os ataques que pode executar são limitados (por exemplo, o uso indevido de protocolos específicos da rede).

- **Malicioso versus racional**: o atacante malicioso não busca vantagens pessoais a partir dos seus ataques e tem como objetivo prejudicar os nós ou a funcionalidade da rede. Por isso, esse pode empregar qualquer meio sem se preocupar com custos e consequências. Esta categoria de invasores é considerada a mais perigosa, uma vez que pode causar graves danos à rede [Tangade e Manvi 2013]. Ao contrário, um atacante racional busca ganho pessoal e, portanto, é mais previsível no que diz respeito às ações e aos alvos. Em [Samara et al. 2010], os autores classificam ainda um invasor como egoísta quando este informa aos outros veículos presentes na rodovia que há congestionamento na estrada, apenas para tomar proveito da situação.

- **Ativo versus passivo**: um atacante ativo é aquele que pode gerar sinais e inserir ou modificar dados na rede com o objetivo de prejudicar outros nós ou parte da rede. Estes podem gerar pacotes contendo informações falsas ou não encaminhar os pacotes recebidos. Já um invasor passivo, apenas obtém informações para posterior uso. Estes atacantes escutam o canal sem fio para coletar informações de tráfego que podem ser transferidas para outros atacantes.

- **Local versus estendido**: um atacante local está limitado ao seu alcance, mesmo que este controle algumas poucas entidades (veículos ou unidades de acostamento). Ao contrário, o invasor estendido controla várias entidades (veículos ou unidades de acostamento) que estão espalhadas em toda a rede, aumentando assim o seu escopo.

### 4.3.3. Ataques

Para construir uma arquitetura de segurança robusta para VANETs, é importante estudar as características dos ataques que podem ocorrer. Assim como as redes clássicas, as redes veiculares são vulneráveis a muitos ataques. O fato destas redes ainda estarem em implementação faz com que a situação seja ainda mais complicada. Alguns ataques são vislumbrados e soluções são concebidas, considerando que um dia estes ataques serão de fato lançados sobre a rede, quando esta estiver em operação [Engoulou et al. 2014]. A seguir, são descritos os principais ataques analisados na literatura.

#### Ataques contra a Disponibilidade

- **Negação de serviço (DoS)**: o principal objetivo desta categoria de ataques é evitar que usuários legítimos acessem aos serviços ou recursos de rede. Em VANETs, este problema pode ser crítico, quando o usuário não pode se comunicar dentro da rede e transmitir dados para outros veículos, uma vez que isto pode resultar em uma situação de risco de vida. Um atacante pode proferir ataques de DoS de três maneiras, a saber: (i) sobrecarrega o nó para que este não possa realizar outra tarefa,

tornando o nó continuamente ocupado; (ii) ataca o canal de comunicação gerando altas frequências de modo que nenhum veículo seja capaz de se comunicar com outros veículos na rede, e (iii) não encaminhamento (bloqueio) de pacotes [Pathre et al. 2013]. A seguir, estão descritos outros ataques desta categoria.

- **Negação de serviço distribuída (DDoS)**: são mais graves do que os ataques de DoS, pois estes partem de diferentes localizações e em diferentes horários, porém o objetivo geral é a violação da disponibilidade de um nó ou de uma unidade de acostamento - RSU. A Figura 4.5 ilustra um ataque de três veículos (B, C, D) que enviam uma grande quantidade de pacotes contra uma unidade de acostamento (RSU), ocasionando indisponibilidade da rede. Quando outros nós tentarem acessar a RSU, não será possível, pois esta estará sobrecarregada [Pathre et al. 2013].

- **Supressão de mensagem**[2]: um atacante intercepta seletivamente pacotes da rede e suprime (bloqueia) esses pacotes que inclusive poderão ser utilizados em outro momento. O objetivo de tal ataque é, por exemplo, impedir o aviso de um congestionamento, evitando que os veículos busquem caminhos alternativos, obrigando estes a entrarem no congestionamento [Tangade e Manvi 2013].

- **Buraco negro (*black hole*)**: buraco negro é uma área da rede na qual o tráfego de rede é redirecionado ou passa naturalmente, porém, ou não há veículos neste local ou veículos maliciosos que estão nesta área se recusam a participar. Isto faz com que os pacotes de dados se percam na rede. A Figura 4.4 ilustra este ataque, no qual veículos maliciosos se recusam a transmitir a mensagem enviada pelo veículo C sobre um acidente e que deveria ser encaminhada para os veículos E e F [Al-kahtani 2012].

- **Temporização**[3]: ocorre quando um veículo malicioso recebe uma mensagem de emergência e não a transmite imediatamente aos veículos vizinhos. Ao invés de transmitir a mensagem, o veículo atrasa o envio, resultando no atraso de recebimento das mensagens pelos veículos vizinhos [Al-kahtani 2012].

- **Jamming**: é um ataque físico de negação de serviço, no qual o atacante transmite um sinal para perturbar o canal de comunicação, o que reduz a relação sinal ruído SNR (*Signal to Noise Ratio*) para o receptor [Mejri et al. 2014]. No cenário das VANETs, um atacante pode de forma relativamente fácil particionar a rede, sem comprometer os mecanismos de criptografia [Pathre et al. 2013]. Resultados publicados na literatura têm demonstrado o impacto devastador na rede causado por esses ataques [Avelar et al. 2014, Lima et al. 2013].

- **Software malicioso - *malware***: dada a existência de componentes de software para operar a unidade de bordo (OBU) e a unidade de acostamento (RSU), a infiltração de software malicioso (*malware*) é possível na rede durante a atualização/instalação do software nas unidades da rede veicular. O efeito de um *malware* é semelhante aos vírus e worms em uma rede de computadores normal, exceto que, em uma

[2]É um ataque que também viola a integridade da mensagem.
[3]Este ataque também viola a integridade.

rede veicular, interrupções de funcionamento normal podem ser seguidas por sérias consequências [Mejri et al. 2014].

- ***Spam***: como na Internet, as mensagens de spam não têm nenhuma utilidade para os usuários. Em uma rede veicular, que é um ambiente de rádio móvel, este tipo de ataque consome largura de banda e provoca colisões voluntárias. A falta de uma gestão centralizada do meio de transmissão torna mais difícil controlar estes ataques. Assim como os softwares maliciosos, os *spams* aumentam significativamente a latência da rede.

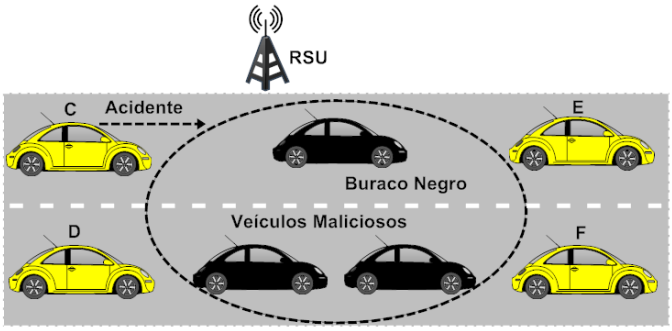



**Figura 4.4. Ataque Buraco Negro (*black hole*)**

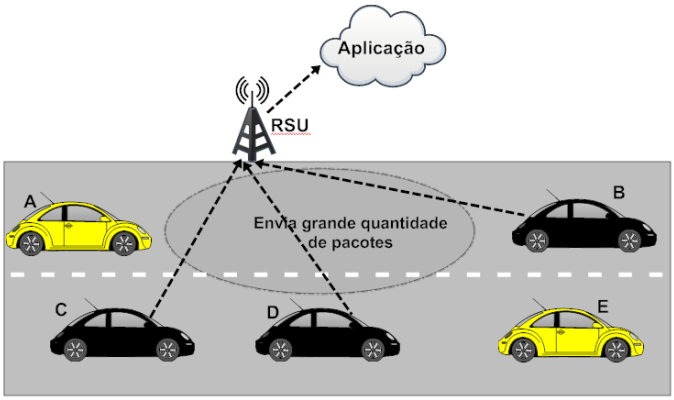



**Figura 4.5. Ataque de Negação de Serviço Distribuído (*DDoS*)**

### Ataques contra a Autenticidade e a Identificação

- **Forjamento de endereço (*address spoofing*)**: acontece quando o ataque fabrica um pacote contendo um endereço falso de origem, fazendo com que o nó atacado acredite que a conexão está vindo de um nó com permissão para se conectar à rede [Al-kahtani 2012].

- **Mascaramento**: um veículo malicioso falsifica sua identidade e finge ser outro veículo para ganhar acesso não autorizado a um recurso da rede. Por exemplo, um veículo malicioso pode fingir ser uma ambulância para obter vantagem no trânsito. Após o mascaramento, um atacante pode introduzir ou substituir um nó de rede para induzir outros a se conectarem a este, ao invés do dispositivo legítimo (p.ex uma unidade de acostamento), permitindo assim a captura de senhas de acesso e informações que por este passem a trafegar [Engoulou et al. 2014];

- **Sybil**: um atacante gera múltiplas identidades para simular vários nós na rede. Cada nó transmite mensagens com múltiplas identidades. Assim, outros veículos percebem que existem muitos veículos na rede ao mesmo tempo. Por exemplo, um atacante pode fingir e agir como uma centena de veículos para convencer os outros veículos da estrada que há congestionamento, com o objetivo de convencer os veículos a trafegarem por outras estradas [Tangade e Manvi 2013].

- **Tunelamento ou *wormhole attack***: neste ataque, pelo menos dois nós maliciosas cooperam para transferir pacotes da rede veicular por um túnel privado criado por estes. O principal objetivo destes nós maliciosos é afetar os pacotes de pedido de roteamento. Neste ataque, estes nós podem diminuir o número de contagens de saltos para que o que os pacotes passem sempre pelo túnel privado [Safi et al. 2009]. Em VANETs, normalmente, para aplicações que requerem roteamento, os protocolos AODV e DSR são usados, sendo que estes são vulneráveis a este tipo de ataque.

- **Replicação do certificado ou da chave**: o ataque consiste na utilização de chaves ou certificados duplicados, que são utilizados como prova de identificação para criar ambiguidade, dificultando a identificação de um veículo pelas autoridades, especialmente em caso de disputa (repudiação) [Mejri et al. 2014].

**Ataques contra a Integridade e Confiança dos Dados**

- **Forjamento de dados de posicionamento do GPS (*GPS spoofing*)**: o satélite GPS mantém uma tabela com a localização geográfica e a identidade do veículo na rede. Os atacantes usam um simulador de satélite GPS para gerar sinais mais fortes do que aqueles gerados pelo satélite real de forma a enganar os veículos, introduzindo uma localização falsa [Rawat et al. 2012]. Os atacantes podem ainda modificar pacotes de localização, repetir pacotes e bloquear pacotes urgentes de localização [Isaac et al. 2010].

- **Ilusão (ataque contra os sensores do veículo)**: é uma nova ameaça de segurança em aplicações VANET na qual o atacante engana ou interfere intencionalmente nos sensores do seu próprio veículo para produzir leituras erradas. Como resultado, mensagens de aviso de tráfego incorretas são transmitidas para os nós vizinhos, criando uma condição de ilusão em VANET. Os métodos de autenticação e integridade utilizados tradicionalmente em redes sem fio são inadequadas contra o ataque ilusão [Isaac et al. 2010, Al-kahtani 2012].

- **Injeção de informação falsa (*bogus information*)**: nesta categoria, um atacante pode ser um intruso ou um usuário legítimo que transmite informações falsas na rede veicular para obter vantagens ou afetar a decisão de outros veículos, por exemplo, em [Rawat et al. 2012, Al-kahtani 2012], os autores apresentam a transmissão de informações erradas sobre as condições de tráfego de modo a tornar mais fácil o seu movimento na estrada. O **ataque social** é um tipo de ataque desta categoria. Neste, o atacante procura confundir e distrair a vítima enviando mensagem antiética e/ou imoral para o motorista ficar perturbado. O usuário legítimo reage de forma irritada

ao receber esse tipo de mensagem podendo se distrair e causar um acidente [Rawat et al. 2012]. A Figura 4.6 ilustra um exemplo deste ataque.

- **Modificação de mensagem (*man in the middle*)**: nas redes veiculares, o atacante é um veículo que está inserido entre dois veículos que se comunicam. O atacante faz o intermédio entre a comunicação das duas vítimas, interceptando as mensagens enquanto estas acreditam que estão se comunicando diretamente. Este é um ataque que viola a autenticidade do remente e a integridade das mensagens [Mejri et al. 2014]. A Figura 4.7 mostra um ataque no qual o veículo malicioso M ouve a comunicação entre os veículos A e B, modifica o alerta recebido e propaga uma informação falsa (para os veículos B e C) como se fosse o veículo A. Em seguida, esta informação é difundida por toda a rede [Al-kahtani 2012].

- **Ataque de Mensagem Antiga - *Replay***: este é um ataque clássico que consiste em retransmitir uma mensagem já enviada, para obter benefícios referentes à mensagem no momento da sua apresentação. Por isso, o atacante injeta novamente os pacotes anteriormente recebidos na rede. Este ataque pode ser usado, por exemplo, para retransmitir quadros de *beacons*, de modo que o atacante possa manipular a localização e as tabelas de roteamento dos nós. Ao contrário de outros ataques, este pode ser realizado por falsos usuários [Mejri et al. 2014].

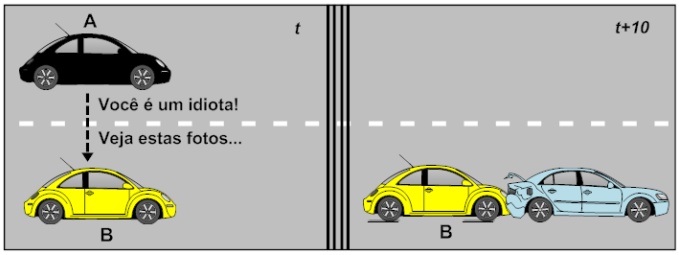



**Figura 4.6. Ataque Social**

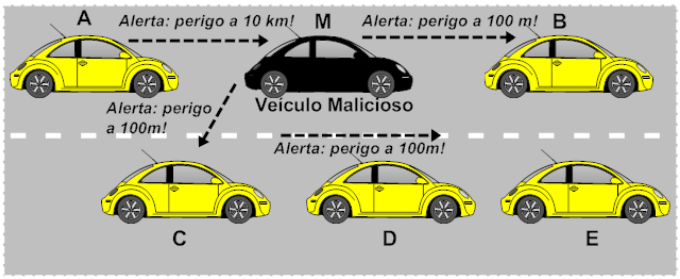



**Figura 4.7. Ataque de Modificação de Mensagem (*Man in the Middle*)**

### Ataques contra a Confidencialidade

- **Análise de tráfego**: um atacante que venha a ter acesso à rede pode interceptar o tráfego de pacotes e coletar dados que estejam sendo transmitidos sem o uso de criptografia. Este é um ataque passivo que é executado em redes *ad hoc* [Isaac et al. 2010].

- **Força bruta**: um ataque de força bruta pode ter como foco as mensagens trocadas (quebrar as chaves criptográficas) ou ainda o processo de identificação e autenticação. Em uma rede veicular, na qual os tempos de conexão são relativamente curtos, um ataque de força bruta não é fácil de realizar, uma vez que consome muito tempo e necessita de muitos recursos [Mejri et al. 2014, Pathre et al. 2013].

- **Revelação de identidade**: um atacante pode obter a identidade do proprietário de um determinado veículo violando a sua privacidade. Normalmente, o proprietário de um veículo também é seu motorista, por isso é simples a obtenção dos dados pessoais desta pessoa [Tangade e Manvi 2013].

**Outros Ataques**

- **Ataques contra o não-repúdio ou perda de eventos de rastreabilidade (*accountability*)**: de acordo com [Mejri et al. 2014], apesar de sua importância, não foi encontrado na literatura um documento afirmando a possibilidade deste ataque ocorrer nas redes veiculares. O ataque de não repúdio consiste em tomar medidas para permitir ao atacante negar a realização por ele de uma ou mais ações. Este tipo de ataque está essencialmente baseado na eliminação de traços de transações, criando uma confusão para a entidade de auditoria. Os ataques de *sybil* e duplicação de chaves e certificados podem servir como preliminar para o ataque de repúdio.

- **Ataques contra a privacidade**: estes ataques representam a violação da privacidade dos condutores e usuários em VANET. Vários estudos na literatura classificam os ataques de privacidade como uma categoria separada para VANETs. Como um exemplo prático, tem-se:

  - Rastreamento: a busca de um veículo durante a sua viagem.
  - Engenharia Social: se um veículo em um determinado momento está na garagem ou em circulação [Mejri et al. 2014].

- **Conluio**: um atacante pode formar alianças com outros nós da rede para alcançar um objetivo comum. Este objetivo pode ser o vandalismo ou o terrorismo na rede, resultando na indisponibilidade da rede ou de uma aplicação ou denegrir a reputação de (confiança) um veículo [Zhang 2011].

### 4.3.4. Correlação dos Ataques, Perfil dos Atacantes e Propriedades Violadas

Esta seção correlaciona, através da tabela 4.2, os ataques apresentados na Seção 4.3 com o perfil do atacante e as propriedades de segurança violadas.

**Tabela 4.2. Comparação dos Ataques de Segurança**

| Nome do Ataque | Perfil do Atacante | Propriedades Violadas |
|---|---|---|
| Negação de serviço (DoS) | Interno ou Externo, Malicioso, Ativo, Estendido | Disponibilidade |
| Supressão de Mensagem | Interno, Racional, Ativo, Local | Disponibilidade Integridade |
| Buraco Negro (*black hole*) | Interno, Malicioso, Ativo, Local, | Disponibilidade |
| Temporização | Interno, Racional, Ativo, Estendido | Disponibilidade |
| Jamming | Interno ou Externo, Malicioso, Ativo, Local | Disponibilidade |
| Negação de Serviço Distribuído (DDoS) | Interno ou Externo, Malicioso, Ativo, Estendido | Disponibilidade |
| Software Malicioso - *Malware* | Interno ou Externo, Malicioso, Ativo, Estendido | Disponibilidade Integridade |
| Spam | Interno ou Externo, Malicioso, Ativo, Estendido | Disponibilidade |
| Forjamento de Endereço | Interno, Malicioso, Ativo, Estendido | Autenticidade |
| Mascaramento | Interno, Racional, Ativo, Estendido | Autenticidade |
| *Sybil* | Interno, Malicioso ou Racional, Ativo, Local ou Estendido | Autenticidade |
| Tunelamento ou *Wormhole* | Interno, Malicioso ou Racional, Ativo, Estendido | Autenticidade Disponibilidade |
| Replicação do Certificado ou da Chave | Interno, Malicioso ou Racional, Ativo, Local | Autenticidade |
| GPS *spoofing* | Interno ou Externo, Malicioso, Ativo, Local | Integridade Autenticidade |
| Ilusão | Interno, Racional, Ativo, Local | Integridade |
| Injeção de informação falsa | Interno, Malicioso, Ativo, Local ou Estendido | Integridade |
| Ataque social | Interno, Malicioso, Ativo, Local | Integridade |
| Modificação de mensagem | Interno, Malicioso, Ativo, Local | Integridade, Disponibilidade |
| Ataque de Mensagem Antiga (*Replay*) | Interno, Malicioso, Ativo, Estendido | Integridade, Disponibilidade |
| Análise de Tráfego | Interno ou externo, Racional ou Malicioso, Passivo, Local | Confidencialidade |
| Força bruta | Interno ou Externo, Racional, Ativo, Local | Confidencialidade, Autenticidade |
| Revelação de Identidade | Interno ou Externo, Malicioso ou Racional, Passivo, Local | Confidencialidade, Privacidade |
| Ataques contra o não-repúdio e *accountability* | Interno, Malicioso, Ativo, Local | Confidencialidade |
| Ataques contra a privacidade | Interno ou Externo, Malicioso ou Racional, Ativo, Estendido | Confidencialidade, Privacidade |
| Conluio | Interno, Malicioso ou Racional, Ativo, Estendido | Integridade Disponibilidade |

## 4.4. Principais Contramedidas e suas Restrições

Esta seção apresenta as principais contramedidas existentes na literatura e suas limitações diante de conter os efeitos de ataques. A Seção 4.4.1 apresenta os serviços de segurança definidos pelo padrão IEEE 1609. A Seção 4.4.2 provê uma descrição dos mecanismos de criptografia e gerenciamento de chaves propostos para redes veiculares. A Seção 4.4.3 fornece uma visão geral sobre os mecanismos de autenticação e técnicas de anonimato. Finalmente, a Seção 4.4.4 detalha os sistemas de reputação e modelos de confiança.

### 4.4.1. Serviços de segurança WAVE (IEEE 1609.2)

Como descrito na Seção 4.2.2, a arquitetura WAVE é composta por seis documentos, dentre estes o documento 1609.2 especifica um conjunto de serviços para prover segurança às

mensagens WAVE contra análise de tráfego (*eavesdroping*), forjamento (*spoofing*) e outros tipos de ataques em ambientes de redes veiculares [Lin et al. 2008]. O padrão IEEE 1609.2 envolve basicamente de três componentes [Schütze 2011]:

- Algoritmos de assinaturas digitais usando criptografia de curvas elípticas (ECC), especificamente o padrão ECDSA (*Elliptic Curve Digital Signature Standard*);
- Esquema híbrido de cifragem assimétrica com ECC, especificamente o esquema ECIES (*Elliptic Curve Integrated Encryption Scheme*). A criptografia assimétrica é utilizada apenas para o transporte da chave simétrica;
- Esquema puramente simétrico para cifragem autenticada é utilizado para garantir a integridade de forma eficiente e, opcionalmente, para trocas cifradas com menos sobrecarga. O CBC-MAC com AES (AES-CCM) é um exemplo de esquema suportado.

O padrão IEEE 1609.2 define uma forma compacta de certificado digital, chamada de certificado WAVE, e define a existência de autoridades certificadoras. O padrão descreve uma aplicação denominada entidade de gerenciamento de certificados, responsável por gerenciar o certificado raiz e armazenar as listas de certificados revogados [Schütze 2011]. Os serviços de segurança WAVE definidos na IEEE 1609.2 consistem em [IEEE 2013]:

- **Serviços de processamento de segurança**: oferecem mecanismos para estabelecer comunicações seguras com o objetivo de proteger os dados e prover segurança para os anúncios de serviços WAVE (*WSAs - WAVE Service Advertisements*).
- **Serviços de gerenciamento de segurança**
  - Serviços de gestão de certificados: serviços providos pela Entidade de Gerenciamento de Certificados (*CME - Certificate Management Entity*) e que gerenciam informações relacionadas à validade de todos os certificados.
  - Serviços de gerenciamento de segurança de provedores de serviços: são providos pela Entidade de Gerenciamento de Provedores de Serviços (*PSSME - Provider Service Security Management Entity*) e gerenciam as informações relacionadas aos certificados e às chaves privadas que são usados no envio seguro dos anúncios de serviços WAVE (WSAs).

Uma implementação do WAVE deve incluir pelo menos um dos seguintes serviços de processamento de segurança indicados na [IEEE 2013]: $(i)$ gerar dados assinados; $(ii)$ gerar dados criptografados; $(iii)$ verificar os dados assinados; $(iv)$ decriptografar e criptografar os dados; $(v)$ gerar Serviços de Anúncio WAVE (WSA) assinados; $(vi)$ verificar assinaturas WSA no receptor; $(vii)$ gerar um pedido de certificado (*Certificate Signing Request - CSR*); $(viii)$ verificar resposta ao pedido de certificado; $(xi)$ verificar a lista de certificados revogados.

Os serviços e as entidades dos Serviços de Segurança WAVE estão ilustrados na Figura 4.8. A figura mostra os pontos de acesso de serviços (SAPs -Service Access

Points), que suportam as comunicações entre entidades dos Serviços de Segurança WAVE e outras entidades. A norma na IEEE 1609.2 especifica o processamento de segurança via primitivas definidas nestes SAPs.

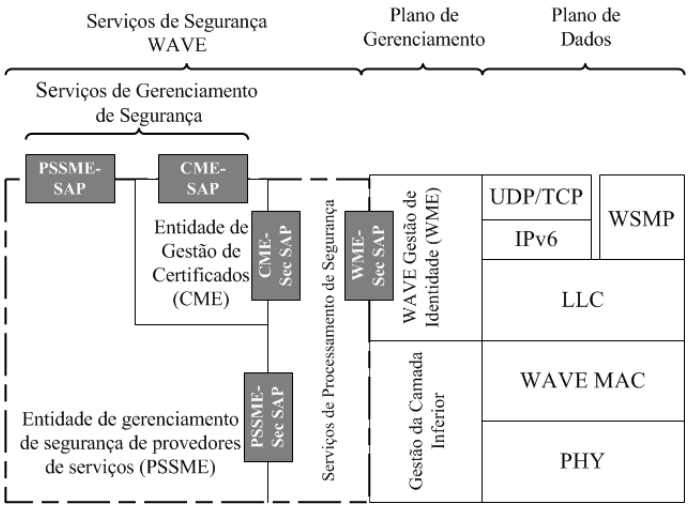



**Figura 4.8. Serviços de Segurança na Pilha de Protocolos WAVE [IEEE 2013]**

A Figura 4.9 ilustra o modelo geral para o processamento da segurança segundo a norma IEEE 1609.2. Os serviços de segurança são utilizados por uma Entidade de Comunicação Segura (SCE – *Secure Communications Entity*) e retornam a saída para a mesma SCE. A entidade remetente (que pode ser uma aplicação, uma outra entidade de camada superior, o WME ou qualquer outra entidade) utiliza os serviços de segurança para realizar o processamento de segurança do lado do remetente. Os resultados deste processamento são devolvidos para a entidade remetente, que, em seguida, transmite a Unidade de Dados do Protocolo de Aplicação (APDU) resultante para a entidade destinatária [IEEE 2013]. A entidade destinatária recebe a APDU e, em seguida, utiliza os serviços de segurança para realizar o processamento da segurança sobre o conteúdo da APDU. Estes retornam o resultado para a entidade remetente para processamento adicional que pode incluir várias chamadas dos serviços de segurança, caso necessário [IEEE 2013].

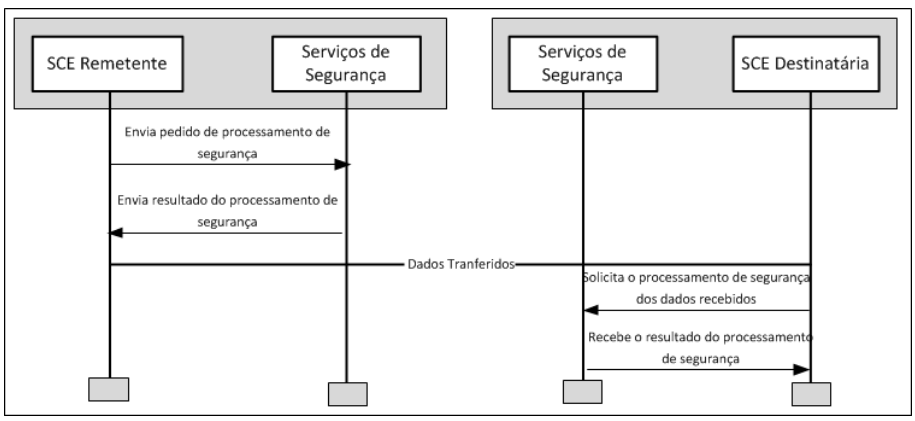



**Figura 4.9. Fluxo do Processamento nos Serviços de Segurança IEEE 1609.2 [IEEE 2013]**

Com a norma IEEE 1609.2, tem-se uma base criptográfica sólida para a concepção de sistemas de transportes inteligentes (ITS) seguros. Porém, isto dependerá dos fabricantes e fornecedores que precisam implementar este padrão em sua forma completa [Schütze 2011]. Além disso, de acordo com o [Schütze 2011], a implementação da norma IEEE 1609.2 em software não é uma solução muito realista, devido às limitações de desempenho. Além do problema de desempenho, os processadores automotivos atuais não têm proteção suficiente contra manipulações maliciosas. De acordo com o autor, os cartões inteligentes (*smartcards*) são uma alternativa por pelo menos duas razões: a maioria dos cartões inteligentes tem uma unidade de número longo aritmética que acelera as operações de chave pública consideravelmente e estes possuem proteção contra adulteração (*tamper resistance*). No passado, os cartões inteligentes não foram qualificados para as condições ambientais dos automóveis, mas as empresas de cartões recentemente começaram a produzir cartões especiais para uso em automóveis, por exemplo, os produtos Infineon SLI70 ou o SLM76 cartões especiais para M2M (*Machine to Machine*).

A norma IEEE 1609.2 especifica apenas os formatos e como ocorre os processamentos para prover segurança criptográfica às PDUs. Porém, a privacidade e o anonimato são questões consideradas fora do escopo, já que estas requerem atenção por parte dos desenvolvedores das diferentes partes de um dispositivo WAVE [IEEE 2013].

### 4.4.2. Mecanismos Criptográficos e Gerenciamento de Chaves

As aplicações de segurança (*safety*) suportadas por VANETs possibilitam a tomada de decisão por um motorista com base em mensagens recebidas de outros veículos. Contudo, se um veículo se comporta de modo malicioso, injetando ou alterando estas mensagens, este pode colocar o motorista em situação de perigo e, em alguns casos, com risco de morte [Wasef et al. 2010]. Desta forma, a autenticidade das mensagens é obrigatória para proteger as VANETs contra nós atacantes. Porém, caso as mensagens não contenham qualquer informação sensível, a confidencialidade não é necessária. Como resultado, a troca de mensagens de segurança em uma VANET precisa de autenticação, mas nem sempre de cifragem. Vários autores [Raya e Hubaux 2007, Mejri et al. 2014] escolheram as assinaturas digitais como solução para autenticação das mensagens. Neste caso, o método mais eficiente e mais simples é atribuir para cada veículo um par de chaves (privada/pública) permitindo ao veículo a possibilidade de assinar digitalmente as mensagens e, assim, autenticar-se perante os destinatários das mensagens. Diante da necessidade de responsabilização das aplicações em VANETs, a abordagem de gestão de confiança auto-organizada, como a do PGP (*Pretty Good Privacy*) não é satisfatória. As chaves públicas utilizadas em uma rede veicular devem ser emitidas e assinadas por uma autoridade confiável. A necessidade de certificados digitais emitidos por uma autoridade implica no uso de uma Infraestrutura de Chaves Públicas (ICP) veicular [Wasef et al. 2010].

O uso de uma ICP tem sido amplamente utilizado como solução para os problemas de segurança em VANETs [Raya et al. 2006b, Wasef et al. 2010]. Uma função importante de qualquer sistema ICP é a renovação e a revogação de certificados. Cada certificado emitido possui uma validade, indicando o período que ele pode ser utilizado por um determinado usuário. Antes que expire sua validade, uma autoridade certificadora (AC) pode executar o processo de renovação, emitindo um novo certificado para o usuário [Nowatkowski 2010]. A revogação de certificados é uma forma de terminar a associação

de um veículo com a rede, de modo que suas mensagens sejam ignoradas.

Uma AC também possui a responsabilidade de emitir e distribuir os certificados de chave pública. Estas responsabilidades são utilizadas para auxiliar na revogação de certificados inválidos. Segundo [Al Falasi e Barka 2011], existem muitas abordagens na literatura que lidam com a informação do estado do certificado, sendo uma das mais discutidas, a distribuição de uma lista de certificados revogados (CRL do Inglês *Certificate Revocation List*). Uma CRL é gerada pela AC utilizando uma chave privada para garantir sua autenticidade e contém a relação dos certificados revogados, a data da revogação e a data de geração da lista. Conforme [Papadimitratos et al. 2008, Al Falasi e Barka 2011], as CRLs são emitidas para revogar os certificados conforme decisões técnicas ou administrativas tomadas pela AC, como por exemplo, baseadas na mudança de proprietário, roubo, ou aluguel do veículo, entre outras.

Conforme [Al Falasi e Barka 2011], o esquema de revogação segue duas abordagens, a centralizada e descentralizada. Na abordagem centralizada, uma entidade central é a única responsável pela tomada de decisão sobre as revogações. De outro lado, na abordagem descentralizada, a decisão é tomada por um grupo de veículos na vizinhança (um salto de distância) do veículo que terá o certificado revogado. De acordo com [Aslam e Zou 2009], as soluções com base em AC centralizadas devem ser organizadas de forma hierárquica para prover um gerenciamento eficiente. A hierarquia pode ser baseada em áreas (países ou continentes), que por sua vez podem ser divididas em regiões (estados ou países). Cada região teria sua AC regional e estaria ligada às demais regiões através da AC de área (raiz), com base em uma relação de confiança para realizar o processo de verificação dos certificados. Em [Al Falasi e Barka 2011], os autores ressaltam o custo computacional desse processo de verificação perante uma grande cadeia de regiões, além de ser difícil, uma vez que a distribuição destas listas deve cobrir todas as regiões, os veículos se movem de uma região para outra. Para exemplificar, se uma AC revoga um determinado certificado, é preciso distribuir esta informação em sua região e também encaminhar à AC raiz para que o mesmo encaminhe a informação para todas as demais ACs, que por sua vez, precisam espalhar a informação em suas regiões. Diversos esquemas de revogação de certificados têm sido propostos na literatura baseados nesta abordagem centralizada, como em [Laberteaux et al. 2008, Aslam e Zou 2009, Nowatkowski 2010, Haas et al. 2011].

Em uma solução descentralizada de revogação de certificados, os veículos são responsáveis por tomar a decisão de excluir da rede um determinado veículo por mau comportamento. Uma AC pode ser utilizada para fazer a revogação, contudo, a decisão e a execução do processo de despejo são dos veículos. Este processo é conhecido como exclusão de nós (*node eviction*), e possui diversas fases, como a detectar o veículo com mau comportamento, relatar o mau comportamento, revogar do certificado (função das ACs) e disseminação da informação [Kherani e Rao 2010]. Existem algumas abordagens utilizando esta solução descentralizada, como por exemplo [Wasef e Shen 2009].

Uma forma alternativa para assinatura digital de mensagens em redes veiculares são os mecanismos de assinatura sem certificados, por exemplo, os que fazem uso de criptografia baseada em identidade (do inglês *Identity-Based Cryptography - IBC*) [Karagiannis et al. 2011, Silva et al. 2008]. O uso de criptografia baseada em identidade elimina a necessidade de verificar a validade dos certificados na tradicional infraestrutura de chave

pública (ICP). Nestes sistemas criptográficos, a chave pública de cada usuário é facilmente calculável a partir de uma sequência arbitrária correspondente à identidade do usuário (por exemplo, um endereço de e-mail, um número de telefone, etc). Os mecanismos de assinatura que fazem uso de IBC são muitos eficientes e recomendados para redes veiculares, pois uma entidade verificadora (por exemplo, uma unidade de bordo - OBU) não necessita armazenar, buscar e verificar os certificados de chaves públicas assinados por uma terceira autoridade confiável [Biswas et al. 2011].

De acordo com [Schleiffer et al. 2013], mecanismos de segurança com base em criptografia já estão sendo aplicados em veículos, em especial, para proteção contra furto, falsificação e atualização segura de softwares. Para proteção contra roubo e falsificação, mecanismos de criptografia que utilizam chaves simétricas ou assimétricas têm sido utilizados. Para atualização segura de softwares utiliza-se basicamente um par de chaves (pública/privada) baseadas em RSA. A chave pública é armazenada no veículo, e a privada em um servidor, que a utiliza para assinalar um *firmware* o qual posteriormente será verificado pelo veículo. A atualização segura de software é encontrada na maioria dos veículos, sendo utilizadas geralmente para sistemas de informação, entretenimento, segurança e em ECUs (*Eletric Control Unit*).

Nos trabalhos de [Schleiffer et al. 2013, Wolf e Gendrullis 2012] são apresentadas as principais iniciativas da indústria automobilística para prover mecanismos de segurança baseados em criptografia, a exemplo do consórcio HIS (*Hersteller Initiative Software*) da Alemanha, que especificou o SHE (*Secure Hardware Extension*). Através de um esquema básico de gerenciamento de chaves, SHE utiliza uma única chave simétrica por veículo, instalada em cada ECU na linha de montagem. Sua funcionalidade central é o armazenamento das chaves simétricas e as operações básicas (encriptação/decriptação) com estas chaves, a fim de prover a atualização segura de aplicações do veículo e outras operações. Outro exemplo é o projeto de pesquisa Europeu EVITA, o qual busca desenvolver três módulos de segurança, EVITA *light*, EVITA *medium* e EVITA *full*. O foco desta iniciativa é a segurança da comunicação V2V/V2I.

Segundo [Mejri et al. 2014], a criptografia moderna oferece diversas técnicas de segurança que atendem aos requisitos de confidencialidade, autenticidade, integridade, não-repúdio, entre outros (ver Seção 4.3.1), presentes nas VANETs, sendo utilizadas como contramedidas aos diferentes tipos ataques existentes nas VANETs:

- **Contra Negação de Serviço (DoS):** uso de mecanismos de autenticação baseados em assinatura e *bit commitment*;
- **Contra *Jamming*:** troca do canal de comunicação e utilização da técnica FHSS (*Frequency Hopping Spread Spectrum*), a qual envolve algoritmos criptográficos para gerar números pseudo-aleatórios para o algoritmo de salto;
- **Contra Ataques de Supressão de Mensagem:** uso de uma ICP veicular ou da técnica de *zero-knowledge*;
- **Contra Buraco Negro:** para estes tipos de ataques não existem soluções de criptografia reais, porém, o uso de hardwares confiáveis, assinatura digital em softwares e sensores podem minimizar os efeitos dos seguintes ataques;

- **Contra Temporização:** utilizar técnica de *timestamp* (baseada em assinatura digital e em funções *hash*) para controle de tempo em aplicações sensíveis ao atraso de pacotes;
- **Contra GPS *Spoofing*:** uso de mecanismos de *bit commitment*, juntamente com sistemas de posicionamento para aceitar somente dados de localização autênticos;
- **Contra Força Bruta:** uso de algoritmos de geração de chaves e encriptação fortes, que sejam inquebráveis dentro de um espaço de tempo razoável;
- **Contra *Sybil*:** reforçar os mecanismos de autenticação com uso de técnicas de criptografia como *bit commitment* e *zero-knowledge*;
- **Contra Replicação do Certificado ou Chave:** uso de chaves e certificados descartáveis; checagem da validade dos certificados digitais em tempo real através da lista de certificados revogados; uso de certificação cruzada entre diferentes ACs envolvidas no esquema de segurança.
- **Ataque de Mensagem Antiga (*Replay*):** as mensagens devem incluir o timestamp. Ele é utilizado pelo receptor para verificar se a mensagem já não está em cache e também para evitar este tipo de ataque. Este tipo de contramedida, que busca garantir a troca de mensagens seguras, está presente no IEEE P1609.2 [Laurendeau e Barbeau 2006].
- **Análise de Tráfego e *Eavesdropping*:** para estes tipos de ataques é necessário criptografar somente as mensagens importantes, as quais colocam em risco a privacidade do usuário, como dados de posicionamento (GPS), identificação do veículo, etc.

Em relação aos esquemas de comprometimento de bit (*bit commitment*) e conhecimento zero (*zero-knowledge*) citados pelo autor, o comprometimento de bit é uma das primitivas fundamentais da criptografia moderna, sendo uma ferramenta essencial na construção de protocolos criptográficos. Um problema comum em criptografia é uma parte $A$ obter provas de que uma outra parte $B$ é quem realmente afirma ser. Atualmente existem muitas formas de se provar uma identidade utilizando a criptografia, como uso de protocolos criptográficos, assinaturas digitais, esquemas que empregam a prova de conhecimento zero, etc. Nos esquemas de prova de conhecimento zero, uma parte $A$ não conhece os detalhes da mensagem da outra parte, mas utiliza determinados meios para verificar se a mensagem é de quem afirma ser (autenticidade) [Ribeiro et al. 2004, Mohr 2007]. Em [Ribeiro et al. 2004] são abordados os principais esquemas de chave pública que empregam a prova de conhecimento zero.

Em geral, um protocolo que usa o esquema de comprometimento de bit segue duas fases, $(i)$ o comprometimento e $(ii)$ a abertura, sendo estas executadas por duas partes, $A$ e $B$. Inicialmente, $A$ se compromete com um valor $v$, enviando determinadas informações para $B$ que contenha vestígios do valor $v$. De modo esperado, $B$ não deve ser capaz de descobrir o valor $v$ a partir dos vestígios do valor encaminhado ou entre as possíveis interações com $A$. Somente na fase de abertura, $A$ pode revelar a $B$ o valor $v$, que por sua vez pode verificar se o mesmo corresponde ao valor esperado com base nos vestígios enviados preliminarmente [Juels e Wattenberg 1999, Alves 2011, Pinto 2013].

### 4.4.3. Mecanismos de Autenticação e Técnicas de Anonimato

Dentre os desafios de segurança em redes veiculares, um dos mais atuais é prover autenticidade e não repúdio, além de preservar a privacidade dos dados durante uma comunicação [Mejri et al. 2014]. Estes requisitos são em alguns momentos conflitantes [Isaac et al. 2010]. Diante das inovadoras aplicações em redes veiculares, um atacante pode controlar um veículo, observando seus padrões de comunicação e movimento. O anonimato é uma preocupação crítica em VANETs e visa ocultar a identidade física de um nó (geralmente um veículo) [Isaac et al. 2010].

O anonimato em redes veiculares é utilizado para evitar ataques por rastreamento e impedir que entidades não autorizadas sejam capazes de localizar ou rastrear a trajetória de um veículo ou grupo de veículos. Em especial, uma entidade não autorizada não deve ser capaz de saber se mensagens diferentes foram criadas pelo mesmo nó. O anonimato dos veículos pode ser garantido pelo uso de pseudônimos que não indicam a identidade dos seus proprietários. Estes pseudônimos podem ser chaves recebidas por uma Autoridade Certificadora (AC), por unidades de acostamento (RSUs) ou que estejam instaladas nas unidades de bordo dos veículos [Chen et al. 2011]. Um problema ocorre quando um veículo totalmente anônimo transforma-se em um veículo malicioso. Neste caso, pode não haver maneira de identificá-lo para revogar o seu anonimato e puni-lo [Huang et al. 2014].

Os mecanismos de segurança devem fornecer privacidade (anonimato) para os veículos, porém também devem ser capazes de monitorar o comportamento destes veículos para que quando um veículo assuma um comportamento inadequado (malicioso), este possa ser identificado. Em outras palavras, os veículos em uma VANET precisam ter uma privacidade condicional. Ou seja, a privacidade dos veículos será garantida se eles se comportarem de forma adequada na rede. Caso contrário, a privacidade deve ser revogada e estes não permanecerão mais como anônimos [Huang et al. 2014, Chuang e Lee 2011, Xiong et al. 2013]. A seguir, são apresentados mecanismos e técnicas de anonimato em VANETs mais proeminentes na literatura.

#### PAAVE [Paruchuri e Durresi 2010]

O Protocolo para Autenticação Anônima em Redes Veiculares (do inglês, *Protocol for Anonymous Authentication in Vehicular Networks*) [Paruchuri e Durresi 2010] aborda a preservação da privacidade com rastreabilidade de autoridade usando cartões inteligentes (*smartcards*), que geram dinamicamente as chaves anônimas. O protocolo faz uso de um módulo de segurança veicular (VSM), em um cartão inteligente, para armazenar de forma segura as informações de identidade de um veículo, incluindo informações do condutor e todas as chaves criptográficas necessárias para comunicação com outros veículos e com as unidades de acostamento (RSUs). Os cartões VSM armazenam informações usadas para autenticar e identificar as OBUs pertencentes a rede veicular.

A Figura 4.10 ilustra a arquitetura proposta para comunicação e autenticação anônima. Toda comunicação a partir de uma OBU passa através do VSM para ser criptografada antes da transmissão. Qualquer mensagem recebida também é decriptografada pelo VSM. Isto ocorre pois é o cartão VSM que armazena as chaves criptográficas necessárias e que executa os protocolos criptográficos. A autoridade confiável da rede veicular (TA, do inglês *Trusted Authority*) é responsável por $(i)$ iniciar os VSMs, $(ii)$ manter o registro das

identidades dos membros da rede e $(iii)$ emitir os certificados das OBUs. O protocolo PAAVE compreende principalmente os seguintes elementos:

- **Mecanismo de Autenticação**: antes de enviar ou receber qualquer mensagem, cada OBU precisa se autenticar em uma RSU (protocolo de desafio-resposta baseado em criptografia de chave pública). A autenticação deve ocorrer mesmo quando a RSU estiver fora do raio de comunicação da OBU. Neste último caso, o processo será intermediado por outras OBUs.
- **Chaves de Sessão para Comunicação**: cada RSU gera uma nova chave de sessão no início de cada sessão. Esta chave de sessão é dada a todas as OBUs autenticadas pela RSU, assim, a chave de sessão pode ser vista como uma chave de grupo compartilhada por todas as OBU autenticadas pela mesma RSU. As OBUs autenticadas por diferentes RSUs receberão chaves de sessão diferentes. Para tratar deste problema, no início de cada sessão, cada RSU se comunica com RSUs vizinhas e obtém as suas chaves de sessão e os seus identificadores. Quando um nó (OBU) se registra em uma RSU, a RSU envia todas as chaves de sessão das RSUs vizinhas, juntamente com a sua própria chave de sessão e os identificadores de chave correspondentes. Cada vez que uma OBU recebe um aviso de uma nova RSU, esta obtém o novo conjunto de chaves de sessão da RSU.
- **Mecanismo de Comunicação e Verificação de Mensagens**: sempre que uma OBU tem que transmitir uma mensagem $m$, a mensagem é criptografada pelo módulo VSM com a chave de sessão compartilhada dentro do *cluster* de RSUs.

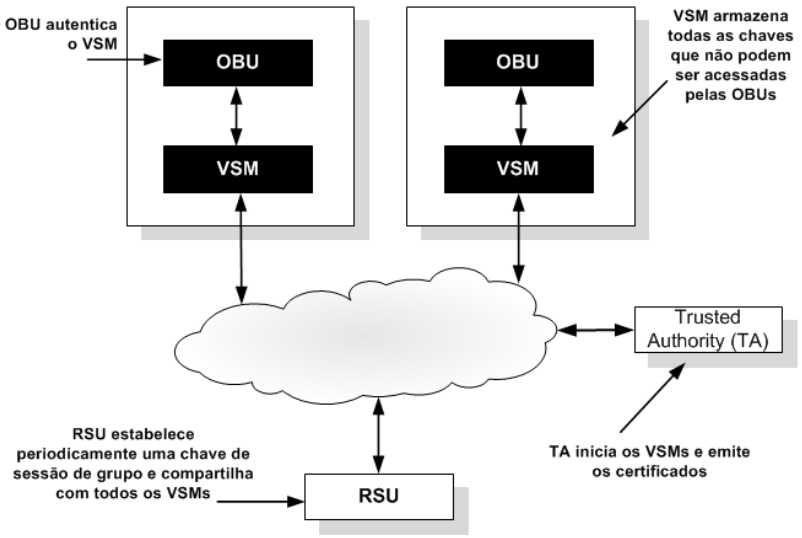



**Figura 4.10. Arquitetura: Comunicação e Autenticação Anônimas [Paruchuri e Durresi 2010]**

O PAAVE não faz uso de listas de certificados revogados (CRLs - *Certificate Revocation Lists*), pois esta técnica causa uma sobrecarga de armazenamento e de processamento significativa em cada OBU. No protocolo, como cada OBU precisa se autenticar em uma RSU para obter a chave de sessão, esta pode negar um pedido de OBUs maliciosas. Além disso, cada RSU compartilha informações sobre OBUs maliciosas com todas as

outras RSUs. Isto evita que a OBU obtenha uma chave de sessão de qualquer ponto da rede. Além disso, se a OBU está se movendo de uma RSU para outra, esta tem que se re-autenticar na nova RSU, assim os privilégios de comunicação da OBU podem ser revogados mesmo antes do fim da sessão.

De acordo com [Paruchuri e Durresi 2010], o protocolo PAAVE apresenta melhor eficiência em relação a propostas anteriores [Lin et al. 2007, Raya e Hubaux 2007, Lu et al. 2008], em termos de armazenamento de chaves anônimas, sobrecarga de comunicação e tempo computacional para verificar mensagens.

### AOSA [Weerasinghe et al. 2010]

As comunicações intermediadas por unidades de acostamento (RSUs) podem ser usadas para rastrear a localização dos veículos, sendo uma séria ameaça à privacidade dos usuários. O protocolo AOSA (*Anonymous Online Service Access*) [Weerasinghe et al. 2010] possibilita acesso anônimo aos serviços online com garantia de privacidade de localização, através de não rastreabilidade (*unlinkability)*. No protocolo AOSA, os autores consideram um modelo de rede veicular segura composta por unidades de bordo (OBUs), unidades de acostamento (RSUs) e servidores de aplicação (administrativos), os quais possibilitam comunicação V2V e V2I. As RSUs estão fisicamente conectadas à infraestrutura da VANET por meio de uma rede cabeada e são gerenciadas por uma autoridade confiável (p.ex. o Departamento de Trânsito). Além disso, uma autoridade de registro confiável (AR) fornece serviços de registro aos veículos. Todos os veículos devem se registrar na AR antes de ingressar na VANET. Uma Infraestrutura de Chaves Públicas (ICP) é implementada na rede veicular e a AR também pode funcionar com uma Autoridade Certificadora (AC) para gerenciar as chaves e os certificados dos veículos. A confiança entre as entidades é mantida, usando chaves e certificados emitidos pela AC confiável.

Para comunicação anônima segura, cada veículo usa chaves privadas/públicas anônimas, chamadas pseudo chaves, com um certificado de chave pública assinado pela AC. Estas pseudo chaves e os certificados são alterados com frequência para manter a privacidade de localização. Pode-se usar um grande número de pares de chaves e certificados pré-carregados ou adquirir frequentemente pseudo chaves e certificados de curta validade junto às RSUs. Em conformidade com a norma IEEE 1609.2, cada mensagem V2V e V2I deve conter a assinatura da mensagem e o certificado de chave pública. Portanto, os veículos usam a pseudo chave corrente para assinar e verificar as assinaturas, fazendo uso do certificado anexado e da chave pública da AC.

A execução do protocolo segue duas fases. Na **primeira fase**, todos os veículos e prestadores de serviços devem se registrar junto à AR. Os veículos que desejam usar serviços online devem também se inscrever para o serviço requirido por meio da AR. Quando AR/AC emite as pseudo chaves públicas/privadas, o certificado de chave pública deve incluir informações sobre todos os serviços registrados com uma assinatura cega (*blind signature*) de cada provedor de serviços. Os veículos só podem usar a parte específica do certificado para o pedido de serviço específico. Cada informação de serviço é criptografada com a chave pública do provedor de serviços, para que cada provedor possa acessar apenas às suas próprias informações.

No protocolo AOSA, os veículos formam grupos dinamicamente, e pequenas

assinaturas de grupo são usadas para lidar com todas as chaves e assinaturas do grupo [Boneh et al. 2004]. A viabilidade do uso de chaves de grupos em cenários VANET está sendo usada e avaliada em diversas pesquisas [Raya e Hubaux 2007, Lin et al. 2007]. Todos os membros de um grupo compartilham uma chave pública de grupo e cada veículo membro tem uma chave secreta única que pode ser utilizada com a chave pública do grupo comum. Além disso, todos os membros compartilham um conjunto de identificadores temporários comuns. Duas assinaturas de um mesmo veículo não podem ser ligadas entre si. No entanto, o líder do grupo e a autoridade de registro podem colaborar para descobrir a verdadeira identidade do assinante da mensagem.

Na **segunda fase**, quando um veículo necessita acessar um serviço, este envia uma solicitação de acesso ao serviço por meio do líder do grupo. A mensagem de pedido deve ser assinada pelo veículo, usando seu pseudônimo atual e o certificado de chave pública emitido pela AC deve ser incluído. Esta mensagem de pedido é primeiro criptografado com a chave pública do prestador de serviços e então é criptografado com a chave secreta do grupo pelo veículo fonte. A mensagem é então enviada para o líder do grupo. O líder do grupo decifra a mensagem e adiciona a sua assinatura e o certificado de chave pública do grupo e encaminha a nova mensagem para um servidor *proxy* através da RSU. O servidor *proxy* verifica o certificado do líder do grupo e encaminha o pedido ao provedor de serviço solicitado. O servidor *proxy* também mantém um registro da localização da RSU que encaminhou a mensagem para fins de resposta. Depois de receber o pedido de serviço, o provedor de serviços decifra o pedido com a sua chave privada e, em seguida, atesta as credenciais anônimas do veículo, usando o certificado do serviço e a chave pública da AC. Finalmente, o provedor de serviços verifica a autorização do veículo para o serviço. O provedor de serviços envia uma chave de sessão para compartilhar com o veículo. Esta mensagem é primeiro criptografada com a chave pública anônima do veículo e, em seguida, encriptada com a chave pública do líder do grupo.

No protocolo AOSA, para comunicação anônima segura entre veículos e provedores de serviços, observa-se o uso de diversas operações de cifragem e de assinatura baseadas em criptografia assimétrica que acarretam sobrecarga computacional que, dependendo da aplicação, pode não ser desprezível. Nas simulações realizadas, os autores não compararam os impactos (sobrecarga) do protocolo AOSA em relação aos outros procolos que também garantem a comunicação anônima.

### TEAM [Chuang e Lee 2011]

De acordo com os autores em [Chuang e Lee 2011], os esquemas de autenticação seguras suportados por criptografia assimétrica não são adequados para ambientes altamente dinâmicos como as VANETs, pois estes esquemas não lidam com o processo de autenticação de forma eficiente. Diante desta limitação, os autores propuseram um esquema de autenticação leve, descentralizado, chamado TEAM (*Trust-Extended Authentication Mechanism*) para comunicação V2V. O mecanismo é leve porque usa apenas operações XOR e uma função *hash*. O mecanismo TEAM adota ainda o conceito de relações de confiança transitiva para melhorar o desempenho do processo de autenticação. O mecanismo proposto satisfaz os seguintes requisitos de segurança: anonimato, privacidade de localização e autenticidade.

No modelo de rede assumido pelos autores, os veículos podem ser classificados com os seguintes papéis (ver Figura 4.11): i) Executor da Lei (VL), como um carro de

polícia que funciona como um servidor de autenticação móvel (SA); ii) Veículo Não Confiável (VNC); e iii) Veículo Confiável (VC). Os autores assumem como premissas que cada unidade de bordo (OBU) dos veículos é equipada com um hardware de segurança, que inclui um *Tamper Proof Device - TPD* para prover o processamento criptográfico das informações e um *Event Data Recorder - EDR*, e que o VL é um veículo confiável. Um veículo é considerado confiável se este puder autenticar-se com sucesso, caso contrário, ele é considerado não confiável. Em um ambiente de comunicação segura, uma OBU deve-se autenticar com sucesso antes de acessar um serviço. Entretanto, nas redes de comunicação V2V, como o número de veículos executores da lei (VLs) é finito, uma OBU nem sempre tem um VL em sua vizinhança. Os autores se baseiam no conceito de relações de confiança transitiva para tratar esta questão, conforme ilustrado na Figura 4.12.

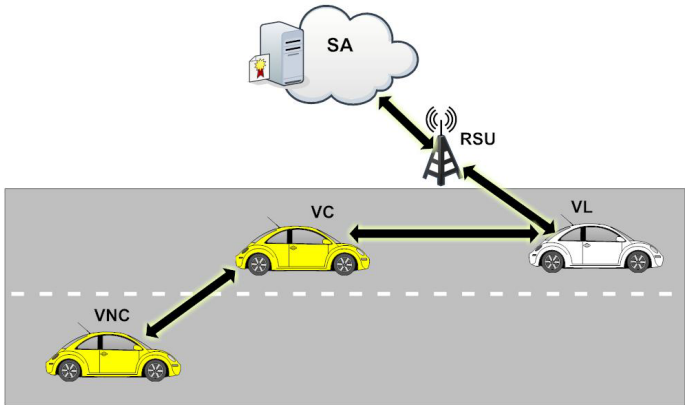



**Figura 4.11. Modelo de Rede Veicular [Chuang e Lee 2011]**

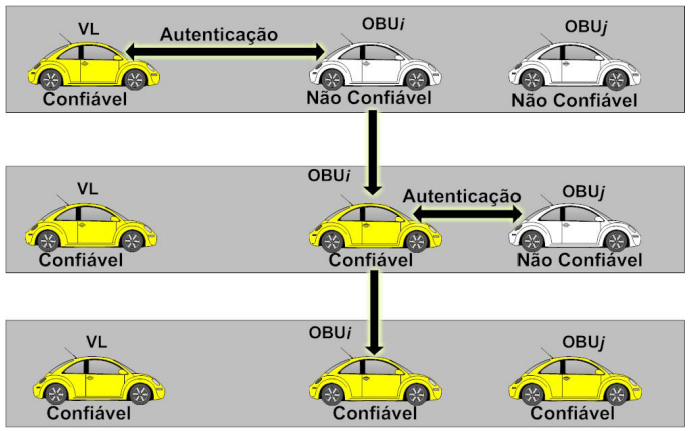



**Figura 4.12. Relação de Confiança Transitiva no TEAM [Chuang e Lee 2011]**

Na Figura 4.12, inicialmente, existem três veículos na rede: um VL confiável e dois outros veículos que não são confiáveis ($OBU_i$ e $OBU_j$). A $OBU_i$ torna-se confiável quando esta autentica-se no VL. Neste momento, $OBU_i$ recebe a autorização para autenticar outras OBUs. Em seguida, como o VL não está acessível para a $OBU_j$, esta pode se autenticar diretamente na $OBU_i$ (já que esta, após a autenticação, assume o papel de VL temporário). Como resultado, todos os veículos de uma VANET podem concluir o processo de autenticação rapidamente, fazendo uso de relações de confiança transitivas.

O mecanismo TEAM é um esquema de autenticação descentralizado, logo um veículo executor da lei (VL) não necessita manter informações de autenticação de todos os veículos. As principais operações do mecanismo são: registro inicial, login, autenticação geral e procedimentos de autenticação de confiança estendida. Antes de um veículo poder participar de uma rede veicular, sua OBU deve-se registrar no servidor de autenticação (AS). Quando um usuário quer acessar um serviço, este deve realizar o login na OBU do veículo e então passa pelos procedimentos de autenticação geral. É importante destacar que os passos para autenticação geral (entre OBU e um VL) e para autenticação de confiança estendida (entre OBU não confiável e OBU confiável) são os mesmos.

Os autores não tratam o problema de nós que, após autenticados, passam a se comportar de forma maliciosa na rede e que por isto precisam ter o seu anonimato revogado. Ataques de negação de autenticação estendida também podem ser executados por estes nós maliciosos. Devidos às relações de confiança transitivas, conluios podem ser facilmente formados para atacar a rede.

### LPP [Shen et al. 2012]

No LPP (*Lightweight Privacy-Preserving Protocol*), o mesmo modelo de rede veicular segura adotado no projeto do protocolo AOSA é assumido como premissa. O LPP provê autenticação mútua entre OBUs e RSUs e faz uso de assinatura de *hash chameleon* baseada em curvas elípticas. No esquema de assinatura proposto pelos autores, a chave pública é atualizada a cada sessão de autenticação (chaves públicas dinâmicas). O protocolo segue três fases: registro; autenticação mútua e rastreamento da Autoridade Certificadora (AC).

Na **fase de registro**, as RSUs e OBUs devem se cadastrar junto a uma AC e efetuar uma pré-carga com informações secretas. Na fase de registro de uma OBU, esta gera um número aleatório como sua chave secreta e a envia juntamente com parâmetros de inicialização do algoritmo de assinatura *chamelon* e a sua identidade real para a AC. A AC gera um certificado com um tempo de expiração e o assina. Este certificado e o identificador da OBU são armazenados na base de dados da AC e enviados à OBU.

Antes de efetuar alguma troca de mensagem, uma RSU e uma OBU devem se **autenticar mutuamente** com as informações pré-carregadas na fase de registro. Nesta fase, a RSU é quem incia a autenticação com a OBU e então estas estabelecem um par de chaves. A RSU gera uma nova chave privada com a chave pública correspondente, de modo a evitar a rastreabilidade. Finalmente, a informação é enviada para a OBU. Ao receber essa informação, a OBU usa a chave pública da AC para verificar a legitimidade da RSU. Caso ocorra um evento de disputa, a AC executa a **fase de rastreamento** para recuperar a identidade real da OBU. Para isto, a RSU deve enviar o certificado da OBU para a AC, para que a identidade da OBU possa ser encontrada no banco de dados da AC.

O protocolo LPP garante o anonimato e a não rastreabilidade devido às seguintes propriedades: (i) as informações enviadas pelas OBUs usam diferentes chaves públicas de diferentes sessões, desta forma não é possível rastrear um veículo; (ii) as informações relacionadas com os certificados são criptografadas utilizando o par de chaves da sessão, de tal forma que o certificado verdadeiro não possa ser extraído; e (iii) o par de chaves é atualizado em todas as sessões.

### PA-CTM [Amro et al. 2013]

De acordo com [Amro et al. 2013], os sistemas de monitoramento de tráfego colaborativo (CTM, *Collabortative Traffic-Monitoring*) são divididos em duas abordagens de acordo com a tecnologia de comunicação de dados subjacente. A primeira é baseada no uso de comunicação de curto alcance dedicada (DSRC) que suporta comunicações V2V e V2I, chamados de sistemas de infraestrutura dedicada (DI - *Dedicated Infrastructure*). A segunda se baseia no uso de tecnologias existentes, como tecnologias de comunicação celulares e outras utilizadas em redes sem fio locais, a fim de criar um sistema de transporte inteligente (ITS). Os sistemas aplicando esta abordagem, chamados de sistemas de infraestrutura existente (EI - *Existing Infrastructure*), seguem uma arquitetura cliente servidor, na qual os clientes enviam suas informações de localização para um servidor, podendo obter uma visão geral do tráfego a partir deste servidor.

O sistema PA-CTM (*Privacy Aware Collaborative Traffic Monitoring System*) [Amro et al. 2013], suportado pela abordagem EI, possibilita aos usuários autenticar-se em servidores de monitoramento de tráfego de forma anônima utilizando pseudônimos. O sistema também permite revelar as identidades por propósitos de força de lei, quando necessário. Os usuários são capazes de mudar seus pseudônimos e, portanto, ocultar suas informações de trajetória completa no servidor de tráfego. O sistema usa um mecanismo autônomo de atualização de localização (ALUM - *Autonomous Location Update Mechanism*) não dependente de uma terceira parte confiável, além de usar apenas parâmetros locais (velocidade e direção) para iniciar uma atualização de localização ou de pseudônimo.

A Figura 4.13 ilustra a arquitetura do PA-CTM. No passo 1, um veículo gera um número de identidades temporárias (pseudônimos) e envia para o sistema CoRPPS (*Collusion Resistant Pseudonym Providing System*) que é um sistema confiável resistente a ataques de conluio. O CoRPPS é composto por três unidades funcionais: i) unidade de registro; ii) unidade de autenticação; e iii) unidade de assinatura de pseudônimos, que cooperam para assinar os pseudônimos dos usuários (passo 2). No passo 3, o veículo se autentica no servidor de tráfego, usando um dos pseudônimos assinados. Os usuários são capazes de alterar os pseudônimos de tempos em tempos e então dividir sua trajetória real em pequenas trajetórias identificadas por seus pseudônimos. No passo 4, o servidor de tráfego verifica a assinatura do pseudônimo recebido e responde para o veículo com o

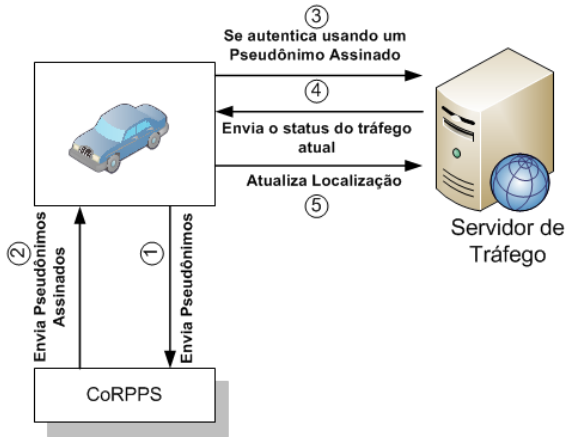



**Figura 4.13. Arquitetura e Fluxo do PA-CTM [Amro et al. 2013]**

status do tráfego atual. Por fim, no passo 5, o veículo atualiza sua localização, utilizando apenas parâmetros locais (velocidade e direção).

Esses pseudônimos são gerados através do sistema CoRPPS (*Collusion Resistant Pseudonym Providing System*). O CoRPPS proporciona confiança entre todas as partes do sistema e é resistente contra ataques de conluio. Os usuários podem assinar seu primeiro pseudônimo e, assim, usar os pseudônimos assinados para autenticar-se no servidor.

**Protocolo para Preservação de Privacidade e Confidencialidade [Xiong et al. 2013]**

Em [Xiong et al. 2013], os autores apresentam um protocolo eficiente para preservação da privacidade e confidencialidade para redes veiculares baseado em *signcryption* de grupo, uma primitiva de chave pública que desempenha simultaneamente as funções de assinatura digital e cifragem. O modelo proposto apresenta as seguintes características: (i) oferece autenticação anônima condicional, na qual o emissor da mensagem pode autenticar-se anonimamente em nome de um grupo de assinantes (veículos), enquanto apenas uma autoridade confiável pode revelar a verdadeira identidade do remetente; (ii) oferece confidencialidade aos motoristas contra observadores não autorizados durante a comunicação; e (iii) é eficiente, uma vez que não necessita de um grande espaço para armazenar os dados do protocolo em cada veículo, pois a verificação de mensagens é rápida, e o rastreamento da identidade de um veículo tem um custo baixo.

Conforme ilustrado na Figura 4.14, o modelo de sistema seguro assumido no trabalho é composto pelo Gerente Membro (MM) (do inglês, *Member Manager*) e as unidades de bordo (OBUs) instaladas nos veículos. A solução não depende de unidades de acostamento (RSUs). Antes dos veículos (OBUs) se conectarem em uma rede veicular estes precisam pré-carregar os parâmetros públicos do sistema MM e gerar suas próprias chaves privadas que serão armazenadas em um TPD (*tamper-proof device*) do veículo. Estes veículos irão registra-se no MM como membros de um grupo (p.ex. envio). O MM é o responsável por registrar todas as OBUs instaladas nos veículos e por revelar a identidade real de um emissor de uma mensagem sempre que necessário. Por fim, um MM é uma entidade confiável equipada com ampla capacidade de armazenamento e processamento.

A Figura 4.14 ilustra a operação do protocolo. Quando um veículo quer enviar mensagens na rede veicular, após a fase de registro no MM como membro de um grupo de envio, o mesmo deve assinar e cifrar (*signcryption*) a mensagem em nome do seu grupo e transmiti-la em *broadcast*. Além disso, o grupo de recebimento deverá ser indicado quando a mensagem for cifrada. O protocolo de comunicação proposto está baseado no esquema de assinatura de grupo cifrada proposto em [Kwak et al. 2006].

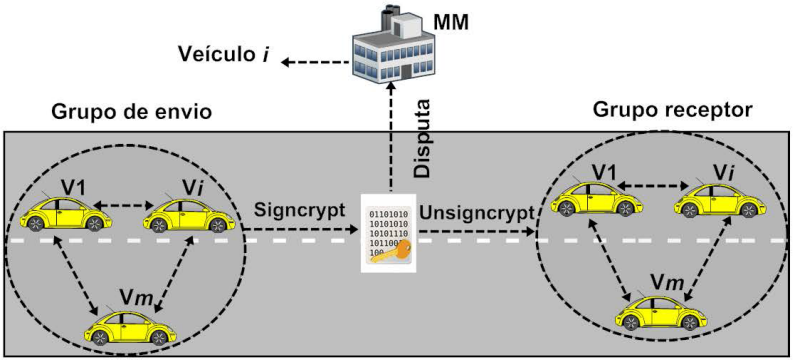



**Figura 4.14. Operação do Protocolo Proposto por [Xiong et al. 2013]**

### PPSCP [Mikki et al. 2013]

Em [Mikki et al. 2013], os autores propuseram o protocolo PPSCP (*Privacy Preserving Secure Communication Protocol*) para garantir a privacidade dos veículos em redes veiculares. Os objetivos do PPSCP incluem: autenticar mensagens de segurança de trânsito anonimamente para garantir a privacidade do veículo evitando assim o rastreamento do mesmo sem autorização; fornecer uma autoridade confiável (AC), capaz de identificar e reconhecer a identidade do veículo a partir de suas mensagens de segurança o que possibilita a rastreabilidade dos veículos; revogar chaves de veículos de forma eficiente, reduzindo o tamanho das listas de revogação; proteger contra ataques de negação de serviço (DoS) e *replay*; aumentar a eficiência do sistema, diminuindo o tempo necessário para autenticação dos veículos e verificação de mensagens dos veículos.

Cada veículo é equipado com o TPD (*Tamper-Proof Device*) que armazena as chaves criptográficas geradas pela AC. Além disso, o TPD executa todas as operações criptográficas necessárias pelo protocolo PPSCP. A identidade do veículo emissor de uma mensagem é cifrada com a chave pública da AC e somente esta pode revelar a identidade do veículo. A identidade cifrada, que é incluída na mensagem de segurança de trânsito, não está relacionada com a mensagem em si e pode ser pre-inicializada para reduzir o tempo. O PPSCP é composto por vários algoritmos. Parte dos algoritmos é usada na comunicação V2V, como por exemplo para proteger mensagens de segurança de trânsito (localização do veículo, velocidade e aceleração) e para verificar as mensagens recebidas. Outra parte dos algoritmos é aplicada na comunicação V2I, quando veículos solicitam um conjunto de chaves compartilhadas.

O gerenciamento de chaves no PPSCP é dado da seguinte forma: cada veículo ($n$) tem uma identidade única ($VIDn$) de 64bits. A AC é responsável por gerar e instalar as $VIDn$ em cada veículo (no TPD). A AC também gera um par de chaves pública e privada ($PubN$, $PrivN$) para cada veículo. Além do $VIDn$, a AC pré-instala o par de chaves do veículo e também a sua chave pública ($PubCA$) no dispositivo TPD de cada veículo. Para garantir o anonimato dos veículos, AC gera periodicamente um conjunto de chaves chamado de "conjunto de chaves compartilhadas", que contém $N$ chaves simétricas. Estas chaves são usadas para autenticação de mensagens entre os veículos. Cada veículo solicita este conjunto de chaves a AC, enviando uma mensagem criptografada com a chave pública a AC. A AC responde com uma mensagem que contém o conjunto de chaves compartilhadas. Essas mensagens são então criptografadas com a chave do veículo.

A autenticação de mensagens é feita com o conjunto de chaves simétricas compartilhadas com todos os veículos, usando código de autenticação de mensagem (*Message Authentication Code - MAC*) de 128 bits. Este conjunto de chaves é gerado e distribuído pela AC , por meio das RSUs. Cada chave do conjunto tem uma validade pré-definida. Quando uma chave expira, a próxima chave do conjunto é usada por todos os veículos. No PPSCP, o conjunto de chaves tem tamanho 4 e o período de duração de uma chave é de uma semana. O algoritmo MAC utiliza chave secreta com tamanho de 128bits. Esse comprimento de chave faz com que um ataque de força bruta seja impraticável.

Quando um mau comportamento de um veículo é detectado, o mesmo deve ser revogado. O esquema de revogação proposto depende de chaves simétricas chamadas de chaves de revogação. Cada veículo possui chaves simétricas de revogação que são

usadas para criptografar o *timestamp*. Quando um veículo precisa enviar uma mensagem de segurança, este adiciona o *timestamp* cifrado com esta chave. Quando um veículo recebe uma mensagem, este tenta decifrar o *timestamp* com todas as chaves da lista de revogação. A AC é responsável por manter e distribuir as listas de revogação. É possível reduzir o tamanho das listas de revogação, incluindo as chaves dos veículos revogados em vez de todos os pseudônimos ou todos os certificados. Quando a vida útil da chave de revogação expira, a mesma é removida da lista, e desta forma a lista se mantém pequena.

A mobilidade dos veículos entre diferentes cidades e países é uma situação frequente. Por isso, no PPSCP, os veículos são geridos por diferentes ACs com base em sua localização, sendo que estas ACs estão interconectadas. Quando um veículo N entra na região de uma nova AC, este se comunica com a AC usando sua chave pública PubN e o certificado. A nova AC verifica o certificado do veículo através da AC anterior e então gera e envia um novo conjunto de chaves para o veículo.

### PACP [Huang et al. 2014]

De acordo com [Huang et al. 2014], muitos esquemas projetados para manter o anonimato em VANETs utilizam uma infraestrutura de chave pública baseada no protocolo RSA ou em criptosistemas de curvas elípticas (ECC). No entanto, segundo os autores, estes esquemas sofrem de uma desvantagem comum, as autoridades envolvidas no processo de geração pseudônimos conhecem os pseudônimos utilizados pelos veículos. Logo, estes esquemas não são verdadeiramente anônimos.

O PACP (*Pseudonymous Authentication With Conditional Privacy*) [Huang et al. 2014] permite que veículos em uma rede veicular usem pseudônimos ao invés de sua identidade real. Neste protocolo, os veículos interagem com as unidades de acostamento (RSUs) para gerar os pseudônimos para comunicação, sendo que estes são conhecidos apenas pelos veículos. O esquema provê ainda um mecanismo de revogação que permite que veículos com mau comportamento sejam identificados e revogados da rede se necessário. Logo, a privacidade condicional é garantida aos veículos até que estes sejam revogados (estes deixam de ser anônimos). PACP se baseia em uma estrutura matemática baseada em em um esquema ECC (*pairing*). O protocolo não necessita armazenar vários certificados pseudônimos emitidos por uma autoridade confiável ou fornecidos por uma RSU. Em vez disso, um veículo gera seus pseudônimos com a ajuda da RSU vizinha.

O modelo de rede assumido no sistema PACP define três tipos de entidades, a saber: veículos, a MVD (*Motor Vehicles Division*) e RSUs. A interação entre estas três entidades no protocolo PACP está representada na Figura 4.15. Um veículo fornece a MVD as informações de identidade requeridas como parte do processo de registro. Em seguida, a MVD emite um ticket para o veículo. O ticket identifica unicamente o veículo, no entanto, este não revela a verdadeira identidade do veículo. Ao mover-se sobre a estrada, o veículo se autentica na RSU mais próxima e obtém um *token* de pseudônimos. O veículo usa o *token* para gerar seus pseudônimos. No protocolo, a RSU só fornece a credencial (ou seja, a assinatura) e restrições (ou seja, um *timestamp*) para que o veículo possa gerar seus pseudônimos e não detém qualquer informação privada do veículo.

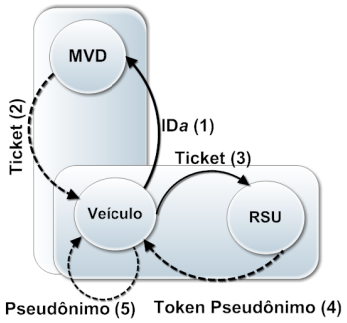



**Figura 4.15. Diagrama de Interação das Entidades envolvidas no PACP [Huang et al. 2014]**

### 4.4.4. Gerenciamento de Confiança e Sistemas de Reputação

Devido às suas características, as redes veiculares estão propensas à presença de nós maliciosos[4] [Raya et al. 2006a]. Os ataques ativos, provenientes desses nós, precisam ser evitados, já que estes podem enviar informações falsas na rede, bem como desviar pacotes, modificar seu conteúdo e até mesmo injetar novas mensagens na rede. Logo, torna-se necessário o desenvolvimento de soluções capazes de incentivar comportamentos cooperativos, mas que identifiquem a presença de nós maliciosos [Dai et al. 2013].

Segundo [Li et al. 2012], detectar nós maliciosos tornou-se um dos problemas mais difíceis no que diz respeito a segurança em VANETs. Minimizar os ataques e as consequências de comportamentos maliciosos é muito importante em soluções que necessitam da cooperação e da honestidade dos nós, tais como as aplicações de segurança no trânsito (p.ex. disseminação de alertas). Para minimizar a ação de nós maliciosos, alguns métodos surgiram visando privilegiar os nós com comportamento correto na rede, dentre estes destacam-se os que utilizam **sistemas de reputação** [Li et al. 2013].

O conceito de reputação pode ser definido como uma medida coletiva de confiabilidade em um dispositivo baseado em indicações ou avaliações de membros de uma comunidade. Cada nó possui um valor de reputação que reflete o seu comportamento. De acordo com [Huang et al. 2014], um sistema de reputação tem por finalidade construir um valor de confiança para cada nó na rede. Essas opiniões são acertadas para formar a reputação que serve como referência para que outros nós possam identificar quais nós podem oferecer recursos confiáveis (confiança no nó). Assim, o nível individual de confiança em um dispositivo pode ser obtido a partir de uma combinação das indicações recebidas de outros dispositivos [Swamynathan et al. 2007].

Alguns sistemas de reputação para redes veiculares, ao invés de focar na confiabilidade dos nós, focam na confiabilidade dos dados [Ostermaier et al. 2007, Lo e Tsai 2009]. De acordo com [Karagiannis et al. 2011], para algumas aplicações veiculares, a confiabilidade dos dados é mais útil que a confiabilidade dos nós que estão se comunicando, para outras a confiabilidade dos nós deve ser provida. A seguir, serão analisados alguns trabalhos que descrevem sistemas de reputação específicos para VANETs (tratam da confiabilidade dos nós ou da confiabilidade dos dados).

[4]Nesta seção, utiliza-se o termo malicioso mesmo quando a ação maliciosa visa um ganho pessoal (chamado de racional).

**[Ostermaier et al. 2007]**

Em [Ostermaier et al. 2007], os autores propuseram um sistema de reputação baseado em votação visando avaliar a credibilidade das mensagens (eventos) e aumentar a segurança das decisões tomadas pelos veículos sobre eventos reportados em aplicações LDW. O trabalho avaliou o resultado de quatro métodos de decisão da confiabilidade no perigo relatado com base no sistema de votação como segue:

- Últimas Mensagens: sempre que uma decisão precisa ser tomada, apenas a mensagem mais recente do alerta é considerada, cujo objetivo é atingir uma alta adaptabilidade em cenários livres de ataques, resultando em poucas decisões erradas.
- Maioria de vitórias: executa uma decisão local de voto sobre todas as mensagens recebidas sobre um determinado alerta. Caso a maioria das mensagens recebidas forem de alertas, uma decisão positiva é tomada; caso contrário, uma decisão negativa é considerada.
- Maioria das Últimas Mensagens: é uma combinação dos dois anteriores. Para uma tomada de decisão, um veículo irá realizar uma votação considerando apenas as últimas mensagens em relação ao alerta em questão.
- Maioria das Últimas $x$ Mensagens com valor mínimo: é uma extensão do anterior, na qual o veículo utiliza apenas as últimas $x$ mensagens recebidas com informações sobre o evento. Desta forma, é verificado um limite inferior, de forma que o mecanismo somente é utilizado caso o veículo receba ao menos um determinado número de mensagens. Quando esse mínimo de opiniões não é atingido, o veículo sempre se decide pela negação do evento.

O sistema de reputação proposto trata da credibilidade das mensagens. O custo computacional de processamento e a sobrecarga na rede decorrentes do uso deste mecanismo não foram avaliados pelos autores. As simulações realizadas comparam a eficácia dos quatro métodos de decisão com cenários livres de ataques [Fernandes et al. 2013].

**DTT [Wang e Chigan 2007]**

O mecanismo de confiança *Dynamic Trust-Token (DTT)* tem o objetivo de detectar a modificação de mensagens na rede por nós maliciosos e isolar estes nós de forma a prevenir que estes interfiram nas próximas mensagens. No mecanismo proposto pelos autores, são utilizadas técnicas de criptografia assimétrica e assinatura digital com o objetivo de garantir a integridade dos pacotes durante a comunicação. Quando um nó viola esta integridade, este é considerado malicioso.

No DTT, um emissor envia uma mensagem para seus vizinhos e caso este vizinho não seja o destinatário, este irá reencaminhar mensagem recebida. O emissor então monitora essas retransmissões e, caso o pacote não sofra nenhuma alteração, o emissor envia um *token* de confiança, assinado digitalmente, para o respectivo vizinho. Os vizinhos devem reencaminhar este *token*, que os certifica como confiáveis, para os veículos responsáveis pelo próximo salto. O processo de escuta e emissão do *token*, feito pelo emissor da mensagem, é agora realizado pelos vizinhos certificados. Esse ciclo se repete em todos os saltos

até que a mensagem atinja seu destino. O mecanismo baseia-se apenas no comportamento dos veículos em tempo de execução, definindo desta maneira a reputação instantânea de um nós não mantendo portanto reputação histórica. Este mecanismo são trata do problema de nós que propagam mensagens falsas na rede.

**RMDTV [de Paula et al. 2010]**

No RMDTV (*Reputation Mechanism for Delay Tolerant Vehicular Networks*), os membros da rede qualificam as informações (corretas ou não) dos outros membros e emitem mensagens de qualificação que atestam a confiabilidade da mensagem (informação correta recebida). O emissor da mensagem armazena estas mensagens de qualificação e as usa quando forem propagar novas mensagens como se estas fossem suas credenciais que comprovam as mensagens corretas já propagadas na rede. Ou seja, o sistema faz uso de qualificações emitidas por terceiros (reputação global) para atestar a confiabilidade dos nós, porém estas qualificações são apresentadas pelos próprios nós emissores do alerta. Desta maneira, os membros da rede podem verificar previamente a confiabilidade de novos vizinhos, antes mesmo da troca de dados.

O mecanismo usa o conceito de redes tolerantes a atrasos e interrupções (*Delay and Disruption Tolerant Networks - DTNs*) para sanar os possíveis problemas de momento de desconexão total, visto que o nó armazena as mensagens recebidas até poder encaminhá-las a outros nós da rede. Cada veículo é responsável por armazenar localmente duas listas contendo os membros considerados confiáveis e os membros maliciosos. Desta forma, um emissor de um alerta pode ser classificado como: malicioso, confiável ou desconhecido. Os nós considerados confiáveis são aqueles aos quais suas mensagens informam o evento corretamente, ao contrário, quando estes eventos são incorretos o nó é punido e passa a ser considerado malicioso. Segundo os autores, o reconhecimento prévio e a exclusão de dados gerados por nós maliciosos é possível devido ao armazenamento de dados históricos sobre o comportamento dos veículos. Além disso, o compartilhamento de experiências permite estabelecer relações de confiança antes do início das transações.

As qualificações possuem pesos diferenciados no mecanismo de decisão e estas qualificações são adicionadas às mensagens de dados geradas pelo veículo. Entretanto, segundo os autores do mecanismo RMDTV, para evitar uma grande sobrecarga na rede, eles consideram que apenas um determinado número de qualificações deve ser adicionado e estas possuem prazo de validade. Desta forma, somente aquelas não expiradas devem ser utilizadas. Neste mecanismo, não existe um rebaixamento progressivo dos nós na rede. Desta forma, basta que o veículo apresente um comportamento malicioso uma única vez, para que este possa ser considerado 100% malicioso. Da mesma maneira, basta que envie uma mensagem de qualificação correta para ser considerado 100% confiável.

**DMV [Daeinabi e Rahbar 2013]**

O mecanismo DMV (*Detection of Malicious Vehicles*) [Daeinabi e Rahbar 2013] visa monitorar nós maliciosos que rejeitam ou duplicam pacotes recebidos de forma a isolá-los dos nós considerados honestos. Cada veículo é monitorado por vizinhos confiáveis chamados de nós verificadores. Um conjunto de veículos está localizado em um agrupamento e cada agrupamento possui um líder. Cada veículo possui duas listas: lista branca e lista negra. Os veículos que compõem a lista branca são aqueles cujos valores de desconfiança ($Td$)

são inferiores a um *threshold* mínimo. Por outro lado, a lista negra contém os veículos na qual seu valor de desconfiança ($Td$) é mais elevados que o *threshold* mínimo. Cada veículo, ao entrar na rede pela primeira vez, tem seu valor de desconfiança igual a um (valor igual para todos os veículos) e está presente na lista branca.

Caso um nó verificador identifique um comportamento anormal de um veículo, este reporta para o líder do agrupamento para que o líder atualize o valor de desconfiança do veículo (ver Figura 4.16). Um nó é considerado malicioso quando o seu valor de desconfiança é superior a um *threshold* mínimo. Quando isto ocorre, este veículo deve ser retirado da lista branca e ser acrescentado na lista negra. Para isto, o líder verifica se valor de $Td$ é menor ou igual que o *threshold* mínimo. Se for, atualiza a lista branca; se não for, notifica a Autoridade Certificadora para que esta atualize a lista negra. A AC transmite periodicamente estas listas para os líderes de agrupamento da rede.

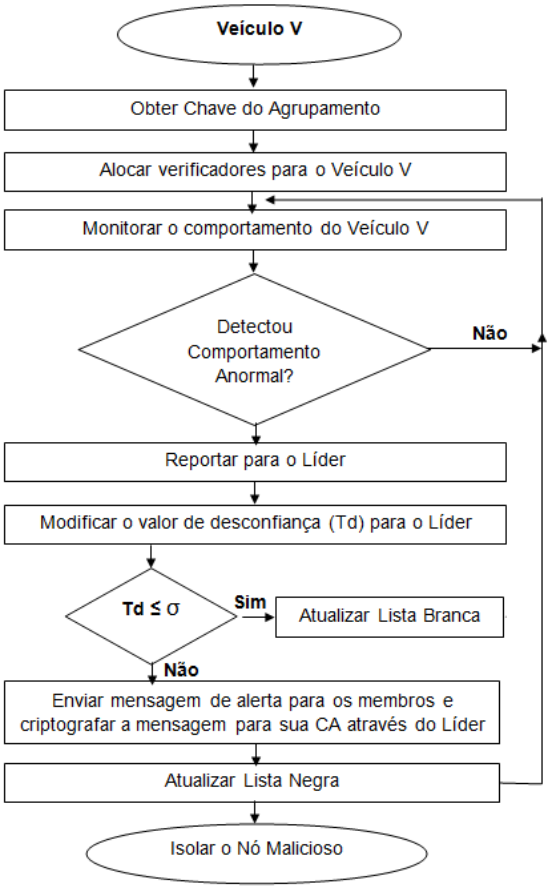



**Figura 4.16. Processo de Monitoramento de Veículos no DMV [Huang et al. 2014]**

Cada Autoridade Certificadora (AC) é uma terceira parte confiável que gerencia as identidades, as chaves criptográficas e as credenciais dos veículos dentro de sua região. Esta transmite sua lista negra periodicamente a todos os líderes de agrupamentos e, em seguida, estes as transmitem para todos os veículos localizados dentro do agrupamento. Os veículos pertencentes à lista negra são isolados da rede. Desta forma, outros veículos não aceitam mensagens vindas destes veículos. O líder de agrupamento é escolhido entre aqueles que têm o menor valor de desconfiança ($Td$), sendo este substituído quando apresentar

comportamentos anormais, ou quando outro veículo possuir o valor de desconfiança ($Td$) menor após atualização.

**[Li et al. 2012]**

Em [Li et al. 2012], foi proposto um sistema de reputação para redes veiculares que permite avaliar a confiabilidade da mensagem recebida de acordo com a reputação do veículo gerador da mensagem. O sistema proposto faz uso de um servidor de reputação centralizado. Uma das finalidades deste servidor é armazenar a reputação dos veículos, isto inclui a coleta de relato de experiências para produzir a reputação, e a propagação desta reputação na rede. O modelo de rede do mecanismo assume ainda a existência de dispositivos de comunicação entre os veículos e o servidor de reputação (chamados de pontos de acesso). Segundo os autores, não é necessária a comunicação contínua entre os veículos e o servidor de reputação. Estes servidores ficam posicionados em locais frequentemente visitados pelos veículos, como postos de combustíveis e semáforos. Quando um veículo recebe uma mensagem, caso este ainda não tenha tido uma experiência anterior com o emissor, ele consulta o servidor de reputação para obter a reputação global calculada através da média ponderada das experiências anteriores dos demais nós da rede. Além dos problemas de falha e desempenho decorrentes da abordagem com servidor centralizado, os autores não tratam a situação quando um nó é desconhecido também para o servidor de reputação.

**[Fernandes et al. 2013]**

Em [Fernandes et al. 2013], foi proposto um sistema de reputação descentralizado para avaliar o nível de confiança dos nós. Com este sistema, é possível identificar a presença de nós maliciosos em uma aplicação LDW (*Local Danger Warning*) e descartar seus alertas, uma vez que o comportamento inadequado destes nós pode comprometer a segurança das redes veiculares. No modelo assumido de redes, as unidades de bordo dos veículos e as RSUs não dependem de um ponto central avaliador da confiança dos nós e armazenador da base de reputação dos nós participantes, sendo o sistema caracterizado como descentralizado. Com as informações sobre o comportamento dos veículos, armazenadas de forma distribuída, a disponibilidade deste conteúdo é garantida. O objetivo da solução descrita é identificar a presença de nós maliciosos em uma aplicação LDW, chamada de RAMS+, de forma a descartar seus alertas, mesmo diante da formação de conluios.

No sistema proposto, a reputação do nó é avaliada consultando outros nós participantes da rede, fazendo uso de uma estratégia otimista na qual os nós têm reputação boa até que se prove o contrário. Cada veículo possui uma base de conhecimento individual (BCI), que contém informações sobre as interações passadas que este teve com outros veículos. A BCI armazena as experiências passadas mais recentes e o veículo a utiliza para o cálculo da reputação direta. Para encontrar a reputação global do veículo emissor do alerta, o veículo que recebeu o alerta calcula ainda a reputação agregada (indireta), definida a partir de informações de terceiros (mensagens recebidas de seus vizinhos). A reputação agregada é muito importante para o cálculo da reputação de veículos desconhecidos. Outra informação que auxilia a tomada de decisão de um veículo que recebeu uma mensagem de alerta é uma Lista de Reputações (*LR*) propagada pelas RSUs. Esta lista se torna mais importante com a proximidade do veículo no local do evento, pois este pode não ter tempo suficiente para

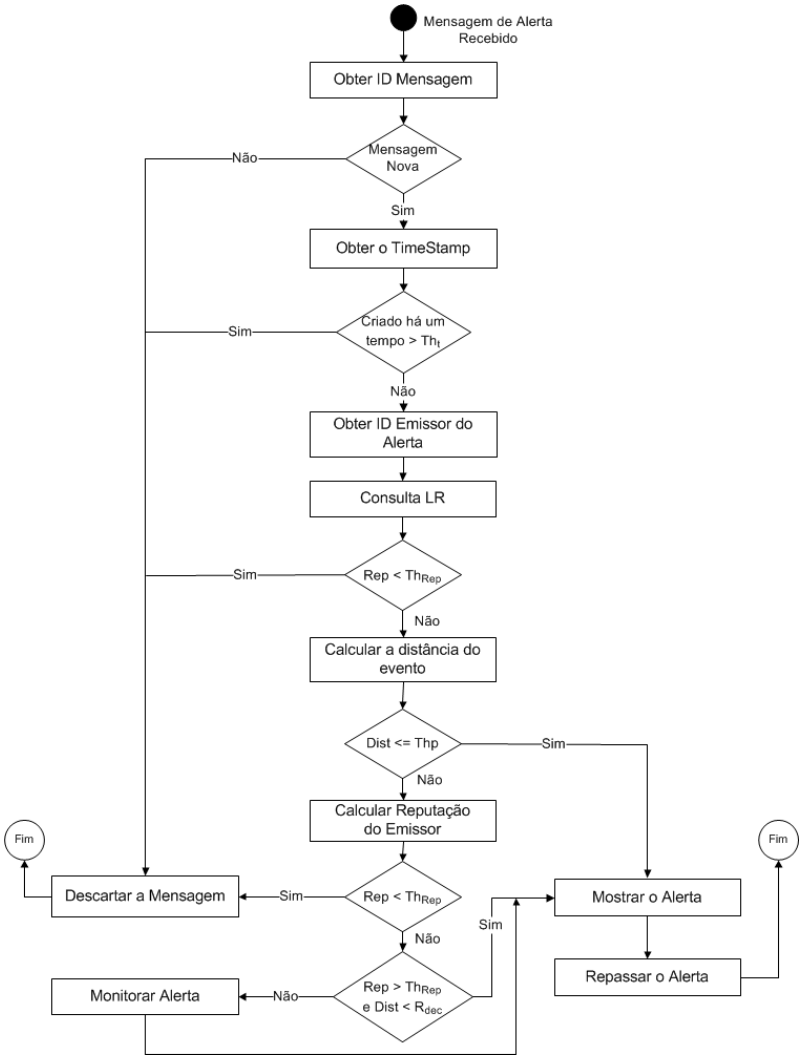



**Figura 4.17. Passos do Sistema de Reputação [Fernandes et al. 2013]**

calcular a reputação global, mantendo uma base de reputação mais abrangente e atualizada.

A Figura 4.17 ilustra os passos executados por um veículo ao receber uma mensagem de alerta (*MenAlert*). Após verificar que a mensagem *MenAlert* é uma mensagem nova e que esta foi criada a um tempo menor que o limiar de mensagem recente, o veículo, por meio do sistema de reputação, deve obter a reputação do emissor, consultando a Lista de Reputações (*LR*) recebida e avaliar se esta está acima de um limiar de reputação que considera este veículo confiável (e.g. 0,5). Caso este veículo seja considerado não confiável, a mensagem é descartada. Caso o veículo não esteja na lista de reputação ou caso este tenha sido considerado como confiável, a aplicação irá calcular a distância que o veículo está do evento relatado. Caso este veículo esteja muito próximo do local do evento (e.g. 100 metros), o alerta é mostrado para o condutor, pois pode não haver tempo hábil para a realização da consulta aos outros veículos. Caso contrário, a aplicação irá calcular a reputação global do emissor do alerta. Se o resultado da reputação for maior que o limiar de reputação, o emissor é avaliado como confiável. Caso este esteja na área de decisão, o alerta é mostrado para o condutor e repassado para os demais veículos. Quando o emissor

for avaliado como confiável e a distância do veículo estiver fora da área de decisão, um novo processo é criado para monitorar a entrada do veículo na área de decisão, mostrando posteriormente o alerta ao condutor quando este entrar nesta área.

## 4.5. Considerações finais: tendências e problemas em aberto

De acordo com [Raya e Hubaux 2007, Karagiannis et al. 2011], as soluções de segurança existentes para redes sem fio e até mesmo para redes *adhoc* não são facilmente aplicadas nas VANETs, dada a natureza destas redes que impõem novos desafios, a saber: restrições de tempo, escala da rede, alta mobilidade dos nós, volatilidade [Engoulou et al. 2014]. Um bom exemplo são os mecanismos de autenticação que usam assinaturas digitais que nas VANETs precisam ser adaptados para diminuir a sobrecarga de computação e de comunicação, garantir a privacidade condicional dos condutores e fazer uso de uma infraestrutura de chaves públicas flexível aos requisitos das VANETs [Isaac et al. 2010].

Para construir uma arquitetura de segurança e soluções que suportem a robustez das VANETs, é necessário o avanço no estudo das características e dos impactos dos ataques que podem ocorrer nestas redes. A gestão e a análise de riscos em redes veiculares são usadas para identificar e gerenciar as ameaças e os ataques potenciais na comunicação veicular. Soluções para o gerenciamento e a análise de tais ataques têm sido propostas [Aijaz et al. 2006, Ren et al. 2011, Ganan et al. 2012], porém é necessário avançar na caracterização do comportamento dos atacantes a fim de construir modelos que identifiquem os limites fundamentais do impacto dos ataque na rede e nas comunicações de forma menos abstrata e que considere as características realistas da rede e da comunicação [Karagiannis et al. 2011, Engoulou et al. 2014].

Garantir a confiabilidade das mensagens trafegadas na rede é relevante a fim de suportar as aplicações de monitoramento, impedindo que os condutores assumam ou tomem ações a partir de informações falsas [Karagiannis et al. 2011]. Um receptor deve não só verificar a integridade da informação recebida (p.ex. verificar a assinatura da mensagem) mas também confirmar a confiabilidade do emissor (confiança centrada na entidade). Os sistemas de reputação podem ser utilizados para estabelecer a confiança tanto dos veículos (confiança centrada na entidade) quanto das mensagens (confiança centrada no dado) [Tangade e Manvi 2013], porém a natureza distribuída e abrangente das redes veiculares tornam estes sistemas complexos. Uma solução adaptativa e ciente de contexto pode ser uma alternativa para tratar esta complexidade, porém, testes em diferentes cenários precisam ser realizados para verificar a efetividade dos sistemas de reputação adaptativos.

A privacidade é um dos maiores desafios na implementação e uso das aplicações de redes veiculares [Engoulou et al. 2014]. Informações como identidade e comportamento do condutor, localização presente e passada do veículo, em muitos casos, devem ser privadas. Conforme analisado na Seção 4.4.3, prover a privacidade condicional é essencial para garantir a revogação da privacidade dos nós com comportamentos maliciosos. De acordo com [Karagiannis et al. 2011], a privacidade é um conceito específico do usuário e um bom mecanismo deve permitir que um usuário selecione a privacidade que este deseja ter (privacidade adaptativa). Usuários podem querer usar diferentes níveis de privacidade dependendo no nível de confiança com quem estão se comunicando. Um requisito de alto

nível de privacidade geralmente resulta em um aumento na sobrecarga computacional e de comunicação, o que não pode ocorrer em VANETs. A privacidade adaptativa é uma questão em aberto, assim como a construção de sistemas de gerenciamento de identidades projetados particularmente para o ambiente altamente dinâmico das VANETs.

Muitas soluções que visam prover comunicação anônima em VANETs adotam o modelo de confiança zero (veículos não confiam nos outros veículos), são baseados em criptografia assimétrica e fazem uso de uma infraestrutura de chaves públicas (ICP). Um conjunto de pesquisadores consideram como consenso o uso de criptografia de curvas elípticas (ECC) para manter o anonimato em VANETs (exemplos foram apresentados na seção 4.4). Diante das características e restrições das VANETs, a sobrecarga computacional e de comunicação, a complexidade para gestão da confiança em uma ICP podem dificultar o uso efetivo destas soluções.

Uma das questões em discussão no momento é como garantir que os veículos terão conexão com a ICP no momento da renovação do certificado, principalmente, quando pseudônimos estiverem sendo utilizados. Será que uma conexão via telefonia celular (4G, 3G, etc) poderia ser assumida? Ou será que comunicações esporádicas com outros veículos ou RSUs seriam suficientes para suportar a realização desta operação? Como tornar essa comunicação o mais leve e rápida a fim de permitir a renovação do certificado com uma baixa sobrecarga na comunicação?

Segurança em VANETs é um tema de pesquisa bastante atual e ativo conforme pode ser observado pelas inúmeras publicações nas principais conferências nacionais e internacionais e nos tópicos de interesse dos periódicos internacionais. Diversas soluções estão sendo providas para os inúmeros ataques que estas redes estão sujeitas. Com o objetivo de avaliar a aplicabilidade, eficiência e eficácia das soluções propostas, pesquisadores realizam diversos experimentos por meio de simulações. Nestes experimentos, simuladores de rede e de tráfego são comumente utilizados a fim de fornecer resultados mais próximos aos ambientes veiculares reais. Porém, apesar das vantagens do uso de simuladores, tais como custo mais baixo e ambiente controlado, esta abordagem apresenta limitações. As simulações podem não refletir totalmente um ambiente real e podem até levar a resultados errados devido a simplificação dos modelos de rede e de tráfego e suposições em relação a propagação e interferências [Qin et al. 2014]. Testes em cenários reais, com veículos (OBUs) e unidades de acostamento reais, são muitas vezes considerados uma avaliação adicional e necessária aos trabalhos de simulação.

Apesar dos diversos trabalhos na literatura que descrevem aplicações de VANETs, ainda é necessário superar uma série de desafios científicos e tecnológicos de segurança para que estas aplicações sejam utilizadas e difundidas em sua forma plena. A implementação e avaliação dos mecanismos e soluções apresentados neste capítulo em cenário reais, já que grande parte dos trabalhos carecem de implementação ou provas formais, é uma oportunidade de pesquisa.

## Referências

[Ahlgren et al. 2012] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., e Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36.

[Aijaz et al. 2006] Aijaz, A., Bochow, B., Dötzer, F., Festag, A., Gerlach, M., Kroh, R., e Leinmüller, T.

(2006). Attacks on Inter Vehicle Communication Systems - an Analysis. In *WIT*, pages 189–194.

[Al Falasi e Barka 2011] Al Falasi, H. e Barka, E. (2011). Revocation in VANETs: A survey. In *International Conference on Innovations in Information Technology (IIT)*, pages 214–219.

[Al-kahtani 2012] Al-kahtani, M. (2012). Survey on security attacks in vehicular ad hoc networks (VANETs). In *International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–9.

[Al-Sultan et al. 2014] Al-Sultan, S., Al-Doori, M. M., Al-Bayatti, A. H., e Zedan, H. (2014). A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392.

[Alves 2011] Alves, V. d. M. (2011). Protocolo de comprometimento de bit eficiente com segurança sequencial baseado no modelo de memória limitada.

[Amro et al. 2013] Amro, B., Saygin, Y., e Levi, A. (2013). Enhancing privacy in collaborative traffic-monitoring systems using autonomous location update. *Intelligent Transport Systems, IET*, 7(4):388–395.

[Aslam e Zou 2009] Aslam, B. e Zou, C. (2009). Distributed certificate and application architecture for VANETs. In *IEEE Military Communications Conference (MILCOM)*, pages 1–7.

[Avelar et al. 2014] Avelar, E., Marques, L., dos Passos, D., Macedo, R., Dias, K., e Nogueira, M. (2014). Interoperability issues on heterogeneous wireless communication for smart cities. *Computer Communications*.

[Biswas et al. 2011] Biswas, S., Misic, J., e Misic, V. (2011). ID-based safety message authentication for security and trust in vehicular networks. In *31st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 323–331.

[Boneh et al. 2004] Boneh, D., Boyen, X., e Shacham, H. (2004). Short group signatures. In Franklin, M., editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin Heidelberg.

[Bononi et al. 2004] Bononi, L., Conti, M., e Gregori, E. (2004). Runtime optimization of IEEE 802.11 wireless LANs performance. *IEEE Transactions on Parallel and Distributed Systems*, 15(1):66–80.

[Chen et al. 2011] Chen, L., Ng, S.-L., e Wang, G. (2011). Threshold anonymous announcement in VANETs. *IEEE Journal on Selected Areas in Communications*, 29(3):605–615.

[Chuang e Lee 2011] Chuang, M.-C. e Lee, J.-F. (2011). TEAM: Trust-extended authentication mechanism for vehicular ad hoc networks. In *International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 1758–1761.

[Daeinabi e Rahbar 2013] Daeinabi, A. e Rahbar, A. G. (2013). Detection of malicious vehicles (DMV) through monitoring in vehicular ad-hoc networks. *Multimedia tools and applications*, 66(2):325–338.

[Dai et al. 2013] Dai, W., Moser, L., Melliar-Smith, P., Lombera, I. M., e team:, Y. (2013). The itrust local reputation system for mobile ad-hoc networks. In *International Conference on Wireless Networks*.

[de Paula et al. 2010] de Paula, W., de Oliveira, S., e Oliveira, J. M. (2010). Um mecanismo de reputação para redes veiculares tolerantes a atrasos e desconexões. In *Simpósio Brasileiro de Redes de Computadores (SBRC 2010)*, page 599.

[Engoulou et al. 2014] Engoulou, R. G., Bellaïche, M., Pierre, S., e Quintero, A. (2014). VANET security surveys. *Computer Communications*, 44(0):1 – 13.

[Faezipour et al. 2012] Faezipour, M., Nourani, M., Saeed, A., e Addepalli, S. (2012). Progress and challenges in intelligent vehicle area networks. *Communications of the ACM*, 55(2):90–100.

[Fernandes et al. 2013] Fernandes, C., de Simas, I., e Wangham, M. S. (2013). In *Simpósio Brasileiro de Segurança em Segurança da Informação e de Sistemas Computacioanais (SBSeg 2013)*, pages 157–169.

[Ganan et al. 2012] Ganan, C., Munoz, J., Esparza, O., Mata-Diaz, J., Alins, J., Silva-Cardenas, C., e Bartra-Gardini, G. (2012). RAR: Risk Aware Revocation Mechanism for Vehicular Networks. In *IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5.

[Gerla 2012] Gerla, M. (2012). Vehicular cloud computing. In *The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 152–155.

[Gerla e Kleinrock 2011] Gerla, M. e Kleinrock, L. (2011). Vehicular networks and the future of the mobile internet. *Computer Network*, 55(2):457–469.

[Grassi et al. 2014] Grassi, G., Pesavento, D., Pau, G., Vuyyuru, R., Wakikawa, R., e Zhang, L. (2014). VANET via named data networking. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 410–415.

[Haas et al. 2011] Haas, J. J., Hu, Y.-C., e Laberteaux, K. P. (2011). Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, 29(3):595–604.

[Hartenstein e Laberteaux 2010] Hartenstein, H. e Laberteaux, K. (2010). *VANET: vehicular applications and inter-networking technologies*. Wiley Online Library.

[Hartenstein e Laberteaux 2008] Hartenstein, H. e Laberteaux, K. P. (2008). A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171.

[He et al. 2014] He, W., Yan, G., e Xu, L. D. (2014). Developing vehicular data cloud services in the IoT environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595.

[Hill e Garrett 2011] Hill, C. J. e Garrett, J. K. (2011). AASHTO connected vehicle infrastructure deployment analysis. Technical report.

[Hossain et al. 2010] Hossain, E., Chow, G., Leung, V., McLeod, R. D., Mišić, J., Wong, V. W., e Yang, O. (2010). Vehicular telematics over heterogeneous wireless networks: A survey. *Computer Communications*, 33(7):775–793.

[Huang et al. 2014] Huang, Z., Ruj, S., Cavenaghi, M. A., Stojmenovic, M., e Nayak, A. (2014). A social network approach to trust management in VANETs. *Peer-to-Peer Networking and Applications*, 7(3):229–242.

[Hussain et al. 2012] Hussain, R., Son, J., Eun, H., Kim, S., e Oh, H. (2012). Rethinking vehicular communications: Merging vanet with cloud computing. *International Conference on Cloud Computing Technology and Science*, 0:606–609.

[IEEE 2013] IEEE (2013). IEEE standard for wireless access in vehicular environments security services for applications and management messages. *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pages 1–289.

[Isaac et al. 2010] Isaac, J., Zeadally, S., e Camara, J. (2010). Security attacks and solutions for vehicular ad hoc networks. *IET Communications*, 4(7):894–903.

[Juels e Wattenberg 1999] Juels, A. e Wattenberg, M. (1999). A fuzzy commitment scheme. In *ACM conference on Computer and communications security*, pages 28–36. ACM.

[Karagiannis et al. 2011] Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., e Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Communications Surveys and Tutorials*, 13(4):584–616.

[Kherani e Rao 2010] Kherani, A. e Rao, A. (2010). Performance of node-eviction schemes in vehicular networks. *IEEE Transactions on Vehicular Technology*, 59(2):550–558.

[Kwak et al. 2006] Kwak, D., Moon, S., Wang, G., e Deng, R. H. (2006). A secure extension of the Kwak–Moon group signcryption scheme. *Computers & Security*, 25(6):435 – 444.

[Laberteaux et al. 2008] Laberteaux, K. P., Haas, J. J., e Hu, Y.-C. (2008). Security certificate revocation list distribution for VANET. In *ACM international workshop on VehiculAr Inter-NETworking*, pages 88–89. ACM.

[Laurendeau e Barbeau 2006] Laurendeau, C. e Barbeau, M. (2006). Threats to security in DSRC/WAVE. In *Ad-Hoc, Mobile, and Wireless Networks*, pages 266–279. Springer.

[Lee et al. 2014] Lee, E., Lee, E.-K., Gerla, M., e Oh, S. (2014). Vehicular cloud networking: architecture and design principles. *IEEE Communications Magazine*, 52(2):148–155.

[Li e Wang 2007] Li, F. e Wang, Y. (2007). Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22.

[Li et al. 2012] Li, Q., Malip, A., Martin, K. M., Ng, S.-L., e Zhang, J. (2012). A reputation-based announcement scheme for VANETs. *IEEE Transactions on Vehicular Technology*, 61(9):4095–4108.

[Li et al. 2013] Li, X., Liu, J., Li, X., e Sun, W. (2013). Rgte: A reputation-based global trust establishment in VANETs. In *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pages 210–214.

[Lima et al. 2013] Lima, A., Albano, W., Nogueira, M., e de Sousa, J. N. (2013). Influência de ataques jamming sobre protocolos de roteamento em redes veiculares. In *Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG) - SBSeg*.

[Lin et al. 2008] Lin, X., Lu, R., Zhang, C., Zhu, H., Ho, P.-H., e Shen, X. (2008). Security in vehicular ad hoc networks. *IEEE Communications Magazine*, 46(4):88–95.

[Lin et al. 2007] Lin, X., Sun, X., Ho, P.-H., e Shen, X. (2007). Gsis: A secure and privacy-preserving protocol for vehicular communications. *IEEE Transactions on Vehicular Technology*, 56(6):3442–3456.

[Lo e Tsai 2009] Lo, N.-W. e Tsai, H.-C. (2009). A reputation system for traffic safety event on vehicular ad hoc networks. *EURASIP Journal on Wireless Communications and Networking*, 2009:9.

[Lu et al. 2008] Lu, R., Lin, X., Zhu, H., Ho, P.-H., e Shen, X. (2008). Ecpp: Efficient conditional privacy preservation protocol for secure vehicular communications. In *IEEE Conference on Computer Communications (INFOCOM)*.

[Macedo et al. 2013] Macedo, R., Melo, R. G. d., Melnisk, L., Santos, A., e Nogueira, M. (2013). Uma avaliação experimental de desempenho do roteamento multicaminhos em redes veiculares. In *Workshop de Gerência e Operações de Redes - SBRC*.

[Mangold et al. 2002] Mangold, S., Choi, S., May, P., Klein, O., Hiertz, G., e Stibor, L. (2002). Ieee 802.11 e wireless lan for quality of service. In *European Wireless*, volume 2, pages 32–39.

[Mejri et al. 2014] Mejri, M. N., Ben-Othman, J., e Hamdi, M. (2014). Survey on {VANET} security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53 – 66.

[Mezghani et al. 2014a] Mezghani, F., Dhaou, R., Nogueira, M., e Beylot, A.-L. (2014a). Content dissemination in vehicular social networks: Taxonomy and user satisfaction. *IEEE Communications Magazine*.

[Mezghani et al. 2014b] Mezghani, F., Dhaou, R., Nogueira, M., e Beylot, A.-L. (2014b). Utility-based forwarder selection for content dissemination in vehicular networks. In *IEEE International Conference on Personal, Indoor and Mobile Radio Communications (PIMRC)*.

[Mikki et al. 2013] Mikki, M., Mansour, Y., e Yim, K. (2013). Privacy preserving secure communication protocol for vehicular ad hoc networks. In *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 188–195.

[Mohr 2007] Mohr, A. (2007). A survey of zero-knowledge proofs with applications to cryptography. *Southern Illinois University, Carbondale*, pages 1–12.

[Nogueira et al. 2009] Nogueira, M., dos Santos, A., e Pujolle, G. (2009). A survey of survivability in mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 11(1):66–77.

[Nowatkowski 2010] Nowatkowski, M. E. (2010). *Certificate Revocation List Distribution in Vehicular Ad Hoc Networks*. PhD thesis, Atlanta, GA, USA. AAI3414506.

[Ostermaier et al. 2007] Ostermaier, B., Dotzer, F., e Strassberger, M. (2007). Enhancing the security of local dangerwarnings in VANETs-a simulative analysis of voting schemes. In *International Conference on Availability, Reliability and Security (ARES)*, pages 422–431.

[Papadimitratos et al. 2008] Papadimitratos, P., Buttyan, L., Holczer, T., Schoch, E., Freudiger, J., Raya, M., Ma, Z., Kargl, F., Kung, A., e Hubaux, J.-P. (2008). Secure vehicular communication systems: design and architecture. *IEEE Communications Magazine*, 46(11):100–109.

[Papadimitratos et al. 2009] Papadimitratos, P., La Fortelle, A., Evenssen, K., Brignolo, R., e Cosenza, S. (2009). Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, 47(11):84–95.

[Paruchuri e Durresi 2010] Paruchuri, V. e Durresi, A. (2010). Paave: Protocol for anonymous authentication in vehicular networks using smart cards. In *IEEE Global Telecommunications Conference (GLOBECOM 2010)*, pages 1–5.

[Pathre et al. 2013] Pathre, A., Agrawal, C., e Jain, A. (2013). Identification of malicious vehicle in vanet environment from ddos attack. *Journal of Global Research in Computer Science*, 4(6):1–5.

[Pinto 2013] Pinto, A. C. B. (2013). Protocolos criptográficos de computação distribuída com segurança universalmente composta.

[Qin et al. 2014] Qin, Z., Meng, Z., Zhang, X., Xiang, B., e Zhang, L. (2014). Performance evaluation of 802.11p wave system on embedded board. In *International Conference on Information Networking (ICOIN)*, pages 356–360.

[Rawat et al. 2012] Rawat, A., Sharma, S., e Sushil, R. (2012). Vanet:security attacks and its possible solutions. *Journal of Information and Operations Management*, 3:301–304.

[Raya e Hubaux 2007] Raya, M. e Hubaux, J.-P. (2007). Securing vehicular ad hoc networks. *Journal Computer Security*, 15(1):39–68.

[Raya et al. 2006a] Raya, M., Papadimitratos, P., e Hubaux, J.-P. (2006a). Securing vehicular communications. *IEEE Wireless Communications*, 13(5):8–15.

[Raya et al. 2006b] Raya, M., Papadimitratos, P., e Hubaux, J.-P. (2006b). Securing vehicular communications. *IEEE Wireless Communications Magazine, Special Issue on Inter-Vehicular Communications*, 13(LCA-ARTICLE-2006-015):8–15.

[Ren et al. 2011] Ren, D., Du, S., e Zhu, H. (2011). A novel attack tree based risk assessment approach for location privacy preservation in the VANETs. In *IEEE International Conference on Communications (ICC)*, pages 1–5.

[Ribeiro et al. 2004] Ribeiro, V., Campello, R., e Weber, R. F. (2004). Mecanismos de conhecimento zero empregados por esquemas de chave publica. In Solar, M., Fernandez-Baca, D., e Cuadros-Vargas, E., editors, *30ma Conferencia Latinoamericana de Informatica (CLEI2004)*, pages 644–650. Sociedad Peruana de Computacion. ISBN 9972-9876-2-0.

[Safi et al. 2009] Safi, S., Movaghar, A., e Mohammadizadeh, M. (2009). A novel approach for avoiding wormhole attacks in vanet. In *Second International Workshop on Computer Science and Engineering*, volume 2, pages 160–165.

[Samara et al. 2010] Samara, G., Al-Salihy, W., e Sures, R. (2010). Security issues and challenges of vehicular ad hoc networks (VANET). In *International Conference on New Trends in Information Science and Service Science (NISS)*, pages 393–398.

[Schleiffer et al. 2013] Schleiffer, C., Wolf, M., Weimerskirch, A., e Wolleschensky, L. (2013). Secure key management-a key feature for modern vehicle electronics. Technical report, SAE Technical Paper.

[Schütze 2011] Schütze, T. (2011). Automotive security: Cryptography for car2x communication. In *Embedded World Conference*.

[Shen et al. 2012] Shen, A.-N., Guo, S., Zeng, D., e Guizani, M. (2012). A lightweight privacy-preserving protocol using chameleon hashing for secure vehicular communications. In *IEEE on Wireless Communications and Networking Conference (WCNC)*, pages 2543–2548.

[Sichitiu e Kihl 2008a] Sichitiu, M. e Kihl, M. (2008a). Inter-vehicle communication systems: a survey. *IEEE Communications Surveys and Tutorials*, 10(2):88–105.

[Sichitiu e Kihl 2008b] Sichitiu, M. L. e Kihl, M. (2008b). Inter-vehicle communication systems: a survey. *IEEE Communications Surveys and Tutorials*, 10(2):88–105.

[Silva et al. 2014a] Silva, C., Cerqueira, E., e Nogueira, M. (2014a). Connectivity management to support reliable communication on cognitive vehicular networks (to appear). In *Wireless Days*.

[Silva et al. 2014b] Silva, C., Cerqueira, E., e Nogueira, M. (2014b). Mecanismo distribuído para seleção de canais em redes veiculares cognitivas. In *Workshop de Redes de Acesso em Banda Larga - SBRC*.

[Silva et al. 2008] Silva, E., Dos Santos, A., Albini, L., e Lima, M. (2008). Identity-based key management in mobile ad hoc networks: techniques and applications. *Wireless Communications, IEEE*, 15(5):46–52.

[Sukuvaara et al. 2013] Sukuvaara, T., Ylitalo, R., e Katz, M. (2013). Ieee 802.11p based vehicular networking operational pilot field measurement. *IEEE Journal on Selected Areas in Communications*, 31(9):409–417.

[Swamynathan et al. 2007] Swamynathan, G., Zhao, B. Y., Almeroth, K. C., e Zheng, H. (2007). Globally decoupled reputations for large distributed networks. *Adv. MultiMedia*, 2007(1):12–12.

[Tangade e Manvi 2013] Tangade, S. e Manvi, S. (2013). A survey on attacks, security and trust management solutions in VANETs. In *International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–6.

[Toor et al. 2008] Toor, Y., Muhlethaler, P., e Laouiti, A. (2008). Vehicle ad hoc networks: Applications and related technical issues. *IEEE Communications Surveys and Tutorials*, 10(3):74–88.

[Uzcategui e Acosta-Marum 2009] Uzcategui, R. e Acosta-Marum, G. (2009). Wave: a tutorial. *IEEE Communications Magazine*, 47(5):126–133.

[Wang et al. 2012] Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R., e Zhang, L. (2012). Data naming in vehicle-to-vehicle communications. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 328–333.

[Wang e Chigan 2007] Wang, Z. e Chigan, C. (2007). Countermeasure uncooperative behaviors with dynamic trust-token in VANETs. In *IEEE International Conference on Communications (ICC)*, pages 3959–3964. IEEE.

[Wasef et al. 2010] Wasef, A., Lu, R., Lin, X., e Shen, X. (2010). Complementing public key infrastructure to secure vehicular ad hoc networks [security and privacy in emerging wireless networks]. *IEEE Wireless Communications*, 17(5):22–28.

[Wasef e Shen 2009] Wasef, A. e Shen, X. (2009). Edr: Efficient decentralized revocation protocol for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 58(9):5214–5224.

[Weerasinghe et al. 2010] Weerasinghe, H., Fu, H., e Leng, S. (2010). Anonymous service access for vehicular ad hoc networks. In *Sixth International Conference on Information Assurance and Security (IAS)*, pages 173–178.

[Wolf e Gendrullis 2012] Wolf, M. e Gendrullis, T. (2012). Design, implementation, and evaluation of a vehicular hardware security module. In *Information Security and Cryptology (ICISC)*, pages 302–318. Springer.

[Xiong et al. 2013] Xiong, H., Zhu, G., Chen, Z., e Li, F. (2013). Efficient communication scheme with confidentiality and privacy for vehicular networks. *Computers and Electrical Engineering*, 39(6):1717 – 1725.

[Xu et al. 2006] Xu, W., Ma, K., Trappe, W., e Zhang, Y. (2006). Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47.

[Zhang 2011] Zhang, J. (2011). A survey on trust management for VANETs. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 105–112.

PROMOÇÃO:


ORGANIZAÇÃO:


PATROCÍNIO: