# Capítulo 3

# Segurança em Redes Centradas em Conteúdo: Vulnerabilidades, Ataques e Contramedidas

Igor C. G. Ribeiro, Flávio Q. Guimarães, Juliano F. Kazienko, Antonio A. de A. Rocha, Pedro B. Velloso, Igor M. Moraes, Célio V. N. Albuquerque

Instituto de Computação, PGC-TCC
Universidade Federal Fluminense

***Abstract***

*Content-Centric Networking (CCN) mitigates a number of security issues related to the current TCP/IP architecture. Presently, to provide authenticity and integrity of data shared in the Internet, it is necessary to ensure the security of the data repository and of the path that it follows towards the end user. Furthermore, the effectiveness of denial of service attacks against the current Internet suggests the need for the network infrastructure itself to provide mechanisms to prevent them. On the other hand, the CCN communication model is focused on the content itself and not on their physical address or location. This chapter provides an overview of the CCN architecture and how its mechanisms mitigate many security issues of today's Internet. Possible attacks and countermeasures proposed in the literature are described, in addition to its practical challenges and future perspectives.*

***Resumo***

*As Redes Centradas no Conteúdo (Content-Centric Networking), ou simplesmente CCN, simplificam a solução de determinados problemas de segurança relacionados a arquitetura TCP/IP. Atualmente, para se prover a autenticidade e a integridade dos dados compartilhados na rede, faz-se necessário garantir a segurança do repositório e do caminho que os dados devem percorrer até o usuário final. Além disso, a contínua eficácia dos ataques de negação de serviço praticados contra a Internet atual sugere a necessidade de que a própria infraestrutura da rede forneça mecanismos para mitigá-los. Por outro lado, o modelo de comunicação da CCN é focado no conteúdo em si e não em sua*

 

*localização física. Este capítulo apresenta uma visão geral da arquitetura CCN e como os seus mecanismos mitigam os problemas de segurança tradicionais. São abordados os ataques e possíveis contramedidas propostas na literatura, além de indicar seus desafios e perspectivas futuras.*

## 3.1. Introdução

Atualmente, a maior parte do tráfego da Internet é gerada por aplicações de recuperação de conteúdo [Labovitz et al. 2010]. Em 2011, o *Cisco Visual Networking Index* apontou que os tráfegos de aplicações par-a-par (*peer-to-peer* - P2P) e de vídeo somados corresponderam aproximadamente a 80% do tráfego da Internet [Kurose 2012]. Essa popularização de aplicações P2P mostra que os usuários estão interessados no conteúdo em si, independentemente do local onde estes estão armazenados ou de quem os distribui. Esses dados introduzem a ideia de que, ao invés de se tentar atender aos requisitos das aplicações de distribuição de conteúdo à Internet atual, deve-se remodelar a rede para atender a tais necessidades de maneira mais simples. Uma vez que o conteúdo é o mais importante para os usuários, os serviços oferecidos pela rede devem ser orientados ao conteúdo em si e não à sua localização física. Assim, os usuários devem ser capazes de requisitar conteúdos pelo nome e cabe a rede localizar este conteúdo, onde quer que ele esteja. Essa é uma das principais primitivas de rede do paradigma de comunicação baseada em conteúdos. Entre suas principais vantagens estão o aumento da eficiência na entrega e da disponibilidade do conteúdo e a maior simplicidade na implementação de mecanismos de segurança [Brito et al. 2012].

Várias arquiteturas de rede baseadas no paradigma de orientação ao conteúdo foram propostas, como *Data Oriented Network Architecture* (DONA) [Koponen et al. 2007], *Publish-Subscribe Internet Routing Paradigm* (PSIRP) [Lagutin et al. 2010] e as Redes Centradas em Conteúdo (*Content-Centric Networking* - CCN) [Jacobson et al. 2009]. Apesar de cada proposta enfatizar diferentes aspectos de projeto, elas compartilham três princípios fundamentais: a recuperação de conteúdos baseada em requisição e resposta, o armazenamento (*caching*) de conteúdos por todos os nós da rede e o modelo de segurança orientado ao conteúdo [Ghodsi et al. 2011a].

Apesar dessa diversidade de arquiteturas, a CCN é aquela que tem impulsionado o interesse da comunidade científica pela pesquisa sobre o paradigma orientado ao conteúdo [Ghodsi et al. 2011a]. Este fato pode ser constatado verificando-se que após sua concepção em 2009, foram criados dois *workshops* específicos para o paradigma orientado ao conteúdo nas conferências ACM SIGCOMM 2011 e IEEE INFOCOM 2012. Em tais *workshops* aproximadamente 70% dos artigos publicados tem a CCN como objeto de estudo. Além disso, a CCN possui um protótipo, o CCNx, [1] que permite a implementação e a validação de novas propostas em redes experimentais.

A CCN torna mais simples a solução de alguns problemas de segurança relacionados à arquitetura atual da Internet. Nesta última, busca-se prover a autenticidade e a integridade dos dados trocados na rede garantindo a segurança do repositório e do caminho que os dados devem percorrer até o usuário final. Esta metodologia é contra intuitiva, já que a necessidade de se assegurar um determinado conteúdo implica na necessidade

[1] Disponível em http://www.ccnx.org.

 

de assegurar todos os dispositivos utilizados no seu encaminhamento, desde a fonte até o destino. Além disso, a contínua eficácia dos ataques de negação de serviço praticados contra a Internet atual sugere a necessidade de que a própria infraestrutura da rede forneça mecanismos para mitigá-los. Por outro lado, o modelo de comunicação da CCN é focado no conteúdo em si e não em sua localização física. Portanto, não há a necessidade de estabelecer a segurança de outros componentes que não o próprio conteúdo. Para isso, a CCN exige que todos os dados trafegados na rede sejam assinados, garantindo sua integridade e autenticidade. Além disso, a CCN implementa um mecanismo de agregação de pacotes que limita a quantidade de tráfego na rede e estabelece que todos os roteadores devam realizar *cache* de conteúdos. Essas características colaboram para a mitigação de ataques de negação de serviço, comuns na Internet atual.

A arquitetura CCN, entretanto, introduz desafios e vulnerabilidades diferentes dos enfrentados atualmente na arquitetura TCP/IP. Apesar do aumento na privacidade dos usuários introduzida pela ausência de informações de origem e destino nos pacotes, a CCN também possui vulnerabilidades neste mesmo sentido. Uma vez que um nome de conteúdo é utilizado como parâmetro para sua recuperação, este deve ser semanticamente relacionado com conteúdo em si. Desta forma, caso um adversário capture uma requisição a um conteúdo, ele será capaz de identificá-lo mesmo sem obtê-lo. Com relação aos ataques de negação de serviço, apesar das técnicas utilizadas na Internet atual serem ineficazes contra a CCN, estas podem ser adaptadas com o objetivo de explorar diversas características desta arquitetura. Por fim, a realização de *cache* de conteúdos por todos os nós da rede permite uma maior disponibilidade e eficiência na recuperação dos mesmos. Em contrapartida, ataques como a poluição de *cache* e *cache snooping*, já existentes na Internet atual, são potencializados na CCN.

O restante deste capítulo está organizado da seguinte forma: a Seção 3.2 apresenta uma visão geral da arquitetura CCN, discutindo em particular o modelo de nomeação adotado e o processo de recuperação de conteúdos. A Seção 3.3 discute como os mecanismos de segurança presentes na CCN são utilizados para mitigar problemas tradicionais das redes TCP/IP. Na Seção 3.4, são abordados alguns ataques específicos para a CCN e suas possíveis contramedidas propostas na literatura. Por fim, a Seção 3.5 apresenta as considerações finais, as perspectivas futuras e os problemas em aberto sobre o tema.

## 3.2. Visão Geral da CCN

A CCN é uma arquitetura de rede projetada para permitir que usuários possam obter conteúdos de maneira eficiente e segura. Na Internet atual, para que um usuário possa obter o conteúdo desejado, ele precisa requisita-lo diretamente ao servidor onde o mesmo está armazenado. Por outro lado, a CCN desassocia os dados de sua localização física, permitindo que usuários (consumidores) requisitem conteúdos pelo nome, sem se preocupar com o local onde os mesmos estão armazenados. A infraestrutura da rede, por sua vez, é responsável por encontrar e devolver o conteúdo requisitado.

Para garantir a maior disponibilidade dos conteúdos, além da maior eficiência em sua recuperação, os roteadores da CCN, chamados de roteadores de conteúdo, armazenam em *cache* os dados por ele recebidos. Assim, em requisições posteriores o conteúdo desejado pode ser recuperado do *cache* mais próximo, reduzindo o tempo de resposta e o

 

consumo de largura de banda no núcleo da rede.

Como pode ser observado na Figura 3.1, apesar da estrutura adotada pela CCN ser bem parecida com aquela adotada pelas redes TCP/IP, ela apresenta algumas diferenças fundamentais. Em primeiro lugar, a "cintura fina"da CCN passa a ser o conteúdo em si e não o protocolo IP (*Internet Protocol*). Essa característica ilustra bem o paradigma de orientação ao conteúdo adotado pela CCN. A segunda diferença é a definição de uma camada específica para segurança. Uma vez que os conteúdos podem ser obtidos de qualquer nó da rede, confiáveis ou não, é preciso garantir que os mesmos sejam autênticos e íntegros, mas sem a necessidade de garantir essas mesmas propriedades à toda a infraestrutura da rede. As questões relativas a camada de segurança serão discutidas na Seção 3.3. Por fim, a CCN também inclui uma camada de estratégia à sua pilha de protocolos. Como será explicado nesta seção, os pacotes trocados na CCN são livres de *loops*, o que permite aos roteadores os encaminharem através de múltiplas interfaces ao mesmo tempo, utilizando qualquer tecnologia disponível, como *Ethernet*, 3G, *Bluetooth* e IEEE 802.11. Ao invés de sempre encaminhar pacotes por todas as interfaces disponíveis, o que poderia não ser muito eficiente, os roteadores podem utilizar diferentes estratégias de encaminhamento de pacotes, de acordo com a camada de estratégia da CCN. Outra característica importante é que a CCN pode funcionar sobre o protocolo IP, o que permite que a mesma seja implementada de forma incremental na Internet.

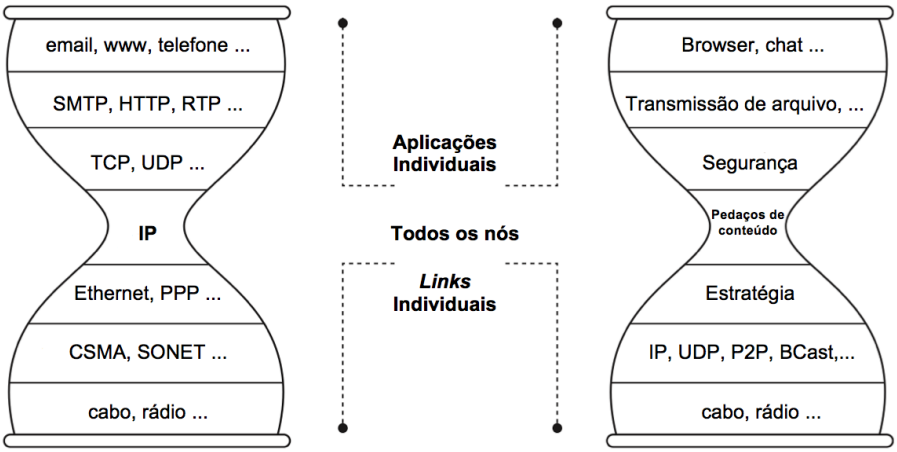


**Figura 3.1. Comparação da pilha de protocolos TCP/IP e CCN [Jacobson et al. 2012].**

### 3.2.1. Nomeação de Conteúdos

A CCN define um esquema de nomes muito parecido com aquele utilizado para nomear arquivos e diretórios no sistema operacional Linux. Os nomes são formados por um conjunto de componentes separados entre si pelo caractere ”/”, representando uma hierarquia [Jacobson et al. 2009]. Os nomes atribuídos aos conteúdos são opacos à rede, o que significa que os roteadores não têm conhecimento da semântica dos mesmos. Apenas sua estrutura hierárquica é relevante para a rede. Dessa forma, contanto que os nomes sigam tal estrutura, os publicadores de conteúdo são livres para adotar qualquer padrão de nomeação que atenda melhor as suas necessidades. Por exemplo, supondo que alunos da disciplina Segurança de Redes quisessem obter o vídeo da primeira aula desta disciplina,

 

o Instituto de Computação da Universidade Federal Fluminense poderia publicá-lo através do nome `/br.uff/ic/segredes/aulas/aula1.mp4`. É importante notar que na ausência de motores de busca, os usuários precisarão realizar requisições para conteúdos informando os nomes dos mesmos. Dessa maneira, os nomes devem refletir o significado dos conteúdos. Isso significa que se o conteúdo é o filme do `batman`, então dar o nome de `superman` a esse conteúdo não faria sentido e confundiria os usuários.

A utilização desse esquema de nomeação torna fácil o estabelecimento de relacionamentos entre pedaços de conteúdo, como ilustrado na Figura 3.2. Por exemplo, um pedaço de conteúdo do terceiro segmento da primeira versão do vídeo `intro.avi` poderia ser nomeado como `/br.uff/videos/intro.avi/1/3`. Dessa forma, uma requisição utilizando o nome `/br.uff/videos/intro.avi` poderia remeter ao primeiro segmento deste conteúdo e usando informações contidas neste, juntamente com o conhecimento do padrão de nomeação utilizado pelo publicador, o consumidor poderia então requisitar os segmentos seguintes. O número de possibilidades de estruturação dos nomes é realmente quase ilimitado, pois só depende da criatividade e organização dos publicadores.

Outra característica muito importante com relação aos nomes hierárquicos utilizados na CCN é a escalabilidade. Uma vez que o roteamento de requisições é baseado nos nomes de conteúdo, a estrutura hierarquia dos mesmos permite que os prefixos sejam agregados nos roteadores, reduzindo assim o tamanho das tabelas de encaminhamento.

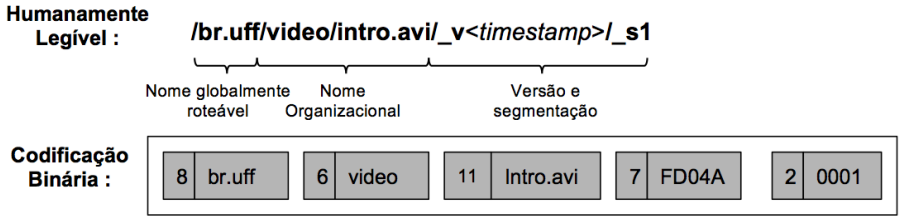


**Figura 3.2. Nome hierárquico estruturado [Jacobson et al. 2012].**

### 3.2.2. Tipos de Pacotes da CCN

Toda a comunicação na CCN é realizada utilizando-se apenas dois tipos de pacotes: pacotes de interesses e pacotes de dados, como ilustra a Figura 3.3, onde cada pacote de interesse é satisfeito pelo seu respectivo pacote de dados.

Os pacotes de interesse são utilizados por um consumidor para requisitar os conteúdos através de seus nomes. Além do nome do conteúdo requisitado, tais pacotes transportam ainda as seguintes informações:

- **Origem da Resposta:** a CCN permite que os conteúdos sejam obtidos através de algum *cache* da rede ou de sua fonte. Em alguns casos, como a recuperação de conteúdos dinâmicos, o usuário sabe, a priori, que o conteúdo requisitado por ele não estará armazenado em qualquer *cache* da rede. Neste cenário, este campo permite ao usuário indicar que o conteúdo deve ser retornado diretamente por sua fonte, evitando assim processamento desnecessário.

 

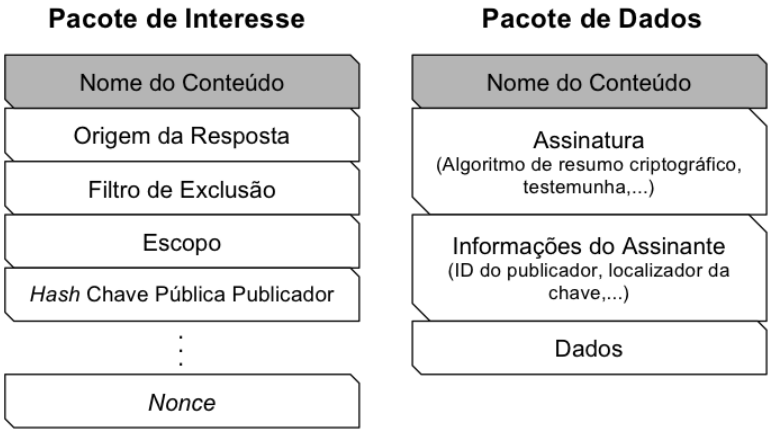


**Figura 3.3. Tipos de pacotes da CCN [Jacobson et al. 2012].**

- **Filtro de Exclusão:** Na CCN os conteúdos são requisitados através de seus nomes completos ou por algum prefixo destes. Quando um pacote de interesse é recebido por um roteador ou fonte de conteúdo, este realiza uma comparação de maior prefixo entre o nome no pacote de interesse e os nomes dos conteúdos armazenados. Caso uma correspondência seja encontrada, o conteúdo é retornado ao requisitante. Se mais de um conteúdo corresponder ao nome requisitado, então será necessária a escolha de uma dentre estas múltiplas opções, já que apenas uma resposta pode ser recebida para cada interesse enviado. Neste sentido, o filtro de exclusão é uma lista de componentes que não devem estar contidos no nome do conteúdo retornado. Por exemplo, suponha que os conteúdos `/br.uff/videos/intro.avi` e `/br.uff/audios/aula.mp3` estão armazenados em *cache* no roteador *A*. Suponha também que o roteador *A* receba um pacote de interesse para o nome de conteúdo `/br.uff` e que seu filtro de exclusão contenha o componente `/videos`. Neste caso, a comparação de maior prefixo resultaria na correspondência de ambos os conteúdos presentes no *cache*. Entretanto, após a análise do filtro de exclusão, o conteúdo `/br.uff/videos/intro.avi` seria eliminado da resposta, restando assim apenas o conteúdo `/br.uff/audios/aula.mp3` a ser retornado.

- **Escopo:** por padrão, a CCN trata as requisições de maneira recursiva. Isso significa que se um roteador não é capaz de atender a um determinado interesse, este encaminha o pacote para o próximo nó no caminho determinado pelo protocolo de roteamento. Para alterar este comportamento, o campo escopo é utilizado para definir até que nível na hierarquia da rede o pacote de interesse deve ser processado. Por exemplo, definindo como ”2” o valor deste campo, o pacote de interesse pode alcançar no máximo o roteador de próximo salto.

- ***Hash* da Chave Pública do Publicador:** este campo permite ao usuário especificar qual deve ser o publicador do conteúdo requisitado. Para tanto, transporta um *hash* (SHA-256) da chave pública do publicador pretendido. Assim, o conteúdo requisitado só será retornado se o *hash* da chave pública contido no pacote de interesse for igual ao *hash* da chave pública contido no pacote de dados.

 

- **Nonce:** para evitar a formação de *loops* na rede, os pacotes carregam um número aleatório, chamado de *nonce*, utilizado pelo roteador para identificar pacotes de interesse duplicados.

Os pacotes de dados são utilizados para transportar os conteúdos requisitados. Por questões de segurança, como discutido na Seção 3.3, todos os pacotes de dados são assinados pelo publicador. Para que essa assinatura possa ser verificada, tais pacotes também contêm informações como o algoritmo de criptografia utilizado e um localizador para a recuperação da chave pública do publicador.

### 3.2.3. Estrutura Interna de um Nó da CCN

Cada nó da CCN mantém três estruturas de dados para operações de encaminhamento de pacotes: um armazenador de conteúdo ( *Content Store* - CS) para *cache* temporário de dados recebidos, uma tabela de interesses pendentes (*Pending Interest Table* - PIT) e uma base de informações de encaminhamento (*Forwarding Information Base* - FIB).

#### 3.2.3.1. Armazenador de Conteúdos - CS

O armazenador de conteúdo provê a funcionalidade do aumento da eficiência da recuperação do conteúdo pelos consumidores através da disponibilidade de conteúdo em *cache* nos roteadores. Para isso, o CS mantém uma tabela de índices atualizada com os nomes dos conteúdos guardados em seu *cache* que possibilita verificar a correspondência do nome do pacote de interesse ou pacote de dados com o nome dos conteúdos armazenados. A atualização da tabela indexada depende do período de expiração e política de substituição de *cache* adotada.

#### 3.2.3.2. Tabela de Interesses Pendentes - PIT

A PIT pode ser entendida, de maneira simplificada, como uma tabela indexada por nomes de conteúdo. Cada uma de suas entradas guarda uma lista de interfaces por onde interesses para um mesmo conteúdo foram recebidos. Dessa forma, ao receber um pacote de dados, o roteador utiliza o nome de conteúdo para indexar a PIT e obter a lista de interfaces por onde interesses para este conteúdo foram recebidos. O pacote de dados é então encaminhado a todas as interfaces contidas nesta lista. Isso significa que os conteúdos retornam aos consumidores seguindo o caminho inverso daquele criado pelo seu respectivo interesse.

Devido a este mecanismo, a CCN pode realizar a agregação de pacotes de interesse, reduzindo assim o consumo de largura de banda no núcleo da rede. Uma vez que os pacotes de dados sempre retornam pelo mesmo caminho do seu respectivo interesse, os roteadores de conteúdo não precisam encaminhar mais de um interesse para o mesmo conteúdo. Se um novo interesse para o mesmo conteúdo for obtido, basta adicionar sua interface de entrada à lista de interfaces presente na entrada correta da PIT. Assim, quando

 

o conteúdo for recebido pelo roteador, todos os interesses para o mesmo serão atendidos de uma só vez.

Uma vez que um pacote de interesse pode ser encaminhado através de múltiplas interfaces, as diversas cópias desse mesmo interesse podem seguir caminhos diferentes, mas com alguns roteadores em comum. Para evitar que pacotes de interesse realizem *loops*, isto é, o mesmo interesse recebido mais de uma vez pelo mesmo roteador sendo tratado como dois interesses distintos, a PIT guarda, além das interfaces de entrada, os *nonces* de cada interesse. Assim, se um mesmo interesse for recebido mais de uma vez pelo mesmo roteador, a comparação do *nonce* do interesse com aquele armazenado na PIT detectará essa situação, resultando no descarte do interesse repetido. Ainda, para que os interesses não permaneçam eternamente pendentes, no caso do conteúdo requisitado não existir, a cada uma das entradas da PIT é associado um temporizador e após sua expiração, a entrada correspondente é descartada.

#### 3.2.3.3. Base de Informações de Encaminhamento - FIB

A FIB de um roteador de conteúdo é semelhante a de um roteador IP, mas diferem em dois pontos fundamentais. Geralmente uma FIB IP associa um prefixo de rede específico a uma única interface de saída. Esta interface de saída faz parte do caminho de melhor custo calculado pelo protocolo de roteamento. Por outro lado, a FIB CCN pode conter uma lista com várias interfaces de saída, permitindo que um pacote de interesse seja encaminhado através de múltiplos caminhos. Outra diferença fundamental é que uma entrada FIB IP contém apenas as informações do próximo salto, enquanto uma entrada FIB CCN também contém informações que auxiliam no processo de encaminhamento adaptativo dos pacotes de interesse, como tratado na Seção 3.2.5.

A estrutura da CCN permite que protocolos de roteamento de estado de enlace usados na Internet atual sejam adaptados para trabalhar com prefixos de nome, ao invés de endereços IP. Dessa forma, as entradas na FIB são obtidas através de tais protocolos e indicam qual deve ser a interface de saída de um interesse, baseado no nome de conteúdo contido no mesmo. Quando um determinado prefixo deixa de ser anunciado pelo protocolo de roteamento, este não é eliminado da FIB imediatamente, já que alguns interesses para este prefixo ainda podem estar em trânsito. Assim, para evitar que pacotes de interesse sejam descartados devido a variações nas rotas durante a fase de convergência do protocolo de roteamento, as entradas da FIB somente são descartadas após a expiração de um tempo de vida associado às mesmas.

Se um pacote de interesses for recebido por um nó da rede e o conteúdo requisitado não estiver em seu CS, e também não existir uma entrada correspondente na PIT ou na FIB, este interesse será descartado. Isso ocorre, pois o nó não possui o conteúdo requisitado e não sabe como encaminhar o pedido na direção de algum outro nó que o contenha. Se esse interesse estiver pendente na PIT de outros roteadores, essas entradas eventualmente expirarão. Entretanto, esperar que as entradas da PIT expirem pode levar a um desperdício excessivo de tempo. Além disso, quando um pacote de interesse não consegue encontrar o conteúdo requisitado, outros interesses para o mesmo conteúdo são bloqueados, devido a agregação desses pacotes no roteador. Assim, [Yi et al. 2012] pro-

 

põem que após descartar o pacote de interesse, o nó envie, pela mesma interface que o mesmo foi recebido, uma confirmação negativa (NACK) contendo o nome de conteúdo presente no pacote de interesse descartado, e um código de erro, para ajudar na identificação do problema. Quando um roteador recebe um NACK para certo conteúdo, ele deve tentar encaminhar novamente o interesse por algum conjunto de interfaces alternativas. Se tal conjunto não existir, então o roteador também não será capaz de atender ou encaminhar os interesses para este conteúdo. Assim, um NACK também deve ser encaminhado pelo roteador através das interfaces contidas na entrada da PIT correspondente, removendo-a em seguida. Essa técnica permite que um problema na recuperação de conteúdos seja detectado mais rapidamente, aumentando assim sua eficiência.

### 3.2.4. Processo de Recuperação de Conteúdos

Para requisitar um conteúdo, o consumidor envia um pacote de interesse à rede contendo, no mínimo, o nome do mesmo. Após receber o interesse, o roteador extrai o nome do conteúdo e faz uma busca pelo maior prefixo em seu CS. Caso uma correspondência seja encontrada, então o roteador gera um pacote de dados para o conteúdo e o envia para a interface de chegada do interesse. Caso contrário, o roteador verifica se já existe uma entrada em sua PIT para o conteúdo. Se existir, o roteador verifica se o *nonce* do interesse recebido está contido na lista de *nonces* armazenada na entrada da PIT. Em caso afirmativo, o interesse recebido é uma cópia duplicada e deve ser descartado. Caso contrário, o interesse é novo e sua interface de entrada e seu *nonce* são armazenados nesta entrada da PIT. Em seguida, o interesse é descartado. Caso nenhuma entrada correspondente na PIT seja encontrada, o roteador faz uma busca de maior prefixo em sua FIB, tentando encontrar interfaces de saída para encaminhar o interesse. Se nenhuma interface de saída for encontrada para o conteúdo requisitado pelo interesse, então este é descartado e um NACK é enviado por sua interface de chegada. Caso contrário, é criada uma entrada na PIT contendo a interface de chegada e o *nonce* do interesse. Em seguida, o roteador consulta sua camada de estratégia para decidir por quais interfaces encaminhar o interesse.

Após receber um pacote de dados, o roteador extrai o nome do conteúdo e verifica se existe alguma entrada na PIT para o mesmo. Se não existir, significa que o conteúdo não foi requisitado e o pacote de dados é descartado. Se existir, o conteúdo do pacote de dados é armazenado no CS do roteador e em seguida é encaminhado por todas as interfaces contidas na entrada da PIT. Se o dado recebido for um NACK, e não um pacote de dados, então o roteador poderá tentar reencaminhar o interesse utilizando um conjunto alternativo de interfaces, ou simplesmente descartar a entrada correspondente da PIT e encaminhar o NACK para todas as interfaces contidas na mesma. A Figura 3.4 ilustra o processo de recuperação de conteúdos.

Como um exemplo, a Figura 3.5 ilustra o estado da PIT, da FIB e do CS de um roteador durante o processo de encaminhamento de dois pacotes de interesse para o conteúdo `/br.uff/videos/intro.avi/v3/s3`. A Figura 3.5(a) mostra o estado inicial do roteador, antes da chegada de interesses para tal conteúdo. Quando o primeiro desses interesses chega ao roteador pela interface ”0”, a tabela PIT está vazia. Assim, a Figura 3.5(b) mostra que após o roteador verificar que o conteúdo requisitado não se encontra em seu CS, uma nova entrada é criada na PIT para este interesse, contendo a in-

 

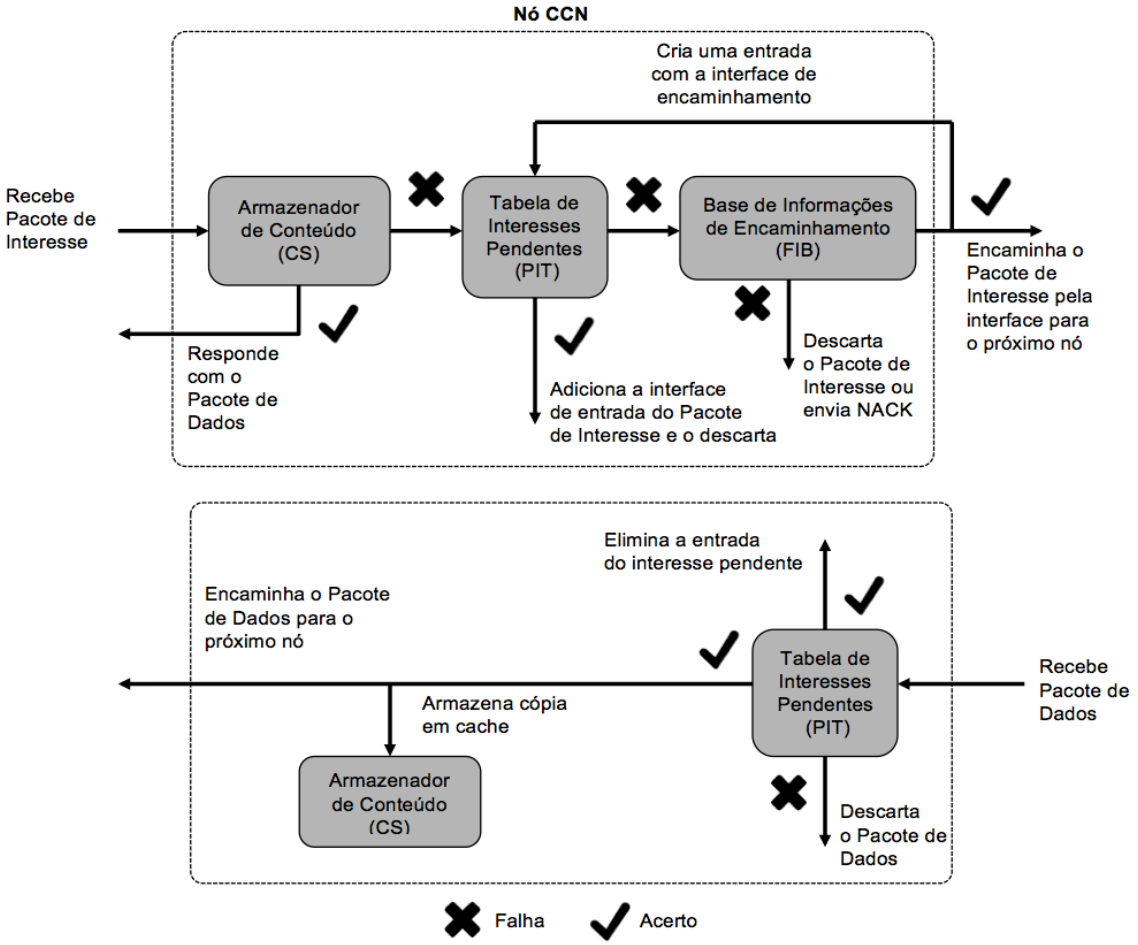


**Figura 3.4. Processo básico dos pacotes de interesses e dados[Yi et al. 2012].**

terface de chegada do mesmo (interface ”0”). Em seguida, este interesse é encaminhado pela interface ”2” de acordo com as regras da FIB. Logo após, um segundo interesse para o mesmo conteúdo é recebido pela interface ”1”. Uma vez que já existe uma entrada na PIT para o conteúdo requisitado, a interface de chegada do interesse (interface ”1”) é adicionada a lista de interfaces desta entrada, como ilustrado na Figura 3.5(c).

### 3.2.5. Encaminhamento Adaptativo

Nas redes IP, os protocolos de roteamento são responsáveis por encontrar uma rota com o menor custo entre dois nós e disseminá-la para os roteadores da rede. Assim, o processo de encaminhamento de pacotes apenas segue o caminho definido pelo protocolo de roteamento. Uma vez que os pacotes da CCN não realizam *loops*, eles podem ser encaminhados por múltiplas interfaces ao mesmo tempo. Assim, a função do protocolo de roteamento passa a ser a de identificar todas as rotas disponíveis para um determinado nome de conteúdo e disseminar essa informação na rede para que os roteadores possam montar suas FIBs. Quando um pacote de interesse precisa ser encaminhado, pode existir mais de uma interface de saída disponível. O roteador pode então decidir encaminhar o interesse por todas as interfaces disponíveis, ou escolher um subconjunto das mesmas.

 

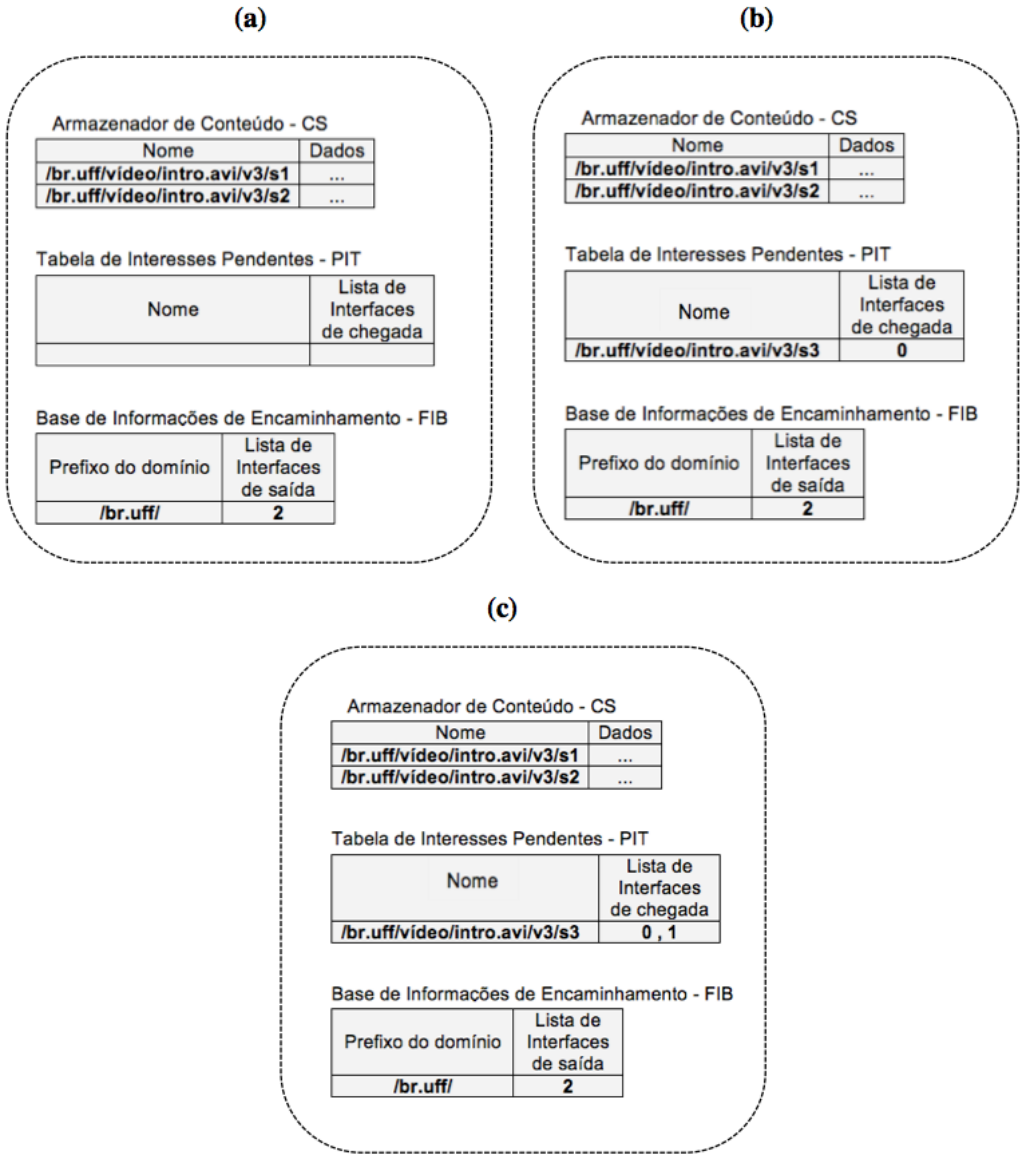


**Figura 3.5. Estruturas do mecanismo de encaminhamento de um nó CCN.**

Outra característica importante da CCN é o fato dos pacotes de dados sempre percorrerem o caminho inverso do interesse que os originou. Dessa forma, cada roteador no caminho de um pacote de interesse estará também obrigatoriamente no caminho do respectivo pacote de dados. Essa característica permite que os roteadores obtenham algumas estatísticas que podem ser utilizadas para apoiar o processo de encaminhamento de interesses. Assim, torna-se possível escolher os melhores caminhos adaptando-se às condições atuais da rede.

Neste contexto, [Yi et al. 2012] propõem que algumas informações adicionais sejam armazenadas na FIB, de modo a permitir uma classificação das interfaces. Quando um determinado interesse é encaminhado por uma interface e o respectivo conteúdo também é recebido pela mesma, o roteador pode estimar o RTT para essa interface e armazenar este valor na entrada correspondente da FIB. Assim, se todos os roteadores escolherem as

 

interfaces que apresentam o menor RTT para um determinado nome de conteúdo, então o caminho com a menor latência terá sido utilizado.

Além da latência, as interfaces também podem ser classificadas quanto ao sucesso na recuperação de conteúdos. Para tanto, é possível utilizar um sistema de cores, onde interfaces que acabaram de ser ligadas, ou que estão contidas em novas entradas na FIB são classificadas como amarelas, indicando um estado neutro. Quando um nó da rede recebe um pacote de dados, a interface de chegada do mesmo é marcada como verde, indicando seu resultado positivo na recuperação do conteúdo. Uma interface passa de verde para amarelo quando expira o tempo de vida de um interesse pendente ou depois que a interface fica sem uso durante um período de tempo determinado. Caso uma ou mais interfaces detectem um erro de enlace, ou recebam um NACK, estas são classificadas como vermelhas.

Uma vez classificadas e ranqueadas, a escolha das interfaces para encaminhar um pacote de interesses depende da política de encaminhamento adotada pelos roteadores. Essas políticas podem ter regras baseadas em diferentes fatores e podem ser implantadas independentemente por diferentes unidades organizacionais. Por exemplo, se a política de encaminhamento estabelece que somente as interfaces com a menor latência devem ser utilizadas, então este processo será baseado no RTT estimado associado as mesmas. Por outro lado, certa política poderia estabelecer uma preferência por um determinado vizinho. Assim, uma grande porcentagem dos pacotes de interesse seria encaminhada pelas interfaces conectadas direta ou indiretamente a este vizinho.

Por fim, [Yi et al. 2012] definem uma possível estratégia de encaminhamento que utiliza as informações contidas na FIB e na PIT para a escolha das interfaces que farão parte do processo de encaminhamento. As decisões tomadas dependem do dado recebido ser um novo pacote de interesse, um pacote de interesse retransmitido ou um NACK de Interesse.

**Pacote de Interesse Novo:** Quando um roteador recebe um novo pacote de interesse, ele realiza o procedimento normal, descrito na Seção 3.2.4. Se o interesse precisar ser encaminhado, então a interface verde melhor ranqueada deve ser a escolhida. Caso não existam interfaces com essa característica, a interface amarela melhor ranqueada deve ser utilizada.

**NACK de Interesse:** após receber um NACK de Interesse, significando que o interesse encaminhado pela interface $x$ não pode ser atendido, o roteador deverá tentar reencaminhar o interesse através da interface melhor ranqueada, mas diferente de $x$. Isso significa que o roteador irá explorar diferentes interfaces para tentar obter o conteúdo. Entretanto, este processo não deve tomar muito tempo, já que o conteúdo requisitado pode realmente estar indisponível. Assim, quando um pacote de interesse é encaminhado pela primeira vez, o roteador inicia um temporizador, chamado de temporizador de exploração, e define seu tempo de expiração baseado numa estimativa média de amostras de RTT. Dessa maneira, o processo de exploração por interfaces alternativas é iniciado após a recepção de um NACK de interesse e termina quando o roteador obtém sucesso ou quando o tempori-

 

zador de exploração expira.

**Pacote de Interesse Retransmitido:** Um interesse é considerado retransmitido se existe uma entrada correspondente na PIT para o mesmo e seu *nonce* não está contido na lista de *nonces* dessa entrada. Quando um interesse retransmitido é recebido por um roteador, antes da expiração do temporizador de exploração, este não é encaminhado. Se o temporizador já tiver expirado, ele será encaminhado através de uma interface verde ou amarela, diferente daquela utilizada por seu antecessor.

Apesar das interfaces classificadas como verdes serem utilizadas preferencialmente no processo de encaminhamento de pacotes de interesse, também é importante sondar periodicamente as interfaces classificadas como amarelas, com o objetivo de descobrir outros caminhos com melhor desempenho. Por exemplo, um caminho pode se tornar disponível após a recuperação de falhas de enlace ou mesmo um caminho anteriormente classificado como amarelo pode se tornar verde após o aparecimento de um conteúdo em *cache* mais próximo do consumidor. Assim, um roteador CCN sonda proativamente as interfaces classificadas como amarelas, encaminhando uma cópia de um interesse para cada uma delas. Tal sondagem prove informações sobre a disponibilidade e desempenho de caminhos alternativos, porém pode resultar na recuperação de dados duplicados. Esse problema pode ser controlado limitando-se a frequência da sondagem.

Além de permitir uma melhor utilização de recursos da rede, o encaminhamento adaptativo de pacotes de interesse também fornece um mecanismo natural de proteção contra alguns tipos de ataque de negação de serviço, conforme tratado na Seção 3.4.4.

## 3.3. Questões de Segurança da Arquitetura CCN

Para se obter um conteúdo na Internet atual, seja uma página web, sejam dados multimídia, é preciso primeiramente determinar o endereço IP do repositório de armazenamento deste conteúdo. Em geral, apenas o nome desse repositório é conhecido e, portanto, são necessários mecanismos de resolução de nomes, como o DNS (*Domain Name System*). Em seguida, o cliente estabelece uma comunicação fim-a-fim com esse repositório e requisita o conteúdo através de algum protocolo de aplicação conhecido. Caso o cliente deseje ter a certeza que o conteúdo recebido é realmente aquele pretendido, deve garantir que o servidor DNS e o repositório do conteúdo sejam confiáveis, que não tenham sido comprometidos e que o canal por onde o conteúdo foi transmitido é seguro. Desta forma, a tentativa de garantir a segurança de uma simples página web, por exemplo, implica na necessidade de garantir a segurança de diversos componentes da infraestrutura da rede. Essa complexidade é reflexo da falta de planejamento de segurança durante a fase de projeto da Internet. Por outro lado, a CCN adota a segurança como um de seus pilares e fornece mecanismos que simplificam a garantia dos requisitos de segurança como autenticidade, integridade, confidencialidade e disponibilidade dos dados. A seguir, são apresentados os mecanismos usados pela CCN para prover tais requisitos.

### 3.3.1. Segurança de Conteúdos Nomeados

Na CCN, um conteúdo pode ser armazenado tanto na fonte (publicador) quanto no *cache* de qualquer nó da rede. Como esses repositórios podem estar sob controle de

 

usuários maliciosos, a segurança do conteúdo deve ser independente de sua localização física. Essa característica facilita a implementação de modelos de segurança, pois agora para garantir a segurança de um conteúdo obtido por um usuário, não é mais necessário confiar em toda a infraestrutura envolvida em sua obtenção. Para isso, deve-se fornecer ao usuário condições para verificar se o conteúdo recebido é uma cópia fiel do conteúdo divulgado pelo publicador (integridade ou validade), quem é o publicador do conteúdo recebido (autenticidade ou proveniência) e se o conteúdo recebido condiz semanticamente com a requisição feita pelo usuário (relevância).

A melhor maneira de nomear os conteúdos de modo a garantir sua integridade, proveniência e relevância ainda é alvo de grande debate na literatura. Apesar dos envolvidos diretamente no projeto da arquitetura CCN adotarem o modelo de nomeação hierárquico e afirmarem que este é o mais seguro [Smetters e Jacobson 2009], outros argumentam a favor da utilização dos nomes auto certificados [Ghodsi et al. 2011b]. As próximas seções descrevem essas duas abordagens.

#### 3.3.1.1. Nomes Auto-certificados

Os nomes auto certificados são aqueles construídos a partir de estruturas criptográficas que permitem uma forte correspondência entre os conteúdos e seus nomes. Existem duas categorias principais de nomes auto certificados, os nomes verificados por *hash* (utilizados em conteúdos estáticos) [Fu et al. 2002] e por chave criptográfica (utilizados em conteúdos mutáveis) [Popescu et al. 2005].

Os nomes verificados por *hash* são gerados a partir do *hash* do conteúdo, como ilustrado na Figura 3.6. Dessa forma, se o conteúdo for corrompido em qualquer *cache* na rede ou durante seu caminho até o consumidor, esse fato poderá ser facilmente percebido pelo consumidor. Para tanto, basta calcular o *hash* do conteúdo e comparar o resultado com o nome do mesmo. Uma correspondência indica um conteúdo íntegro. Apesar de garantir a integridade dos conteúdos, esse esquema de nomeação não permite que o usuário possua garantias sobre a sua proveniência, já que os conteúdos não são assinados. Entretanto, uma vez que os usuários requisitam os conteúdos pelo nome, ou seja, o *hash* dos conteúdos, é esperado que o conteúdo obtido seja aquele pretendido pelo usuário, já que, desconsiderando as possíveis colisões, não existem dois conteúdos cujos *hashes* sejam iguais. Dessa forma, a real proveniência desse conteúdo se torna de pouca relevância. Apesar da CCN originalmente não adotar esta abordagem, a utilização de nomes verificados por *hash* é discutida em [Baugher et al. 2012].

Os nomes verificados por chave são formados pelo *hash* da chave pública do publicador. Além disso, o conteúdo também possui metadados contendo sua assinatura digital, gerada a partir da chave privada do publicador. Uma vez que, o nome do conteúdo é o *hash* da chave pública do publicador, estes dois entes estão fortemente ligados. A chave pública, por sua vez, está relacionada ao conteúdo, devido à assinatura digital presente no mesmo (metadado). Como resultado, este esquema de nomeação estabelece uma forte relação (indireta) entre o conteúdo e seu nome.

Além da integridade dos dados, os nomes verificados por chave também garantem a proveniência do conteúdo, através da verificação de sua assinatura digital. É interessante

 

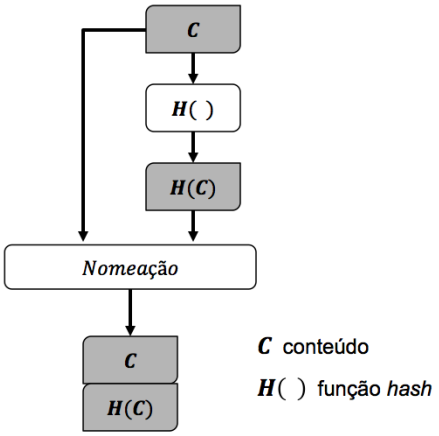


**Figura 3.6. Processo de formação de nomes verificados por *hash*.**

observar que apesar dos nomes verificados por chave utilizarem criptografia assimétrica para a assinatura digital dos conteúdos, não é necessário recorrer à sistemas de infraestrutura de chaves públicas como o X.509 [Myers et al. 1999] ou o PGP [Zimmermann 1995]. Uma vez que os conteúdos são recuperados pelo nome, e este contém uma representação inalterável (*hash*) da chave pública do publicador, o usuário especifica o publicador do conteúdo desejado. Para verificar a assinatura digital do conteúdo obtido, o usuário pode obter a chave pública do publicador através de meios externos e verificar sua integridade através da comparação do *hash* dessa chave com o nome do conteúdo. A Figura 3.7 ilustra o processo de formação de nomes verificados por chave.

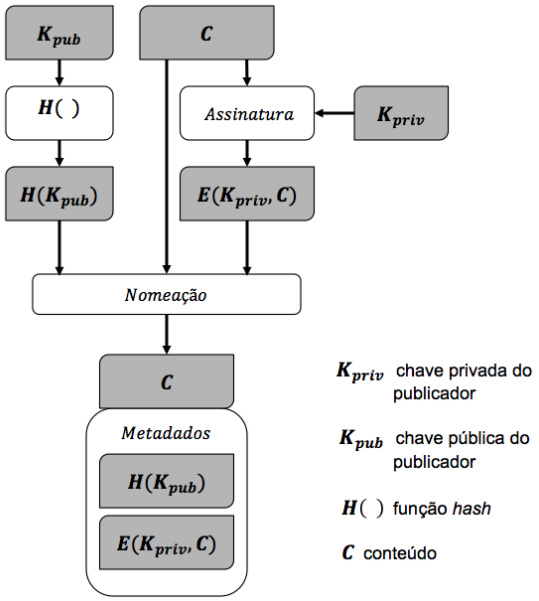


**Figura 3.7. Processo de formação de nomes verificados por chave.**

Outra questão importante com relação aos nomes auto certificados é sua unicidade global. Os nomes verificados por *hash* são únicos, pois uma função *hash* aplicada à

 

conteúdos diferentes sempre produzirá um resultado diferente[2].

Por outro lado, os nomes verificados por chave calculam o *hash* sobre a chave pública do publicador. Para que um publicador possa publicar vários conteúdos diferentes com nomes globalmente únicos, seriam necessários tantos pares de chaves pública-privada quanto fosse o número de conteúdos publicados pelo mesmo. Essa situação introduz uma maior complexidade para a geração e gerenciamento de chaves. A utilização de um nome verificado por chave da forma $Hash(Chavepub_P)||Label_C$, é uma alternativa a esse problema, onde $Hash(Chavepub_P)$ é o *hash* da chave pública do publicador, $||$ representa a operação de concatenação e $Label_C$ é um conjunto de bits qualquer, único para cada conteúdo publicado pelo publicador P. Por exemplo, uma ideia é fazer com que $Label_C$ seja o *hash* do conteúdo C, o que resultaria em nomes globalmente únicos do tipo $Hash(Chavepub_P)||Hash(C)$.

De uma maneira geral, os nomes auto certificados são sequências binárias ilegíveis para os seres humanos. Dessa forma, é improvável que um usuário consiga lembrar ou compreender esses nomes. Essa restrição traz a necessidade da existência de um mecanismo de tradução de nomes. Assim como o DNS traduz um nome de domínio FQDN (*Fully Qualified Domain Name*) para um endereço IP, seria necessário um mecanismo que recebesse como entrada um nome compreensível por seres humanos e gerasse como saída um nome auto certificado. Analisando mais profundamente essa questão, é possível perceber que a necessidade de um mecanismo de tradução de nomes anula a segurança provida pelos nomes auto certificados. Como exemplo, suponha que um usuário queira obter da rede um conteúdo $C$. Ele então utiliza o mecanismo de tradução de nomes para obter o nome auto certificado correspondente a esse conteúdo. Nesse momento, o usuário pode requisitar tal conteúdo à rede utilizando tal nome. O conteúdo é então obtido e o usuário procede com as verificações de integridade e autenticidade do mesmo. Ao término do processo, caso todas as verificações tenham sido satisfeitas, o usuário estará certo que recebeu o conteúdo $C$ válido. Entretanto, caso o mecanismo de tradução de nomes seja comprometido, o usuário obterá um nome auto certificado para um conteúdo diferente daquele pretendido. Como para o usuário tal nome não tem qualquer significado, ele não saberá que foi enganado. Logo, o conteúdo obtido passará em todas as verificações, já que estará íntegro e a chave pública contida em seu nome foi aquela utilizada para assiná-lo, porém o conteúdo recebido não será aquele requisitado.

Para minimizar a falta de inteligibilidade dos nomes auto certificados, são propostos nomes com a forma $Hash(Chavepub_P)||Label_C$, onde o rótulo $Label_C$ é uma *string* legível por seres humanos e única dentro do conjunto de conteúdos publicados pelo publicador $P$ [Koponen et al. 2007]. Porém, mesmo com a inteligibilidade introduzida pelo rótulo, os nomes gerados ainda são ininteligíveis.

[2]Na realidade, devido a problemas de colisão, onde uma função *hash* aplicada a dados diferentes produz resultados iguais, pode ocorrer de conteúdos diferentes terem o mesmo nome verificado por *hash*. Apesar disso, a escolha de algoritmos de *hash* fortes, como o SHA-1 [Rd e Jones ], reduzem a probabilidade de ocorrência de colisões. Dessa forma, praticamente pode-se dizer que os nomes verificados por *hash* são globalmente únicos.

 

### 3.3.1.2. Abordagem da CCN

Ao contrário do que ocorre com os nomes auto certificados, o esquema de nomeação adotado pela CCN é compreensível por seres humanos e permite aos usuários um grande poder de expressão, como visto na Seção 3.2.1. Apesar disso, tais nomes não oferecem quaisquer informações para a verificação da integridade ou proveniência dos conteúdos. De acordo com o triângulo de Zooko [Wilcox-O'Hearn 2003], os nomes só podem ter, ao mesmo tempo, no máximo duas das seguintes propriedades: unicidade global, segurança e inteligibilidade, conforme ilustrado na Figura 3.8. Uma comparação entre os nomes auto certificados e o esquema de nomeação da CCN, com relação a essas três propriedades, é apresentado na Tabela 3.1. Deste modo, os nomes auto certificados são globalmente únicos, seguros e ininteligíveis. Já os nomes na CCN são globalmente únicos[3], inseguros e inteligíveis.

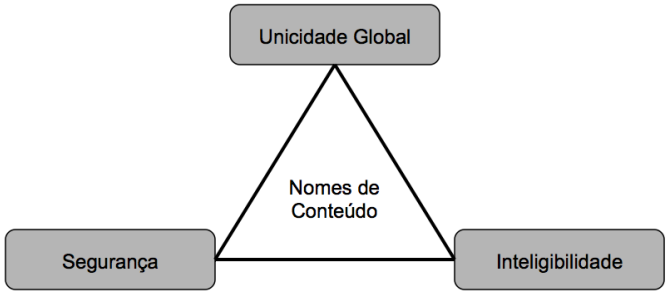


**Figura 3.8. Triângulo de Zooko.**

**Tabela 3.1. Propriedades dos esquemas de nomeação.**

| | Nomes Auto-certificados | Nomes na CCN |
|---|---|---|
| Unicidade Global | ✓ | ✓ |
| Segurança | ✓ | |
| Inteligibilidade | | ✓ |

A escolha dos nomes auto certificados, por conta de sua unicidade e segurança, gera fragilidades impostas pela falta de inteligibilidade dos nomes gerados. Por outro lado, a flexibilidade da CCN em gerar nomes inteligíveis e únicos confere a completa falta de segurança dos mesmos. Assim, para simplificar este cenário, a CCN divide o processo de requisição e obtenção de conteúdos nas funções de identificador, localizador e autenticador de conteúdo.

A função de identificador possibilita aos seres humanos especificar os conteúdos de seu interesse. A função de localizador permite que a rede localize um determinado conteúdo, onde quer que ele esteja. Já a função de autenticador determina se o conteúdo recebido é válido ou não. O esquema de nomeação auto certificado impõe que os nomes

[3]A unicidade global dos nomes na CCN depende de uma entidade externa que regule a atribuição de nomes aos conteúdos dos publicadores. Uma vez que os nomes de domínio são globalmente únicos, nomes de conteúdo construídos a partir dos nomes de domínio de seus publicadores também o serão.

 

cumpram todas essas funções ao mesmo tempo, gerando os problemas mencionados anteriormente. Na abordagem da CCN, os nomes desempenham as funções de identificador e localizador de conteúdos. A função de autenticador é delegada à assinatura digital do mapeamento entre o conteúdo e o seu nome. Quando um publicador deseja publicar um certo conteúdo, ele deve fazê-lo sob a forma da tripla $M_{(N,P,C)} = (N, C, Assin_P(N,C))$, onde N é o nome do conteúdo, C é o conteúdo em si e $Assin_P(N,C)$ é a assinatura do publicador, efetuada sob o mapeamento entre o nome e o conteúdo. Portanto, um usuário deve requisitar o conteúdo através de seu nome N obtendo o conteúdo C e a assinatura $Assin_P(N,C)$. Com isso, o usuário precisa obter a chave pública P do publicador para verificar a assinatura de forma a se certificar de que o conteúdo é íntegro e que ele foi realmente publicado pelo publicador detentor da chave pública P.

Um benefício obtido com este modelo de nomeação é permitir que o conteúdo seja armazenado em qualquer nó da rede. A obtenção de $M_{(N,P,C)}$ permite ao usuário verificar a integridade e autenticidade do conteúdo C e do mapeamento entre o nome N e conteúdo C. O conteúdo pode ser armazenado tanto no publicador quanto em quaisquer *caches* da rede, confiável ou não. Essa característica é de fundamental importância para a CCN, já que os *caches* são essenciais para a recuperação eficiente de conteúdos.

Outro benefício é que a abordagem da CCN não faz suposições a respeito do formato do nome N em questão. Diferentes aplicações podem possuir diferentes restrições quanto ao formato dos nomes utilizados. Com isso, é permitido que os nomes tomem qualquer forma, de acordo com as necessidades das aplicações, garantindo a segurança do mapeamento entre os conteúdos e seus nomes, o valor e o significado atribuídos pelo padrão de nomes utilizado.

Utilizando nomes auto certificados do tipo $Hash(Chavepub_P)||Label_C$, o usuário precisa decidir, a priori, de qual publicador P ele quer obter uma cópia de C. Utilizando a autenticação dos mapeamentos entre nomes e conteúdos, um usuário deve obter um conteúdo através de seu nome N e verificar se o publicador P que o publicou é aceitável. Ocorre que a primeira decisão é frequentemente mais difícil de tomar do que a segunda.

Em essência, assinar o mapeamento entre o conteúdo e seu nome é o mesmo que gerar um certificado digital para o conteúdo, assegurando que este realmente é íntegro e possui o nome especificado. Essa abordagem corresponde à mudança de um modelo no qual os nomes precisam carregar consigo informações de autenticação para um modelo onde o receptor do conteúdo decide se este é aceitável ou não. Porém, tal modelo é mais vulnerável a ataques de negação de serviço, uma vez que o usuário necessita primeiramente obter o conteúdo e somente depois decidir se o mesmo é ou não válido. Assim, adversários podem tentar fazer com que o usuário sempre receba conteúdos inválidos [Ghodsi et al. 2011b]. Outra abordagem seria exigir que os roteadores realizassem a verificação da validade dos conteúdos antes de armazenarem os mesmos em *cache*. O esquema de nomeação da CCN exige que exista uma cadeia de confiança que autentique o mapeamento entre a chave pública de um publicador e seu nome no mundo real, como tratado na Seção 3.3.2. Exigir que cada roteador percorra essa cadeia de confiança para verificar a autenticidade de uma chave pública antes de utilizá-la para verificar a assinatura de cada conteúdo, pode ser demais. Como consequência do excesso de processamento, os roteadores podem se tornar muito vulneráveis a ataques de negação de serviço, como

 

descrito na Seção 3.4.

### 3.3.2. Estabelecimento e Gerenciamento de Confiança

Como foi observado na Seção 3.3.1.1, os nomes auto certificados verificados por chave utilizam o *hash* da chave pública do publicador em sua formação. Dessa forma, implicitamente, ao especificar o nome do conteúdo na requisição, os usuários estão especificando também qual deve ser o publicador desse conteúdo. Nada impede, porém, que publicadores maliciosos utilizem a chave pública de outro publicador legítimo para nomear seus conteúdos. Para evitar fraude, os conteúdos também são assinados utilizando a chave privada, par da chave pública usada na nomeação. Assim, para conferir a autenticidade dos conteúdos é necessário seguir algumas etapas.

Primeiramente, obter a chave pública do publicador, sendo que a mesma pode estar presente no conteúdo como metadado. Em seguida, verificar se a chave pública obtida é realmente aquela utilizada para formar o nome do conteúdo. Para tanto, basta calcular o *hash* da chave pública e comparar o resultado com o nome do conteúdo. Após os passos iniciais, utiliza-se a chave pública obtida para verificar a assinatura do conteúdo. Caso essa verificação seja bem sucedida, então o conteúdo é considerado como íntegro e autêntico.

Para evitar que os usuários obtenham conteúdos inválidos, essa verificação poderia ser empregada no momento da publicação do conteúdo, permitindo à rede descartar publicações de conteúdos inválidos. Como resultado de todo esse processo, os usuários passam a confiar nos conteúdos recebidos, já que eles confiam que estes são íntegros e foram publicados pelo publicador esperado.

No esquema de nomeação adotado pela CCN, os nomes também identificam o publicador do conteúdo, porém não através da chave pública e sim pelo próprio nome do publicador. Por exemplo, seja `/br.uff/videos/intro.avi` um nome de conteúdo. Nesse caso, podemos supor que o publicador do conteúdo em questão tem o nome `/br.uff`. Quando o usuário requisitar este conteúdo, ele irá esperar obter como resposta um conteúdo publicado pela UFF e não pela UFRJ. Uma vez que esses nomes de conteúdo não possuem qualquer relação direta ou indireta com o conteúdo em si, a CCN utiliza a assinatura do mapeamento entre o conteúdo e seu nome para esse fim. Quando o conteúdo é recebido, o receptor deve obter a chave pública do publicador e utilizá-la para verificar a assinatura presente naquele conteúdo. No entanto, caso a verificação seja positiva, tudo que o receptor saberá é que a chave privada que faz par com a chave pública utilizada na verificação foi realmente aquela que assinou o mapeamento entre o conteúdo e seu nome. Entretanto, como não existe relação entre o nome do conteúdo e a chave pública utilizada na verificação, o receptor não pode ter certeza que esta chave pública pertence realmente ao publicador esperado.

Para criar uma relação indireta entre o nome e a chave pública do publicador, pode-se estabelecer um mapeamento seguro entre o nome do publicador e sua chave pública[4]. Esse mapeamento seguro é caracterizado por um certificado digital. Por consequência, o

[4]Como o nome do conteúdo tem um mapeamento direto para o nome do publicador que, por sua vez, possui um mapeamento direto para a sua chave pública, o nome do conteúdo passa a ter um mapeamento para a chave pública do publicador, devido a transitividade.

 

esquema de nomeação da CCN implica na utilização de um sistema de infraestrutura de chaves públicas. Dentre os possíveis sistemas (tais como o PGP, X.509 e SDSI/SPKI), aparentemente o mais indicado seria o SDSI/SPKI, já que os mesmos utilizam o conceito de espaço de nomes local, utilizado para nomear chaves públicas [Zhang et al. 2010]. Dessa forma, os usuários da CCN poderiam requisitar a chave pública de um publicador através do seu nome, da mesma forma que qualquer outro tipo de conteúdo é requisitado. Por exemplo, suponha que o publicador P tenha certeza que a chave pública $K_A^+$ pertença ao publicador A e resolva divulgar este fato na rede. Para tanto, P publica $M_{(N,P,C)} = (N, C, Assin_P(N,C))$, onde N é o nome nomeP/certs/nomeA e C é na verdade $K_A^+$. Uma vez que N contém o nome de A (nomeA), $M_{(N,P,C)}$ representa na verdade um certificado digital, onde P é a autoridade certificadora.

Uma alternativa ao SDSI/SPKI é o mecanismo de estabelecimento de confiança em chaves públicas de publicadores proposto por [Pournaghshband e Natarajan ], de forma a permitir que aplicações possam determinar quais chaves são confiáveis. Neste mecanismo os consumidores devem requisitar recomendações sobre a chave pública de publicadores a todos os membros de sua comunidade de confiança. Uma comunidade de confiança é formada por pessoas que o usuário conhece através de algum relacionamento no mundo real. Assim, uma política local, definida pelo próprio consumidor, é utilizada para avaliar a consistência das respostas e consequentemente, a confiança depositada na chave de um determinado publicador.

Até o momento, os mecanismos fornecidos pela CCN permitem que um usuário possa verificar se um conteúdo recebido é válido e se o mesmo foi publicado pelo publicador esperado. Entretanto, para requisitar um conteúdo, o usuário precisa confiar, a priori, no publicador cujo nome consta no nome do conteúdo. Por exemplo, é fácil confiar que os vídeos encontrados no domínio `/br.uff` foram publicados pela UFF, já que é sabido, através de meios externos, que tal domínio realmente pertence à UFF. Porém, para confiar em um publicador completamente desconhecido a CCN propõe um mecanismo conhecido como segurança baseada em evidências, que visa auxiliar os usuários no processo de estabelecimento de confiança nos publicadores [Smetters e Jacobson 2009].

Para entender o conceito de segurança baseada em evidência, suponha uma situação em que um publicador $A$ decide publicar uma página web na rede. O publicador $A$ também possui uma relação de confiança com um publicador $B$, que por sua vez, publica um determinado vídeo popular. Suponha ainda que um determinado usuário confie no publicador $A$, mas queira receber o vídeo popular cujo publicador seja $B$. Este usuário pode confiar no publicador $B$ e por consequência confiar no vídeo publicado por ele, pois o publicador $A$ disponibiliza um *link* na página web apontando para o vídeo publicado pelo publicador $B$. Assim, como o usuário em questão confia no publicador $A$, ele pode seguir esse *link* e obter o vídeo publicado por $B$. Para garantir que realmente foi o publicador $A$ quem criou o *link* para o vídeo, esse *link* precisa ser assinado por $A$. No contexto da CCN, um *link* seguro é um mapeamento $N \rightarrow (N', H(P'))$, onde $N$ é o nome do *link*, $N'$ é o nome do conteúdo para o qual o *link* aponta e $H(P')$ é o *hash* da chave pública do publicador do conteúdo $N'$. Quando um publicador publica o *link* $N$, ele está dizendo, na verdade, que para ele, o conteúdo referente ao nome $N$ é aquele que o publicador $P'$ nomeou como $N'$. Se cada conteúdo obtido possuir *links* como os descritos, os usuários

 

poderiam interpretá-los como evidências de que os conteúdos apontados são seguros.

Para outro exemplo da ideia de segurança baseada em evidências, suponha que um publicador honesto publicou um conteúdo popular com o nome $N^{'}$. Para enganar os usuários, um publicador malicioso publicou um conteúdo diferente com o mesmo nome $N^{'}$. Apesar disso, se os usuários utilizarem as evidências (*links*) contidas em conteúdos confiáveis obtidos anteriormente, poderão verificar que a maioria dos publicadores confiáveis indicam, através dos *links*, que para ser confiável, o conteúdo de nome $N^{'}$ tem que ter sido publicado por um publicador honesto. Apesar de interessante, essa proposta cria relações de confiança estáticas, já que os *links* estão presentes nos conteúdos e esses, uma vez obtidos, não são mais alterados por seus publicadores. Dessa forma, se um publicador confiável passa a ser malicioso, os usuários podem ser levados a confiar erroneamente nos mesmos, já que as evidências presentes nos conteúdos obtidos por eles ainda apontam neste sentido. Além disso, assim como é importante um mecanismo de revogação de certificados digitais para os sistemas PKI, o processo de revogação de um *link* também é importante para o modelo de segurança baseada em evidências. Entretanto, tal modelo não prevê nenhum mecanismo para tanto.

### 3.3.3. Controle de Acesso à Conteúdos

Tradicionalmente, sistemas como o Kerberos [Neuman et al. 1993] são utilizados para autenticar e prover aos usuários credenciais de acesso a dados e serviços. Nesses sistemas, geralmente um servidor de autenticação verifica a identidade dos usuários através de um processo de *login* e senha. Caso essa identidade seja comprovada, o usuário recebe credenciais para acessar os dados ou recursos a ele permitidos. Para que esse mecanismo funcione, é preciso que um servidor intercepte as requisições de acesso a dados feitos pelos usuários e avalie se deve concedê-las ou não. Na CCN, entretanto, os conteúdos são armazenados em qualquer nó da rede, inclusive em roteadores, e os usuários os requisitam através do nome do conteúdo e não diretamente para um servidor específico. Dessa maneira, montar um sistema como o Kerberos na CCN seria muito difícil. O controle de acesso aos conteúdos deve então ser implementado em termos do próprio conteúdo, independentemente da localização do mesmo na rede. Para alcançar esse objetivo, é possível empregar a criptografia simétrica desses conteúdos de forma que, somente o usuário ou grupo de usuários que conheçam a chave poderiam acessar o conteúdo. Para distribuir uma chave simétrica, o publicador do conteúdo cujo acesso é controlado, poderia criptografá-la utilizando a chave pública do usuário ao qual o acesso ao conteúdo será permitido. Em seguida, essa chave simétrica criptografada poderia ser publicada na rede, através de um nome específico, da mesma maneira que um conteúdo qualquer é publicado na CCN. O usuário poderia então recuperar essa chave simétrica através do nome e utilizar sua chave privada para decriptá-la.

### 3.3.4. CCN e os Ataques de Negação de Serviço Tradicionais

A disponibilidade está entre os principais serviços de segurança de redes de computadores [Stallings 2006]. A recomendação X.800 da *International Telecommunication Union* (ITU), *Security architecture for Open Systems Interconections* (OSI), define a disponibilidade como ”a propriedade de estar acessível e utilizável mediante requisição de uma entidade autorizada” [Telegraph e Committee 1991]. A disponibilidade pode ser de-

 

finida de uma forma menos abrangente como a capacidade de usar a informação ou um recurso desejado [Bishop 2003]. Um complemento desta definição, caracteriza a disponibilidade como a capacidade de usar a informação ou recurso desejado de forma confiável e oportuna [Abliz 2011]. Em uma descrição mais formal, de acordo com o RFC 2828 - *Internet Security Glossary*, a disponibilidade é definida como sendo a propriedade de um sistema, ou de um recurso do sistema, de ser acessível e utilizável sob demanda por uma entidade autorizada, de acordo com suas especificações de desempenho [Shirey 2000]. Assim, um sistema se torna disponível caso ofereça seus serviços sempre que solicitado por usuários legítimos. Um serviço pode ser um sistema de busca de páginas web, de venda de produtos ou de troca de mensagens entre dois nós da rede [Laufer et al. 2005]. Logo, um serviço pode ser definido como uma determinada função de um nó ou infraestrutura da rede que visa atender a uma demanda específica. Neste contexto, a CCN, por ser uma rede basicamente de compartilhamento de dados, deve ser capaz de entregar os conteúdos requisitados para os usuários, caso estes existam na rede.

Para impedir o acesso de usuários a determinados serviços, adversários geralmente implementam ataques de negação de serviço (*Denial of Service* - DoS). Estes ataques normalmente exploram a limitação de recursos em servidores, como processamento e armazenamento, utilizados na execução de um serviço. Uma vez atingido tal limite, o serviço passa a estar indisponível para novas requisições. Atualmente, os ataques DoS mais comuns são aqueles que tem como alvo serviços específicos (por exemplo, servidores web). Nesses casos, o objetivo do adversário é enviar uma quantidade de requisições maior do que a aplicação pode tratar. Como resultado, as aplicações passam a utilizar todos os recursos disponíveis para atender as requisições falsas geradas pelo adversário, enquanto as requisições de usuários legítimos são descartadas. Uma variação deste ataque, o *Distributed Denial of Service* - DDoS, pode ser utilizado para alcançar o mesmo objetivo, mas exigindo menos recursos do adversário. Para este ataque, os adversários precisam reunir um grande número de máquinas que o auxiliem a enviar as requisições que causarão a indisponibilidade do serviço alvo. Para tanto, o método mais comum é o adversário utilizar um ataque secundário, como o *phishing*, para propagar na rede um *software* malicioso que tem a capacidade de receber e executar remotamente as ordens emitidas pelo adversário. O conjunto de tais máquinas controladas por um adversário é conhecido como *Botnet* e cada máquina nesse conjunto é chamada de Zumbi. Doravante, será utilizada a sigla DoS tanto para indicar o ataque de negação de serviço centralizado quanto o distribuído (DDoS).

O crescente número de ataques DoS bem sucedidos evidencia a inabilidade da Internet atual em garantir a disponibilidade de seus serviços. Dessa forma, propostas de arquiteturas substitutas precisam ter em seu projeto mecanismos que, no mínimo, mitiguem os ataques DoS tradicionais. A CCN é uma dessas propostas e no restante dessa seção, será discutido como algumas de suas características são capazes de inviabilizar os ataques DoS de reflexão, esgotamento de largura de banda, buraco negro e falsificação de prefixos.

 

#### 3.3.4.1. Ataques de Reflexão

Na ausência de uma *botnet* o adversário pode tentar realizar um DDoS enganando alguns nós da rede (alvos intermediários), fazendo com que eles ataquem o alvo real no lugar do adversário. Para tanto, o adversário deve ser capaz de gerar pacotes IP cujo endereço de origem é o endereço IP do alvo real (técnica conhecida como IP *spoofing*. Estes pacotes são então enviados aos alvos intermediários, que por sua vez, deverão enviar respostas para os pacotes recebidos. Uma vez que o endereço de origem dos pacotes recebidos pelos alvos intermediários é o do alvo real, todas as respostas geradas pelos alvos intermediários serão enviadas diretamente para ele. Para que este tipo de ataque seja efetivo, a quantidade de dados utilizada pelo adversário deve ser menor do que a quantidade de dados recebida pelo alvo real.

A CCN é geralmente capaz de mitigar este tipo de ataque devido aos pacotes de dados sempre retornarem pelo caminho inverso do interesse que os originou. Note, entretanto, que os roteadores de conteúdo podem realizar o *broadcast* de pacotes de interesse por todas as suas interfaces. Dessa forma, como sugerido em [Gasti et al. 2012], o único ataque de reflexão efetivo que poderia ser implementado na CCN seria aquele onde o adversário está na mesma sub-rede que a vítima. Dessa forma, ele poderia enviar pacotes de interesse em *broadcast* para todas as suas interfaces (forjando o endereço de camada 2 da vítima) esperando que múltiplas cópias do mesmo conteúdo fossem direcionadas para a vítima. Entretanto, a CCN implementa um mecanismo de supressão que impede roteadores no mesmo domínio de *broadcast* de propagarem o mesmo conteúdo mais de uma vez.

#### 3.3.4.2. Esgotamento de Largura de Banda

Utilizando uma *Botnet*, um adversário é capaz de gerar uma grande quantidade de tráfego IP, geralmente utilizando os protocolos UDP, TCP ou ICMP. Esse tráfego é direcionado a uma vítima específica, com o objetivo de esgotar seus recursos de rede e causar a sua indisponibilidade para se comunicar.

Uma vez que não existe maneira de especificar o destinatário de um pacote de interesse, o adversário só pode tentar atacar o publicador do conteúdo, já que os pacotes de interesse são encaminhados para ele. Entretanto, após a primeira recuperação do conteúdo requisitado, ele poderá ser obtido diretamente de diversos *caches* na rede. Esta característica reduz drasticamente a eficiência deste tipo de ataque, já que nem todos os interesses enviados atingirão o publicador.

#### 3.3.4.3. Falsificação de Prefixo e Buraco Negro

Em um ataque de falsificação de prefixo, o adversário fornece informações falsas de roteamento com o objetivo de atrair a maior quantidade de tráfego para si. Dessa maneira, o adversário pode implementar o chamado buraco negro, onde ele simplesmente descarta todos os pacotes recebidos. A Internet atual é bastante vulnerável a este tipo de ataque em conjunto, já que os roteadores não são capazes de detectá-lo.

 

A CCN é resistente a este ataque já que os roteadores tem acesso a uma quantidade muito maior de informações, o que permite a detecção de anomalias no processo de recuperação de conteúdos. Em primeiro lugar, a correspondência entre um pacote de interesse e seu respectivo pacote de conteúdo permite aos roteadores gerarem uma estatística sobre a quantidade de interesses não atendidos por prefixo. Dessa forma, o roteador pode detectar uma falsificação de prefixo e escolher outra interface, de acordo com sua camada de estratégia, para encaminhar os interesses para este prefixo. Além disso, uma vez que o processo de roteamento de pacotes de interesse é livre de *loops*, a CCN pode encaminha-los por múltiplas interfaces ao mesmo tempo. Dessa forma, mesmo que exista um buraco negro em um determinado caminho, interesses que sigam por caminhos diferentes poderão ser atendidos pelo publicador ou, eventualmente, por algum *cache*.

### 3.3.5. Privacidade de Usuários

Para os usuários, saber que um conteúdo é íntegro, autêntico e confiável é importante, mas não é suficiente. Uma vez que o roteamento de pacotes de interesse é realizado através de uma comparação de prefixos mais longos nos nomes de conteúdo, tais nomes não podem ser completamente ocultados da rede. Adicionalmente, os nomes de conteúdo na CCN são compreensíveis por seres humanos e como eles são utilizados para requisitar os conteúdos, devem possuir uma relação semântica com o conteúdo em si. Isso significa que, qualquer adversário que monitore a rede e capture os pacotes de interesse será capaz de identificar o conteúdo que está sendo requisitado. Isso caracteriza uma fragilidade na privacidade do sistema, conforme tratado na Seção 3.4.

Apesar deste problema, a CCN oferece um maior nível de privacidade em relação à Internet atual. A captura de pacotes de interesse revela o que está sendo requisitado, mas não quem está requisitando o conteúdo, já que tais pacotes não possuem informações de origem ou de destino, conforme a Figura 3.3. Com o objetivo de obter um nível ainda maior de privacidade, [Jacobson et al. 2009] discutem a possibilidade de criptografar parte dos componentes presentes no nome dos conteúdos. Para que esta técnica tenha sucesso, a criptografia do nome deve ser realizada considerando a possibilidade de inexistência em *cache* do conteúdo com o nome criptografado. Assim, os interesses ainda poderiam ser roteados até a fonte, que por sua vez enviaria o conteúdo requisitado no caminho inverso do interesse. Para tanto, é importante que a porção do nome de conteúdo referente ao nome do publicador seja deixada em texto claro, já que todos os nomes de conteúdo publicados por um publicador podem ser agregados nas tabelas de roteamento dos roteadores da rede.

A exemplo, suponha que o publicador `/uff.br` publique os seguintes conteúdos: `uff.br/ic/cursos/bd/aulas/aul1.mp4`, `uff.br/ic/cursos/prog1/aulas/aul5.mp4` e `uff.br/ic/cursos/dados/aulas/aul8.mp4`. Nos roteadores mais distantes do publicador, as rotas para esses três conteúdos podem ser sumarizadas em apenas uma, ou seja, `/uff.br`. Dessa forma, mesmo que os componentes após `/uff.br` estejam criptografados, os pacotes de interesse para estes conteúdos ainda poderão ser roteados pela rede. Segundo [DiBenedetto et al. 2012], mesmo com a criptografia de parte dos componentes, os nomes ainda revelam muitas informações, como a identidade do publicador do conteúdo.

 

Por fim, se o adversário controlar o roteador de primeiro salto da vítima, este saberá o que está sendo requisitado, devido ao nome compreensível do conteúdo, e também quem está requisitando esse conteúdo, já que o adversário terá acesso a tabela PIT do roteador e saberá por qual interface o pacote de interesse foi recebido. A Seção 3.4 discute alguns tipos de ataques que exploram o vazamento de informações nos nomes de conteúdos na CCN e também algumas propostas encontradas na literatura para mitigar tais ataques.

## 3.4. Ameaças à CCN e Propostas de Contramedidas

Como visto na Seção 3.3, a CCN provê diversos mecanismos com o objetivo de garantir a integridade, autenticidade e disponibilidade dos conteúdos. Apesar desses mecanismos constituírem um importante avanço em relação à Internet, eles ainda podem ser explorados para adaptar ataques tradicionais da Internet atual para CCN, como negação de serviço, poluição de *cache* e *cache snooping*. Esta seção aborda estas ameaças à arquitetura CCN e as propostas de contramedida já definidas na literatura. Além disso, novas ameaças como a censura de conteúdos e a análise comportamental dos usuários também serão tratadas nesta seção.

### 3.4.1. Censura de Conteúdos

Para descrever este ataque, considere que o adversário não seja um usuário, mas sim um país ditatorial com o controle de todo o núcleo da rede. Isso significa que, é possível para o suposto adversário monitorar todas as atividades de rede em seu domínio geográfico. Dessa maneira, pacotes de interesse e de dados que trafegam pela rede podem ser interceptados e analisados pelo adversário. Suponha também que o adversário queira proibir a distribuição em massa de alguns conteúdos alvo e também queira determinar quais usuários os estão requisitando. Além disso, o ataque deve ocorrer em tempo real, ou seja, a censura de conteúdos deve ocorrer ao mesmo tempo em que esses conteúdos (ou interesses para esses conteúdos) trafegam na rede. Neste contexto,[Arianfar et al. 2011] propõem dois possíveis tipos de ataque:

> **Ataque baseado em lista:** o adversário possui uma lista negra de nomes de conteúdos a serem censurados. Quando pacotes de interesse ou dados são capturados, o adversário utiliza o nome de conteúdo contido no pacote para fazer um *lookup* na lista e verificar se existe alguma correspondência. Se existir, o adversário saberá que um determinado usuário, ou grupo de usuários, está acessando um conteúdo classificado como sensível e poderá tomar medidas ofensivas contra o mesmo.
>
> **Ataque de análise de conteúdo:** neste ataque, o adversário analisa pacotes de dados capturados em busca de características que os identifiquem como passíveis de censura.

Dado que, o objetivo do adversário é descobrir a origem de requisições para conteúdos, e os pacotes de interesse e dados não transportam informações sobre a origem da requisição, então, de que forma o adversário poderia saber "quem"pediu "o que"? Note que foi definido que o adversário possui o domínio abrangente da infraestrutura da rede.

 

Dessa forma, basta o monitoramento do tráfego que passa pelos roteadores de borda, para saber a origem de uma requisição.

Como contramedida para os ataques baseados em lista e de análise de conteúdo, discutidos nesta seção, [Arianfar et al. 2011] propõem um mecanismo que aumenta a privacidade, mesmo em face de um adversário global, ou seja, com controle pleno da rede. Para tanto, primeiramente o conteúdo a ser requisitado é dividido em $n$ blocos, da seguinte forma: $T = t_1, t_2, ..., t_n$. Esses blocos então são reordenados aleatoriamente por uma função $r(.)$. Logo após, escolhe-se um conteúdo, conhecido como *cover*, que é dividido em $m$ blocos, da seguinte forma: $C = c_1, c_2, ..., c_m$. Tais blocos também são reordenados aleatoriamente pela função $r(.)$. Tanto os usuários quanto os adversários conhecem os nomes de ambos os conteúdos. Em seguida, para cada $k$-tupla formada por blocos do conteúdo e do *cover* é feito um ou-exclusivo entre os $k$ elementos. Se $k = 2$ e os blocos considerados são $r(t_1), r(t_2), r(c_1)$ e $r(c_2)$, o publicador deverá calcular e publicar os *chunks* $r(t_1) \oplus r(t_2)$, $r(t_1) \oplus r(c_1)$, $r(t_1) \oplus r(c_2)$, $r(t_2) \oplus r(c_1)$, etc. Para obter um conteúdo, deve-se recuperar em determinado conjunto de *chunks* cuja execução da operação de ou-exclusivo entre todos os elementos do conjunto retorne o conteúdo desejado.

Os nomes dos blocos aleatórios e dos *chunks* gerados são computados através da seguinte metodologia:

- O nome $n(t,i)$ para o bloco $t_i$ é $n(t,i) = H(H(T),i)$, onde $H(.)$ é uma função *hash* bem conhecida.
- O nome dos *chunks* é calculado executando-se a função *hash* $H(.)$ dos nomes dos blocos constituintes. Assim, o nome do *chunk* $r(t_1) \oplus r(c_1)$ é $H(n(t,1), n(c,1))$.

A força dessa abordagem está baseada no fato de que se um adversário capturar pacotes de interesse para *chunks* específicos, ele não saberá a qual conteúdo esses *chunks* se referem. Isso dificulta o ataque baseado em lista. Muito embora, o adversário possa capturar vários *chunks*, conseguir combinar todos os possíveis *chunks* para obter o conteúdo original é computacionalmente mais difícil para o adversário do que para os usuários.

### 3.4.2. Análise Comportamental

Suponha que exista um *Internet Service Provider* - ISP que deseja monitorar os padrões de tráfego de seus usuários para extrair informações comportamentais que poderiam ser vendidas para empresas para a realização de propaganda direcionada. Assim como no caso do adversário com domínio de toda a infraestrutura da rede, descrito na Seção 3.4.1, o ISP terá controle sobre os roteadores de primeiro salto de seus usuários e poderá obter informações de "quem"está requisitando "o que". É possível realizar ainda uma variação desse ataque onde o adversário não é um ISP, mas sim um usuário que está compartilhando uma rede sem fio aberta com a vítima. Nesse contexto, o adversário também poderia saber "quem"está requisitando "o que", simplesmente configurando sua interface de rede para operar em modo promíscuo.

Na realidade, a análise comportamental é uma especificação de um ataque mais geral, onde um adversário local, ou seja, um usuário ou ISP que consiga monitorar as

 

requisições de outros usuários na rede local deseja saber que conteúdos um determinado usuário está requisitando. Uma vez que o adversário é local, ele não controla a infraestrutura em múltiplos ASes. Para mitigar esse tipo de ataque, [DiBenedetto et al. 2012] propõem o ANDANA, um mecanismo baseado no conceito de roteamento cebola, assim como o Tor [Dingledine et al. 2004].

Antes de passar para os detalhes do funcionamento do Andana, é interessante entender como funciona um mecanismo genérico de roteamento cebola. Inicialmente, o usuário deve escolher um conjunto de roteadores habilitados para o roteamento cebola. Essa escolha pode ser feita a partir de uma lista obtida através de algum diretório na rede. Cada roteador possui um par de chaves pública-privada. Esses roteadores, agrupados em certa ordem, formam um caminho inicial por onde a mensagem do usuário deve passar. O usuário sabe qual é a ordem dos roteadores nesse caminho inicial e criptografa a mensagem sucessivamente utilizando a chave pública de cada roteador, mas na ordem inversa da posição de cada roteador no caminho. Isto é, a última criptografia é feita com a chave pública do primeiro roteador no caminho, a penúltima criptografia é feita com a chave pública do segundo roteador, e assim sucessivamente. A mensagem criptografada enviada pelo usuário é então decriptada, camada a camada, conforme passa pelos roteadores cebola. Por fim, a mensagem é encaminhada ao seu destino em texto claro. A Figura 3.9 ilustra as camadas do roteamento em cebola.

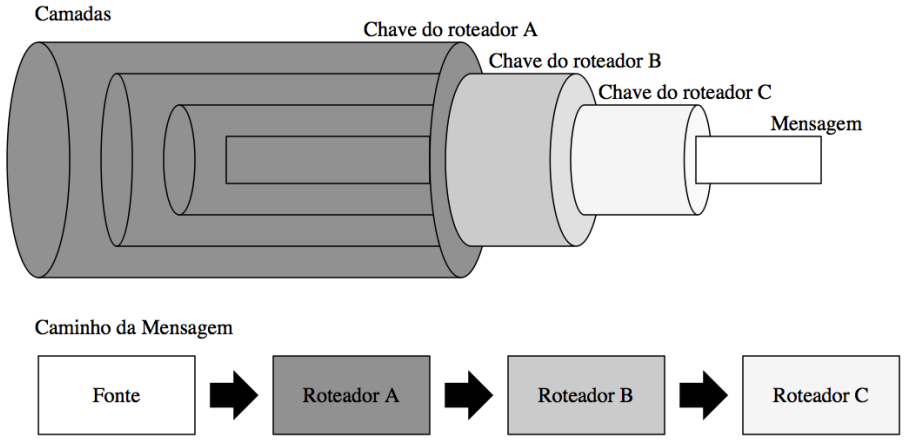


**Figura 3.9. Estrutura de um mecanismo genérico de roteamento em cebola.**

O funcionamento do Andana é simples e está ilustrado na Figura 3.10. Antes de enviar quaisquer interesses, os usuários devem escolher dois, entre vários roteadores habilitados para o Andana [5]. A maneira de fazer essa escolha é livre, mas é preferível que os roteadores sejam escolhidos uniformemente, já que assim todos os roteadores tem a mesma probabilidade de serem escolhidos, garantindo um maior anonimato [DiBenedetto et al. 2012]. Os dois roteadores escolhidos formam então um circuito efêmero que é utilizado para transportar apenas um ou alguns poucos interesses criptografados. Tal circuito é desfeito quando o conteúdo para um dado interesse é recebido ou

[5]Roteadores habilitados para o Andana são apenas roteadores de conteúdo da CCN que executam os procedimentos definidos pelo Andana

 

quando um pequeno período de tempo se esgota. O primeiro roteador nesse circuito é conhecido como Roteador de Entrada - RE e o segundo como Roteador de Saída - RS. Para dificultar a possibilidade de adversários relacionarem a entrada no RE com a saída no RS, os dois roteadores pertencentes ao circuito devem ser, preferencialmente, de domínios administrativos diferentes.

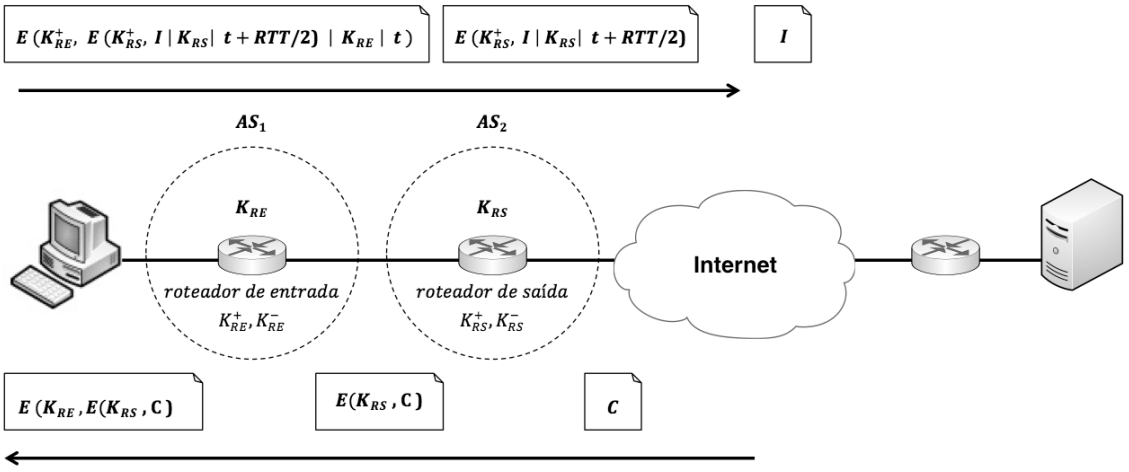


**Figura 3.10. Esquema de funcionamento do Andana.**

Uma vez construído o circuito, o usuário deve conhecer as chaves públicas dos dois roteadores que o compõem. Sejam $K^{+}_{RE}$ a chave pública do roteador *RE* e $K^{+}_{RS}$ a chave pública do roteador *RS*. Dessa forma, um interesse é primeiramente criptografado com $K^{+}_{RE}$ e em seguida novamente com $K^{+}_{RS}$. Após este processo, o interesse terá duas camadas de criptografia e um adversário local não será capaz de saber o que está sendo requisitado. Cada roteador no circuito, além das funções tradicionais correspondentes aos roteadores de conteúdo, deve decriptar o interesse utilizando sua chave privada. Dessa forma, a cada salto no circuito, uma camada de criptografia é removida até que o interesse seja encaminhado pelo roteador *RS* em texto claro, da maneira como a CCN espera.

Para garantir a privacidade dos usuários, além dos pacotes de interesse, os pacotes de dados também precisam trafegar de maneira criptografada na rede local. Dessa forma, cada roteador deverá fazer o processo inverso, ou seja, introduzir uma camada de criptografia no pacote de dados. Para tanto, é necessário que o usuário estabeleça um segredo compartilhado com cada roteador no circuito. Estes segredos são as chaves criptográficas $K_{RE}$ e $K_{RS}$, geradas aleatoriamente pelo usuário. Para realizar a distribuição desses segredos para os roteadores, o usuário adiciona, em cada camada de criptografia do pacote de interesse, a chave correspondente ao roteador que, após remover a camada anterior, obterá o segredo.

Para exemplificar o processo, suponha que $I$ seja um pacote de interesse para um certo conteúdo $C$. Suponha também que $E^1$ seja o algoritmo criptográfico utilizado para criptografar/decriptar interesses e $E^2$ o algoritmo criptográfico utilizado para criptografar/decriptar pacotes de dados. Para qualquer algoritmo $E$, $E(K,D)$ significa a cifragem do dado $D$, utilizando a chave $K$ e o algoritmo $E$. Assim, o roteador *RE* receberá o interesse $I$ criptografado pelo usuário da seguinte maneira: $E^1(K^{+}_{RE}, E^1(K^{+}_{RS}, (I + K_{RS}) +$

 

$K_{RS}) + K_{RE})$. Em seguida ele utilizará sua chave privada $K_{RE}^{-}$ para decriptar o interesse e obterá duas informações. A primeira é o interesse com uma camada a menos de criptografia, ou seja, $E^1(K_{RS}^{+}, (I + K_{RS})$. Este interesse é encaminhado para o roteador *RS*. A segunda informação é o segredo $K_{RE}$, que será utilizado posteriormente para criptografar o pacote de dados. O roteador *RS*, por sua vez, fará o mesmo processo, ou seja, decriptar o interesse recebido com sua chave privada $K_{RS}^{-}$ . Mais uma vez, duas informações são obtidas: o interesse em texto claro $I$ e o segredo $K_{RS}$. Uma vez que o pacote de dados $C$ seja recebido pelo roteador *RS*, ele será criptografado utilizando-se o segredo $K_{RS}$ e o resultado, ou seja, $E^2(K_{RS}, C)$, será encaminhado para o roteador *RE*. Este roteador então criptografará os dados recebidos usando o segredo $K_{RE}$, o que resultará em $E^2(K_{RE}, E^2(K_{RS}, C))$. Logo após, *RE* encaminhará os dados para o usuário. Como este possui $K_{RE}$, $K_{RS}$ e sabe a ordem em que as camadas de criptografia foram adicionadas, ele pode decriptar os dados recebidos, camada a camada, até obter $C$.

Uma possibilidade de ataque a esse sistema seria o adversário capturar pacotes de interesse criptografados e utilizar o próprio circuito para decriptá-los. Entretanto, como já explicado na Seção 3.2, a CCN agrega pacotes de interesse para um mesmo conteúdo e utiliza o *cache* universal para responder a interesses mais rapidamente. Assim, se o adversário enviar um interesse já enviado anteriormente, uma das três seguintes situações pode ocorrer:

1. Já existe uma entrada na PIT para o conteúdo: isso significa que o interesse será agregado na PIT e não será encaminhado para fora do circuito.

2. O conteúdo já foi recuperado e está armazenado em cache: nesse caso o novo interesse será atendido pelo cache e não será encaminhado para fora do circuito.

3. O conteúdo já foi recuperado, mas não está armazenado em cache (o conteúdo é impopular e seu tempo de expiração estourou): nesse caso, como não existe entrada na PIT para o conteúdo e este não está armazenado em cache, o interesse será encaminhado para fora do circuito.

Claramente, as características da CCN reduzem o espaço deste tipo de ataque para apenas o caso 3. Para eliminar essa ameaça, o Andana adiciona *timestamps* em cada camada de criptografia. As camadas mais internas possuem *timestamps* maiores que as mais internas, compensando assim os atrasos de transmissão e processamento em cada roteador. Uma vez que um interesse criptografado é recebido por um roteador do circuito, este verifica se o *timestamp* do interesse está dentro de uma janela de tempo. Se estiver, o interesse é então encaminhado para o próximo salto e caso contrário, o interesse é descartado.

O Andana fornece um bom grau de anonimato com relação a adversários locais. Entretanto, adversários globais, como discutido na Seção 3.4.1, ainda podem comprometer a segurança. Além disso, uma das críticas que se faz a Internet atual é que, por não ter sido projetada tendo a segurança como um de seus objetivos, a implementação de mecanismos que garantam a segurança é mais difícil e mais propenso a falhas. Nesse sentido a CCN representa uma nova arquitetura para a Internet e como tal tem a segurança como um de seus pilares. Entretanto, a adoção do Andana à CCN soa como um ”remendo”,

 

assim como feito na Internet atual. A necessidade de remendar uma arquitetura que ainda nem sequer foi implementada em larga escala parece ser um problema a ser considerado.

### 3.4.3. *Cache Snooping*

Na Internet atual, diversos serviços fazem uso de *caches* para aumentar sua eficiência. Para tanto, conteúdos requisitados com maior frequência são guardados em dispositivos intermediários, de forma que o acesso a esses conteúdos ocorra mais rapidamente. Navegadores web, por exemplo, armazenam páginas recentemente requisitadas localmente. Dessa forma, futuras requisições para essas páginas são prontamente atendidas, sem a necessidade de serem recuperadas do servidor web original. Apesar de aumentar a eficiência na recuperação de conteúdos, a adoção de *caches* na rede também pode introduzir problemas de violação de privacidade. Um usuário malicioso que consiga obter acesso aos dados presentes no *cache* poderá saber que conteúdos os usuários associados a este *cache* estão requisitando. Além disso, uma vez que os pacotes IPs contém a informação sobre o endereço de origem dos dados, um adversário poderia saber "quem", e "o que"está sendo pedido. A técnica de tentar extrair rastros de comunicação dos *caches* para inferir informações sobre seus usuários é conhecida como *cache snooping*.

A maioria dos ataques de *cache snooping* são baseados no tempo de resposta da requisição para um determinado conteúdo, como tratado em [Felten e Schneider 2000]. Assim, suponha que o objetivo de um adversário seja saber se o usuário *A* acessou o site de um usuário *B*. Caso o usuário *A* tenha acessado a página do usuário *B*, o adversário saberá que o usuário *A* provavelmente acessou um documento principal `index.html`. Dessa forma, o adversário prepara um *script* que realiza uma requisição ao `index.html` da página web do usuário *B* e calcula o tempo que o conteúdo leva para ser obtido. Este *script* é então inserido na página web do adversário. O usuário *A* é então levado a acessar o *script* presente na página do adversário (através de um ataque de *phishing*, por exemplo). Nesse momento, o *script* é executado e informa ao adversário sobre o tempo que o usuário *A* leva para receber o arquivo `index.html` do usuário *B*. Se este arquivo já estiver no *cache* do navegador do usuário *A*, este tempo deverá ser pequeno, indicando que o usuário *A* acessou a página web do usuário *B*.

A CCN expande a utilização de *caches* para todos os roteadores e os utiliza como um de seus pilares para garantir a recuperação eficiente e aumento da disponibilidade de conteúdos. Dessa forma, é natural imaginar que se o *cache snooping* é um problema para a Internet atual, ele será ainda maior no contexto da CCN. Diferentemente das redes IP, a CCN não registra informações sobre a origem das requisições. Isto implica em uma maior garantia à privacidade dos usuários, já que mesmo que um adversário explore o *cache* para obter informações sobre os conteúdos nele presentes, nenhuma informação sobre quem os pediu poderá ser extraída. Entretanto, a hierarquia de *caches* produzida pela CCN faz com que um número reduzido de usuários esteja associado a um determinado *cache*, aumentando assim a exposição dos usuários. Nesta seção serão discutidos dois ataques de *cache snnoping* para a CCN, conforme apresentado em [Schwetzingen 2010] :

1. Obtenção de uma cópia dos conteúdos presentes em *cache*. Neste ataque o objetivo é saber que conteúdos os usuários associados ao *cache* estão requisitando. Esse ataque é útil quando o adversário quer verificar se algum dos usuários associados

 

ao *cache* acessou algum conteúdo classificado como interessante pelo adversário.

2. Análise dos acessos a um conteúdo com um nome específico. Este ataque pode ser executado quando um adversário possui um ou mais nomes de conteúdo que ele deseja saber se foram acessados por algum dos usuários associados ao *cache*.

Para a realização desses dois ataques, as seguintes suposições são feitas:

- Os diversos *caches* da rede são organizados hierarquicamente através de uma estrutura em árvore. Os usuários são conectados diretamente a um *cache* folha. Quando ocorre um *cache miss*, as requisições são roteadas para cima na hierarquia, no sentido do publicador. Não existe colaboração entre *caches* vizinhos. A Figura 3.11 ilustra essa topologia.
- Todos os *caches* possuem um tamanho limitado e utilizam o *Least Recently Used* - LRU como política de substituição de entradas.
- Todos os *caches* são construídos utilizando a mesma tecnologia. Isso significa que o tempo de acesso a memória (ou disco) é o mesmo para todos os *caches* da rede.
- Não ausência da possibilidade de uma identificação precisa dos usuários, os adversários sempre estarão interessados no menor grupo possível de usuários onde um ou mais deles requisitou um determinado conteúdo sensível. Observando a Figura 3.11 é possível perceber que ambos os *caches* B e C possuem quatro usuários associados. Subindo na hierarquia, o *cache* A possui oito usuários associados. Isso significa que os adversários sempre estarão interessados em obter informações sensíveis sobre seus vizinhos (usuários na mesma rede) utilizando o *cache* folha diretamente ligado a ele. Essa característica pode ser observada na Figura 3.11, onde o adversário está associado ao *cache* B e deseja obter informações sensíveis a respeito dos usuários A, B e C. Doravante, a palavra *cache* deve ser entendida como o *cache* folha diretamente ligado ao adversário.

Mesmo que o cabeçalho dos pacotes de interesse ou de dados não revelem a origem da requisição dos conteúdos, em algumas situações é possível para o adversário inferir o usuário que requisitou certo conteúdo. Essa informação pode ser obtida através da análise do conteúdo ou do seu nome. Suponha que um publicador gere conteúdos dinâmicos para os seus usuários e que esses conteúdos tenham nomes com a seguinte estrutura: `publicador/conteúdo/nomeUsuário`. Dessa forma, se o usuário João deseja receber sua versão personalizada da página `index.html` do publicador UFF, ele a requisita através do nome `/br/uff/index.html/João`. Se o adversário for capaz de constatar a presença do conteúdo `/br/uff/index.html/João` em cache e conhecer a estrutura de nomeação de conteúdos, ele será capaz de inferir que o usuário João requisitou uma cópia personalizada na página `index.html` do publicador UFF.

Na ausência de conteúdos que forneçam informações mais precisas a respeito da identidade dos requisitantes, o adversário não será capaz de identificar exatamente o requisitante de um determinado conteúdo. Nesse caso, a única alternativa é associar os

 

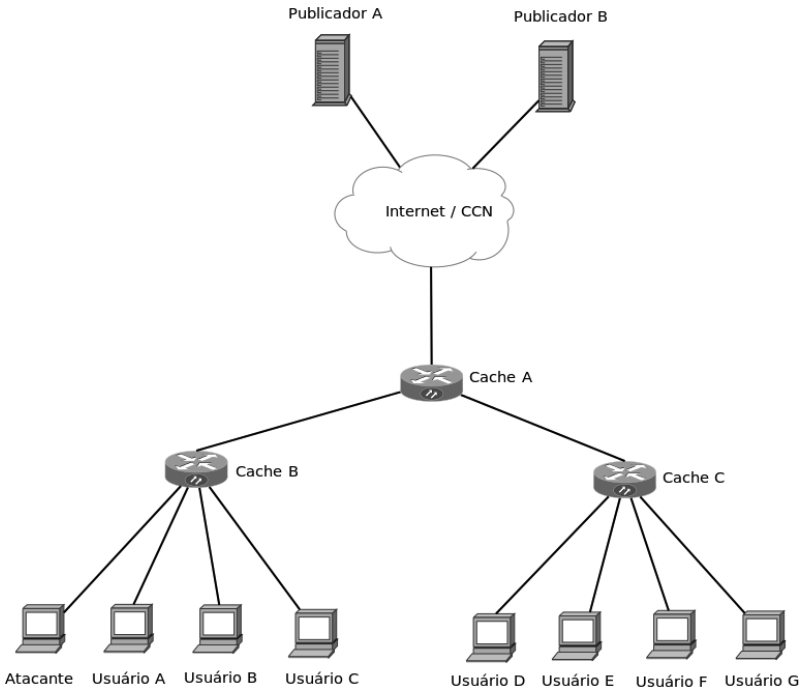


**Figura 3.11. Topologia simplificada para um ataque de *cache snooping*.**

conteúdos a um grupo de possíveis requisitantes, ou seja, ao grupo contendo todos os usuários associados ao mesmo *cache* que o adversário.

Além do alvo do ataque, o adversário também precisa determinar, a priori, quais os conteúdos de interesse para o seu objetivo. A ideia é que o adversário gere uma lista de nomes de conteúdo interessantes ou então determine características que, se presentes em um conteúdo, o tornariam interessantes para o ataque.

#### 3.4.3.1. Requisições Recursivas e Não-Recursivas

Para saber se determinado conteúdo foi acessado por um ou mais usuários, o adversário precisa requisitá-lo à rede e determinar se a resposta obtida é oriunda do *cache* associado a estes usuários. Existem basicamente duas maneiras de se enviar provas ao *cache*: requisições recursiva e não-recursiva.

Na requisição não recursiva (Figura 3.12 (a)), se um *cache miss* ocorrer, o adversário não receberá dados e saberá que tal conteúdo não foi requisitado por nenhum dos usuários associados ao *cache*. Por outro lado, se os dados forem recebidos, o adversário saberá que um ou mais vizinhos o requisitou.

Na requisição recursiva, ilustrada na Figura 3.12 (b), um *cache miss* faz com que o roteador encaminhe o interesse para o próximo *cache*, na direção do publicador do conteúdo. Dessa forma, se o conteúdo existir, o adversário receberá o pacote de dados após algum intervalo de tempo. Como o adversário sempre recebe os dados, ele não pode utilizar este evento como uma indicação de presença ou ausência do conteúdo em *cache*. Para contornar este problema, o adversário deve tentar inferir o tempo de resposta

 

para um conteúdo obtido diretamente do *cache*. Se o tempo de resposta de obtenção de um determinado conteúdo for próximo ao tempo de resposta inferido, então o adversário poderá concluir que o conteúdo foi obtido do *cache*.

Entretanto, na requisição recursiva, o envio de provas ao *cache* altera o seu estado e essa característica deve ser levada em conta ao definir a taxa de envio de provas. Por exemplo, se um *cache miss* ocorrer para a primeira prova enviada, os dados serão recuperados através de um outro *cache* na rede ou da fonte, o que causará o armazenamento do conteúdo por todos os *caches* no caminho. Se a próxima prova for enviada antes da expiração da entrada do *cache* inserida pela primeira prova, o adversário observará um falso positivo.

Em geral, o ataque baseado em provas não recursivas é mais simples de implementar[6] e mais efetivo, já que sempre fornece uma resposta exata. Apesar disso, os ataques baseados em provas recursivas são mais difíceis de detectar, já que o mesmo corresponde ao comportamento normal da CCN [Schwetzingen 2010], mas fornecem respostas menos precisas devido a variação dos tempos de resposta inferidos. Devido a maior dificuldade de implementação e elaboração de contramedidas, os ataques estudados nessa seção são baseados em provas recursivas.

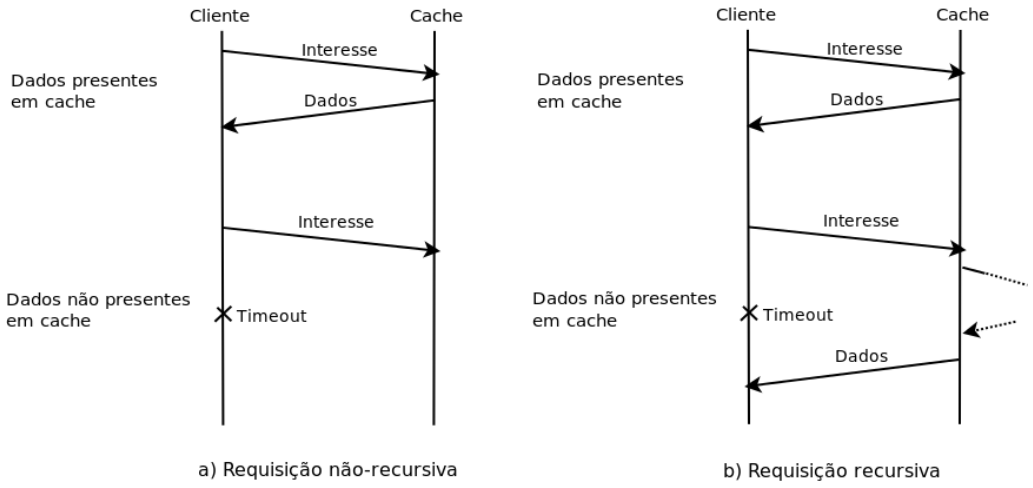


**Figura 3.12. Tipos de requisições de conteúdos ao *cache*.**

### 3.4.3.2. Estimando o Tempo Característico e o RTT para o *Cache*

Antes da realização do ataque propriamente dito, o adversário precisa coletar algumas informações de temporização para compreender melhor a topologia da rede. A primeira informação necessária é o tempo que um conteúdo leva para ser obtido através do *cache*. Para estimar esse tempo, o adversário precisa requisitar um conteúdo que esteja em *cache* com absoluta certeza. Se o adversário não souber de nenhum conteúdo armazenado em *cache*, ele pode requisitar por duas vezes consecutivas um mesmo conteúdo

[6]Para implementar requisições não-recursivas basta atribuir o valor 2 ao campo escopo (explicado na Seção 3.2.2) presente nos pacotes de interesse.

 

à rede e utilizar a resposta da segunda solicitação para obter a estimativa do *Round-Trip Time* - RTT. O adversário então requisita o conteúdo em *cache* e calcula a diferença entre os tempos de envio do interesse e da obtenção dos dados. Esse processo deve ser repetido várias vezes pelo adversário, com o objetivo de se obter uma medição mais precisa. O tempo entre as duas requisições deve ser pequeno o suficiente para garantir que o conteúdo não seja removido do *cache* e grande o bastante para que as provas possam representar o RTT em diferentes cenários de tráfego.

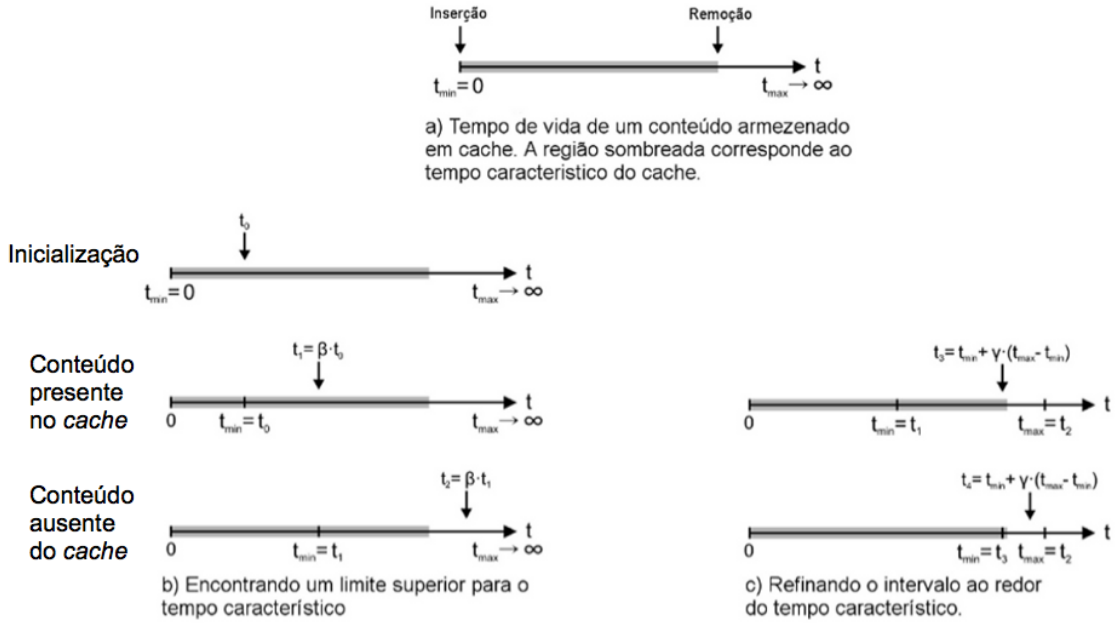


**Figura 3.13. Algoritmo para estimativa do tempo característico do *cache*.**

O segundo parâmetro necessário é o tempo de expiração de um conteúdo armazenado em *cache*, chamado de tempo característico ($t_c$) de um *cache* LRU [Che et al. 2002], ilustrado na Figura 3.13(a). Para tanto, [Schwetzingen 2010] estabelece o seguinte algoritmo:

**Inicialização:** Faça $t_{min} = 0$, $t_{max} \to \infty$ e $i = 0$. Use $t_0 > 0$, $\beta > 1$, $0 < \gamma < 1$ e $n \in N^+$ como parâmetros fixos, mas com valores passíveis de aperfeiçoamento.

**Encontrando um limite superior:** Insira um novo conteúdo no *cache* e o requisite após o instante de tempo $t_i$. Assim, dois casos podem ocorrer. No primeiro, como o conteúdo já não está em *cache*, faça $t_{max} = t_i$, $i = i+1$ e vá para o próximo passo do algoritmo. No segundo caso, se o conteúdo ainda estiver em *cache*, faça $t_{min} = t_i$, $t_{i+1} = \beta.t_i$, $i = i+1$ e repita este passo. Este processo é ilustrado na Figura 3.13(b).

**Refinando a estimativa:** Insira um novo conteúdo no *cache* e o requisite após o tempo $t_i = t_{min} + \gamma.(t_{max} - t_{min})$. Assim, dois casos podem ocorrer. No primeiro, como o conteúdo não está em *cache*, faça $t_{max} = t_i$. No segundo caso, se o conteúdo ainda estiver em *cache* faça $t_{min} = t_i$. Incremente $i$ e repita este passo até que um número pré-determinado $n$ de iterações tenha sido alcançado. A Figura 3.13(c) ilustra este processo para $n = 2$.

 

É importante observar que, cada prova enviada para um conteúdo armazenado em *cache* faz com que o tempo de expiração desse conteúdo seja reiniciado. Isso significa que, após cada prova, o conteúdo em questão sempre levará $t_c$ unidades de tempo para ser removido do *cache*. Dessa forma, o algoritmo apresentado leva em conta essa característica e tenta, através de tentativa e erro, encontrar um intervalo $[t_{max} - t_{min}]$ que corresponda ao $t_c$. Inicialmente, o algoritmo precisa definir um valor para a variável $t_0$. Se o adversário possuir uma estimativa de $t_c$, mesmo que grosseira, esta deve ser atribuída a $t_0$. Caso o valor escolhido para $t_0$ seja menor que $t_c$, então o algoritmo continuará a procurar um limite superior. Nesse ponto, o parâmetro $\beta$ indica o quão rápido a estimativa do limite superior aumenta. Quanto maior o valor de $\beta$, mais rápido o limite superior é encontrado, apesar de ser mais impreciso. Após o limite superior ter sido obtido, o intervalo $[t_{max} - t_{min}]$ precisa ser reduzido até representar, com a maior precisão possível, o valor de $t_c$. Dessa forma, diversos pontos dentro do intervalo são testados e cabe ao parâmetro $\gamma$ indicar o próximo ponto a ser testado. O critério de parada $n$ define o quão precisa a estimativa de $t_c$ será. Seu valor pode ser definido experimentalmente ou a fase de refinamento pode ser executada indefinidamente até que uma alteração mínima no intervalo seja observada.

Neste ponto, com os valores estimados do RTT para requisições feitas ao *cache* e do $t_c$, o adversário já tem condições de saber se um conteúdo está ou não em *cache*, e em caso positivo, quanto tempo ele permanecerá lá. A partir dessas informações, é possível realizar os dois ataques enunciados nesta seção e explicados a seguir.

#### 3.4.3.3. Ataque 1: Obtenção de uma cópia de todos os conteúdos presentes em *cache*

A CCN realiza a comparação de nomes de conteúdo pelo maior prefixo para a verificação da existência de um conteúdo em *cache*. Isso significa que conteúdos podem ser requisitados especificando-se apenas prefixos de seus nomes. Como vários conteúdos em *cache* podem compartilhar um mesmo prefixo, esta comparação pode resultar em mais de uma resposta possível. Neste caso, para decidir qual conteúdo deve ser retornado, a CCN disponibiliza alguns campos no cabeçalho dos pacotes de interesse. Um desses campos representa uma lista de componentes que não devem estar presentes no nome do conteúdo retornado. Assim, é possível requisitar um conteúdo por um determinado prefixo e excluir possíveis ambiguidades através do campo filtro de exclusão.

[Schwetzingen 2010] apresenta o seguinte algoritmo para listar os conteúdos presentes em *cache*:

- Seja $p$ uma entrada para o algoritmo, onde $p = /p_0/p_1/.../p_i/$ é o prefixo no espaço de nomes que o adversário deseja explorar. Faça $E = \emptyset$, onde $E$ representa o filtro de exclusão.

- Envie para o *cache* um pacote de interesse contendo $p$ e $E$. Se houver um *cache miss* o algoritmo termina. Considere $n = /n_0/n_1/.../n_j$ o nome do conteúdo recebido, onde $j \geq i$, $p_0 = n_0$ e $p_1 = n_1$, ..., $p_i = n_i$. Se $j > i$, então $\forall e \in E : n_{i+1} \neq e$. Em outras palavras, se o nome do conteúdo retornado for maior do que o prefixo $p$ requisitado, então nenhum de seus componentes adicionais poderá ser igual a

 

qualquer componente na lista $E$. Por fim, faz-se $E = E \cup \{n_{i+1}\}$ e repete-se este passo.

A partir do algoritmo descrito acima, apenas os conteúdos com prefixo $p$ serão descobertos. Para descobrir todos os conteúdos presentes em *cache* o adversário poderia começar com $p = /$ e executar o algoritmo até que nenhum conteúdo seja retornado.

#### 3.4.3.4. Ataque 2: Análise dos acessos a um conteúdo de nome específico

O objetivo deste ataque é monitorar os acessos a um conteúdo específico, cujo nome é conhecido pelo adversário. Para este propósito, o algoritmo de precisão arbitrária proposto em [Schwetzingen 2010] é capaz de estimar o tempo de inserção e remoção de um conteúdo em *cache*. Executando-se tal algoritmo periodicamente é possível monitorar os acessos ao conteúdo e, inclusive, calcular a taxa de acesso ao mesmo.

##### Descoberta dos Tempos de Inserção e Remoção de um Conteúdo em *Cache*

Através dos processos já explicados nesta seção, é possível verificar se um conteúdo está ou não em *cache*. Suponha que o conteúdo é requisitado no instante de tempo $t_r$. Se o conteúdo estiver em *cache*, então será possível concluir que o mesmo foi requisitado ao menos uma vez nas últimas $t_c$ unidades de tempo. Caso contrário, não é possível concluir se o mesmo nunca foi requisitado anteriormente ou se no momento da requisição de prova o conteúdo já havia expirado. Em qualquer caso, não é possível inferir em quais momentos o conteúdo foi inserido e removido do *cache*.

Para contornar esse problema, uma solução seria enviar provas periódicas ao *cache* para testar os momentos em que o conteúdo estava ou não estava em *cache*, permitindo assim estabelecer uma estimativa sobre os tempos de inserção e remoção. Entretanto, enviar provas sucessivas ao *cache* para um mesmo conteúdo limita a taxa de requisição a pelos menos uma a cada $t_c$ unidades de tempo. Isto ocorre, pois a cada requisição um *cache miss* indica que o conteúdo será obtido da fonte e inserido em *cache*, o que será desfeito após $t_c$ unidades de tempo. Quanto maior a capacidade do *cache*, maior será o $t_c$ e para *caches* baseados em disco, o $t_c$ pode ser de horas, ou mesmo dias. Nesses casos, as provas periódicas se tornariam impraticáveis. Apesar disso, outra característica da CCN pode ser aproveitada para viabilizar este processo. Quando um conteúdo é muito grande, a CCN o divide em pedaços menores (*chunks*) e os nomeia individualmente. Ocorre que a requisição de um desses pedaços é normalmente precedida da requisição dos pedaços subsequentes. Dessa maneira, para verificar se o conteúdo maior foi requisitado, é possível verificar a presença em *cache* de cada um de seus pedaços. Como cada um desses pedaços representa uma entrada separada no *cache*, o adversário pode se valer dessa característica para efetuar as provas periódicas, enviando várias requisições, uma para cada um dos diferentes pedaços do mesmo conteúdo, aumentando assim a taxa de requisição. Por simplicidade, o ataque aqui descrito assume que todos os *chunks* do mesmo conteúdo são requisitados exatamente ao mesmo tempo.

Considere $t_c$ o tempo característico do *cache*, $\varepsilon$ a incerteza do $t_c$ estimado e $t_p$

 

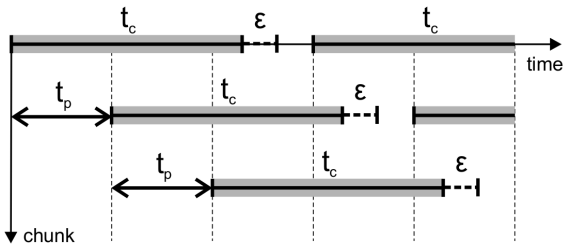


**Figura 3.14. Provas paralelas usando três *chunks*[Schwetzingen 2010].**

o intervalo entre as provas, determinado pelo adversário. . Assim, o número de *chunks* necessários para que o ataque possa ser executado é $\lceil \frac{t_c+\varepsilon}{t_p} \rceil$. Para *caches* com $t_c$ pequeno e um grande número de *chunks*, é possível definir $t_p$ como um valor pequeno para se obter uma estimativa mais precisa do tempo de inserção e remoção de um conteúdo em *cache*. A Figura 3.14 ilustra a execução do ataque para três *chunks*.

### Inferindo a Taxa de Requisição de um Conteúdo em *Cache*

Após encontrar os instantes de tempos de inserção e remoção do conteúdo em *cache*, como discutido anteriormente, o adversário pode estimar a taxa de requisição desse conteúdo. Para que este ataque seja possível, os instantes de tempo entre a remoção do conteúdo e sua posterior reinserção deve ser consideravelmente mais longo do que a precisão do algoritmo de requisição. Isso significa que a frequência de acesso aos itens utilizados nas provas não deve ser muito alta.

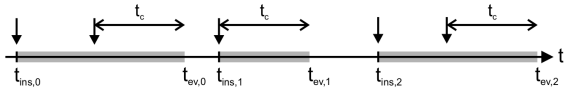


**Figura 3.15. Medição da taxa de requisição de um conteúdo em um *cache* LRU[Schwetzingen 2010].**

A Figura 3.15 mostra o tempo de vida de um conteúdo em um *cache* LRU. Tal conteúdo sempre é inserido nos instantes $t_{ins,i}$ e removidos nos instantes $t_{ev,i}$ e estes são os únicos tempos conhecidos pelo adversário. Cada vez que o conteúdo é requisitado, seu tempo de remoção é atrasado por $t_c$ unidades de tempo a partir do tempo de requisição. Entretanto, somente com os valores de $t_{ins,i}$ e $t_{ev,i}$, o adversário não é capaz de determinar quantas requisições ocorreram durante este intervalo. É possível apenas estabelecer o limite inferior $\lceil \frac{t_{ev,i}-t_{ins,i}}{t_c} \rceil$, mas não o limite superior. Apesar disso, um adversário é capaz de inferir o intervalo de tempo entre duas requisições sucessivas se o conteúdo foi removido do *cache* entre tais requisições. Este intervalo de tempo é calculado da seguinte forma: $t_{gap,i} = t_{ins,i+1} - t_{ev,i} + t_c$. Se existir um número grande desses espaçamentos entre duas requisições, então o adversário será capaz de computar a taxa de requisição global da seguinte maneira: $\lambda \approx \frac{1}{avg(t_{gap,i})}$.

 

#### 3.4.3.5. Contramedidas

Segundo [Schwetzingen 2010], as contramedidas para os ataques de *cache snooping* podem ser divididas em duas categorias: contramedidas de prevenção e contramedidas de detecção.

**Contramedidas de Prevenção**

Para impedir que adversários sejam capazes de efetuar o Ataque 1 as seguintes medidas podem ser tomadas:

- A CCN possui um mecanismo próprio que permite a enumeração de conteúdos em *cache*. Por medidas de segurança, esse mecanismo deve ser desabilitado.
- A funcionalidade de exclusão de componentes dos nomes de conteúdo deve ser alterada de tal forma que os pacotes de dados contenham informações que digam quais partes do nome do conteúdo (nenhuma, por padrão) podem ser excluídas. Dessa forma, o publicador passa a ter mais controle sobre como seus conteúdos são pesquisados. Uma alternativa mais radical seria desabilitar totalmente a funcionalidade de exclusão.
- O mecanismo de comparação de prefixos deve limitar o número de componentes não coincidentes. Por exemplo, se esse limite for 2, então um interesse para `/br` recebido por um cache que possui o conteúdo `/br/uff/videos/video1` não retornará dados, já que a comparação de prefixos resultará em três divergências.

Para impedir que adversários sejam capazes de efetuar o Ataque 2 as seguintes medidas podem ser tomadas:

- Requisições não recursivas devem ser ignoradas por roteadores. Isso implica que se um interesse tiver o campo escopo definido com o valor 2, o interesse deve ser descartado pelo roteador.
- Para dificultar o adversário, um roteador de conteúdo poderia inserir um atraso na resposta que seria equivalente a um RTT para um *cache* na hierarquia que tivesse um grande número de usuários associados. Fazendo isso, mesmo que adversário seja capaz de detectar a taxa de acesso ao *cache*, o número de possíveis requisitantes será tão grande que essa informação será muito pouco útil. Claramente essa contramedida só funcionaria se não houvesse outras maneiras do adversário identificar o requisitante, como por exemplo, a presença do nome de usuário do requisitante dentro do conteúdo requisitado.

**Contramedidas de Detecção**

No caso da prevenção não ser possível, os ataques devem ser ao menos detectados. Todos os ataques abordados nessa seção possuem as seguintes características em comum:

 

- O roteador de acesso poderá observar uma aglomeração de requisições, oriundas do mesmo enlace, quando o adversário está realizando o Ataque 2.
- Da mesma forma, o Ataque 1 envolve a requisição de um grande número de conteúdos em um tempo muito pequeno, com uma grande taxa de *hits* no *cache* e com uma grande cobertura dos conteúdos em *cache* para o espaço de nomes especificado.

Como ocorre na maioria das detecções de ataque, a utilização de verificação de padrões utiliza heurísticas que, em alguns casos, podem gerar falsos positivos. Por exemplo, a requisição de um conteúdo muito grande (com muitos *chunks*) em *cache* pode ser confundida com o Ataque 1.

### 3.4.4. Ataques de Negação de Serviço

Como abordado na Seção 3.3, a CCN mitiga os ataques de negação de serviço comumente implementados na Internet atual. Apesar disso, é possível criar variações efetivas de tais ataques. Como os pacotes de dados são enviados somente após a recepção de seu respectivo pacote de interesse, a única técnica de inundação disponível para o adversário é a de pacotes de interesse. Além disso, os *caches* dos roteadores também podem ser explorados por ataques de envenenamento de *cache* com o objetivo de disseminar conteúdos corrompidos, impedindo assim que os usuários obtenham conteúdos válidos. Estes dois ataques, inundação de interesses e envenenamento de *cache*, serão discutidos neste seção, além de algumas contramedidas propostas para os mesmos.

#### 3.4.4.1. Inundação de Pacotes de Interesse

Em tais ataques os adversários geram um grande número de pacotes de interesse para conteúdos específicos, esperando causar o mau funcionamento da infraestrutura da rede ou da fonte de conteúdos. No caso do alvo do ataque ser a infraestrutura da rede, o objetivo do adversário é fazer com que os interesses gerados por ele consumam toda a memória disponível na PIT dos roteadores. Dessa forma, assumindo que a CCN implemente a política de substituição *Tail Drop* [Wählisch et al. 2012], os interesses enviados por usuários legítimos encontrarão a PIT com recursos exauridos e serão descartados. Da mesma forma, o objetivo do ataque às fontes de conteúdo também é causar a recusa de interesses legítimos. Entretanto, tais interesses são descartados pelas fontes de conteúdo, e não pelos roteadores da rede. Em ambos os casos, o adversário pode utilizar *botnets* para amplificar a eficiência do ataque.

Os pacotes trafegados na CCN não transportam informações de origem e destino em seus cabeçalhos. Assim, da mesma forma que a rede é orientada ao conteúdo, os ataques também precisam ser. Ao invés de escolher o endereço de destino do alvo, o adversário precisa decidir quais conteúdos requisitar para alcançar seu objetivo. Segundo [Gasti et al. 2012], os ataques de negação de serviço por inundação de interesses podem ser divididos em três categorias, dependendo do tipo de conteúdo requisitado no ataque:

1. Estático ou existente: são conteúdos que foram publicados e estão disponíveis na rede para serem requisitados pelos usuários.

 

2. Gerado dinamicamente: são conteúdos gerados apenas quando requisitados através de pacotes de interesse.

3. Não existentes: são conteúdos cujos interesses nunca serão atendidos.

Ataques de inundação de interesses do tipo 1 tem efetividade limitada e são mais frequentemente utilizados para atacar a infraestrutura da rede, e não as fontes de conteúdo. Devido a implementação de *caches* de conteúdo nos roteadores da rede, os interesses para um conteúdo estático somente atingirão a fonte em um primeiro momento. Nas próximas requisições, os interesses serão atendidos pelos *caches*.

Os ataques do tipo 2 são mais adequados quando o objetivo do adversário é causar a inoperabilidade de uma fonte de conteúdos específica. Uma vez que os conteúdos são gerados dinamicamente, estes não são armazenados em *cache*, e interesses para eles são sempre roteados até a fonte. Se a geração de tais conteúdos for computacionalmente cara, tanto em memória quanto em processamento, a fonte de conteúdos pode ficar facilmente inoperante, dependendo do número de interesses de ataque que a atinjam. Por outro lado, o impacto deste ataque nos roteadores depende da sua proximidade em relação a fonte de conteúdos. Os roteadores mais próximos tendem a manter um maior número de interesses pendentes maliciosos em sua PIT, devido a concentração do tráfego a ser encaminhado à fonte de conteúdos.

Os ataques do tipo 3 poderão ser mais eficientes caso o objetivo do adversário seja afetar a infraestrutura da rede. Da mesma forma, os interesses também não podem ser atendidos por *caches* e são propagados até a fonte de conteúdos. Uma vez que os conteúdos requisitados são inexistentes, a fonte de conteúdos pode descartá-los rapidamente, evitando assim o desperdício de recursos. Porém, o processo de roteamento desses interesses até a fonte consome a memória da PIT dos roteadores, o que pode gerar sua inoperabilidade. Outra característica desse tipo de ataque é a maior facilidade de evasão do mecanismo de agregação de interesses. Como os conteúdos requisitados não existem, a lista de nomes de conteúdo disponíveis para requisição é virtualmente infinita. Dessa forma, gerar muitos interesses para conteúdos diferentes é mais fácil neste ataque do que nos outros dois tipos. Para gerar nomes de conteúdo inexistentes [Gasti et al. 2012] propõe as seguintes opções de algoritmos:

1. Definir o nome de conteúdo utilizando o formato `/prefixo/nonce`, onde *nonce* é um valor aleatório. Como a CCN realiza a checagem de prefixo mais longo, os interesses para esses nomes serão roteados até a fonte de conteúdos.

2. Definir o *hash* da chave pública do publicador, contido nos pacotes de interesse como um valor aleatório. Uma vez que nenhum conteúdo legítimo conterá o mesmo valor para o *hash* da chave pública do publicador, o interesse nunca será atendido.

3. Utilizar o filtro de exclusão do pacote de interesse contendo todos os campos presentes no nome do conteúdo requisitado. Dessa forma, ainda que exista em *cache* um conteúdo com o nome requisitado, ele não será devolvido, já que o filtro de exclusão o excluirá das possíveis respostas.

 

A falta de informações de origem e a ausência de assinatura nos pacotes de interesse dificulta o rastreamento da origem dos ataques de inundação de interesses. Assim, qualquer usuário pode gerar um fluxo de pacotes de interesse e manter seu anonimato. Para contornar este problema, seria possível adotar uma solução onde os consumidores fossem obrigados a assinar os pacotes de interesse por ele gerados. Dessa forma, em um ataque de inundação de interesses seria possível descobrir qual usuário os originou e tomar as devidas contramedidas. Porém, esta solução causaria sérios problemas à privacidade dos usuários e não seria completamente efetiva, já que adversários operam *botnets* para realizar os ataques. Para mitigar os ataques de negação de serviço por inundação de interesses sem interferir na privacidade dos usuários, [Gasti et al. 2012] propõem contramedidas baseadas em estatísticas nos roteadores e em mecanismos de *Push-Back* [Ioannidis e Bellovin 2001].

### Contramedidas Baseadas em Estatísticas nos Roteadores

Os roteadores da CCN podem facilmente manter estatísticas de interesses não atendidos (expirados) e utilizar essas informações para limitar:

- O número de interesses pendentes por interface de saída: a CCN mantém um balanceamento de fluxo entre os pacotes de interesse e de dados. Em resposta a um determinado interesse, apenas um pacote de dados pode ser recebido e será encaminhado pelo caminho inverso do interesse que o originou. Dessa forma, é fácil para os roteadores calcularem o número máximo de interesses pendentes por interface de saída que o próximo salto pode suportar. Assim, os roteadores nunca devem enviar um número de interesses maior do que esse limite superior, baseando-se no tamanho médio dos pacotes, tempo de expiração dos interesses e na largura de banda do enlace.

- O número de interesses por interface de entrada: pelo mesmo princípio do balanceamento de fluxo, os roteadores podem detectar quando estão recebendo um número de interesses maior do que o suportado, devido a limitações físicas do enlace.

- O número de interesses pendentes por *namespace*: quando certo prefixo está sendo alvo de um ataque DoS, os roteadores (principalmente aqueles mais próximos à fonte de conteúdos) podem facilmente detectar o ataque baseando-se no número de interesses não atendidos para este prefixo. Após a detecção, os roteadores podem limitar o número de interesses pendentes para o prefixo supostamente em ataque e reduzir o número de interesses pendentes para as interfaces de entrada por onde os interesses pendentes para tal prefixo foram recebidos.

### Contramedidas Baseadas em Mecanismos de *Push-Back*

Após detectar um ataque de inundação de interesses para um determinado *namespace*, o roteador deve limitar a taxa de encaminhamento de interesses para o prefixo em ataque. Além disso, o roteador também deve propagar um alerta de ataque pelas interfaces por

 

onde os interesses para tal prefixo foram recebidos. Cada roteador que receber o alerta deve realizar o mesmo processo [Ioannidis e Bellovin 2001]. O objetivo deste mecanismo é limitar o ataque à sua origem ou ao menos, a região onde o ataque é detectável, como ilustrado na Figura 3.16. Uma característica importante dessa contramedida é que ela pode ser implementada sem alterações na infraestrutura atual da CCN.

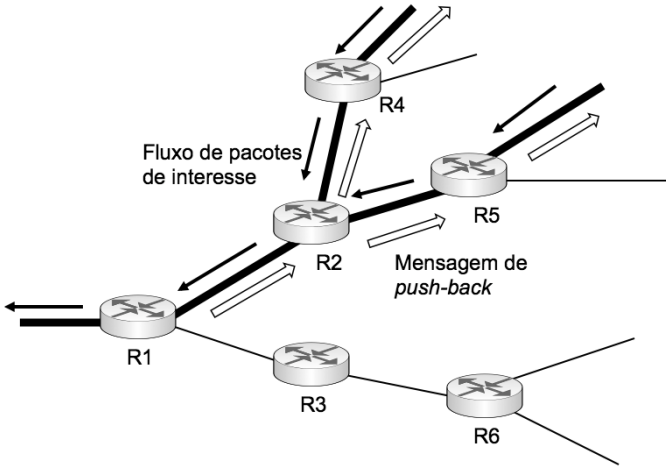


**Figura 3.16. Exemplo de mecanismo de *push-back* com detecção de ataque pelo roteador R1**

### 3.4.4.2. Envenenamento de *Cache*

Como explicado na Seção 3.3, a CCN exige que os publicadores assinem cada pedaço de conteúdo publicado. Dessa maneira, um consumidor pode verificar a assinatura e ter a garantia de que o conteúdo recebido é íntegro e autêntico. Entretanto, para os roteadores, a verificação da assinatura de cada pedaço de conteúdo recebido não é viável por dois motivos:

1. Criptografia Assimétrica: para assinar conteúdos, os publicadores utilizam um algoritmo de criptografia assimétrica, como o RSA, que mais tarde também será utilizado para a verificação da assinatura. Uma vez que tais algoritmos envolvem operações computacionalmente caras, como a exponenciação, a verificação da assinatura de todos os conteúdos pode ocupar muito tempo de processamento dos roteadores, o que pode ocasionar na negação de serviço. Mesmo que um processador dedicado seja usado para a criptografia, o atraso causado pela verificação das assinaturas pode inviabilizar muitas aplicações. Como exemplo, [Gasti et al. 2012] mostram que uma implementação otimizada em software da verificação de assinaturas RSA-1024, rodando em um processador Intel Core 2 Duo 2.53 GHz, permite verificar a assinatura de cerca de 150Mbps de tráfego, assumindo que o tamanho dos pacotes de dados seja de 1500 bytes. Como observação, [Gasti et al. 2012] destacam que foi utilizado o menor expoente público possível para o RSA, resultando em apenas duas multiplicações modulares por verificação de assinatura. Logo, é possível concluir que para interfaces na ordem de Gbps a quantidade de poder computacional necessária para a verificação de todas as assinaturas é um limitador.

 

2. Gerenciamento de Confiança: para verificar uma assinatura é preciso conhecer a chave pública do assinante. Porém, para garantir que uma determinada chave pública pertence realmente a entidade esperada, é preciso utilizar mecanismos de gerenciamento de confiança, como o PGP e o SPKI/SDSI. Em geral, esses mecanismos requerem que seja verificada uma cadeia de confiança para garantir a autenticidade da chave pública. Dessa forma, além da necessidade da verificação da assinatura em si, também é necessária a verificação da autenticidade da chave pública.

Uma vez que a verificação da assinatura em todos os conteúdos seria inviável para os roteadores, a CCN permite que esse processo seja opcional e torna obrigatória a verificação das assinaturas apenas pelos consumidores. Assim, é possível para usuários maliciosos gerarem conteúdos corrompidos (assinatura inválida) e falsos (assinatura válida, mas com a chave privada incorreta) e propagarem os mesmos na rede. Esses conteúdos serão eventualmente recuperados por usuários e serão armazenados em *caches*, aumentando assim a disponibilidade de conteúdo ilegítimo na rede. Quando um conteúdo falso ou corrompido é recebido por um consumidor, este identificará esta situação, através da verificação da assinatura, e os descartará. Em seguida, o consumidor deve requisitar o conteúdo novamente, utilizando o filtro de exclusão para eliminar o conteúdo recebido anteriormente como possível resposta.

Apesar do consumidor constantemente identificar os conteúdos falsos ou corrompidos, se o mesmo nunca conseguir recuperar um conteúdo válido, então o serviço de recuperação de conteúdos será negado a ele, já que os conteúdos recebidos serão sempre inúteis. Para explorar esta vulnerabilidade, [Gasti et al. 2012] abordam duas variações de ataques:

1. O adversário controla um ou mais roteadores da rede e conhece os interesses pendentes presentes nos mesmos. Assim, para cada interesse pendente o adversário pode gerar uma resposta corrompida ou falsa, que será armazenada nos *caches* de outros roteadores.

2. O adversário é capaz de prever que muitos interesses para um determinado conteúdo popular, ainda não presente em *cache*, serão enviados à rede. O adversário então faz com que várias máquinas zumbis enviem interesses para estes conteúdos quase simultaneamente. Em seguida, um nó da rede comprometido que receba um ou mais desses interesses responderá aos mesmos com conteúdos falsos ou corrompidos. Quando os usuários legítimos enviarem interesses para estes conteúdos, os conteúdos inválidos em *cache* serão recuperados e não sua versão legítima.

Embora os dois tipos de ataque exijam esforços diferentes para os adversários, ambos mantém o mesmo impacto à infraestrutura da rede. Assim, as contramedidas necessárias se tornam, na maioria das vezes, eficazes para ambos os ataques. Algumas contramedidas são propostas em [Gasti et al. 2012], como a implementação de funcionalidades dos nomes auto-certificados aos nomes de conteúdo da CCN, a verificação probabilística de assinaturas e a utilização de alertas de conteúdo corrompido.

 

### Implementação de Funcionalidades dos Nomes Auto-Certificados aos Nomes de Conteúdo da CCN

Como abordado na Seção 3.3.2, os nomes auto-certificados estabelecem uma relação criptográfica entre um pacote de interesse e seu respectivo pacote de dados. Os conteúdos são requisitados pelos nomes, que por sua vez são construídos a partir do *hash* da chave pública do publicador ou do *hash* do conteúdo em si. Assim, quando um consumidor requisita um conteúdo, ele tem a garantia que a resposta obtida é aquela esperada. Uma vez que os nomes auto-certificados não são inteligíveis por seres humanos, é preciso utilizar um mecanismo que traduza nomes legíveis para seus correspondentes nomes auto-certificados. Por outro lado, o mecanismo de nomeação usado pela CCN é inteligível por seres humanos, porém precisa de um mecanismo para gerenciamento de confiança. Essa necessidade inviabiliza a verificação da assinatura de todos os conteúdos pelos roteadores, levando aos ataques DoS tratados nesta seção.

Assim, [Gasti et al. 2012] propõem o mecanismo de nomeação *Self-Certifying Interests/Data Packets* (SCID), uma união de algumas funcionalidades dos nomes auto certificados à CCN, sem alterar sua estrutura de nomes. O objetivo do SCID é permitir que roteadores decidam se um determinado conteúdo é a resposta correta para um interesse recebido. O SCID possui duas variantes: uma para conteúdos estáticos (S-SIDC) e outra para conteúdos dinâmicos (D-SIDC).

Quando um conteúdo é publicado, o publicador deve criar um nome para ele, o que é feito de acordo com suas políticas de nomeação. Além dos componentes selecionados pelo publicador, a CCN também adiciona ao final do nome o *hash* do pacote de dados. Os consumidores podem então decidir utilizar o nome do conteúdo com ou sem o *hash* nas requisições. Dessa maneira, a ideia do S-SIDC é fazer com que os roteadores comparem o *hash* presente no nome do conteúdo do pacote de interesse com aquele presente no pacote de dados. Assim, mesmo que o adversário possa criar conteúdos falsos ou corrompidos, estes não serão enviados aos consumidores, já que o *hash* dos mesmos não corresponderá ao do pacote de dados legítimo.

O mecanismo S-SIDC funciona bem para obter conteúdos completos. Para recuperar conteúdos grandes, divididos em pequenos *chunks*, seria necessário que o consumidor conhecesse o *hash* de cada um dos *chunks* que compõem o conteúdo, o que tornaria o S-SIDC impraticável. Uma solução para este problema é fazer com que cada *chunk* contenha, como metadado, o *hash* do próximo *chunk* na sequência, reduzindo assim o problema a descobrir apenas o *hash* do primeiro *chunk*. Essa técnica, entretanto, faria com que o processo de requisição de conteúdos se tornasse sequencial, o que limitaria a vazão da rede. Para superar esse problema, [Gasti et al. 2012] propõem que os *chunks* armazenem como metadados não só o *hash* do *chunk* seguinte, mas também os próximos *n chunks*. Dessa maneira, de posse do primeiro *chunk*, o consumidor seria capaz de requisitar mais *n chunks* em paralelo. Um problema com essa técnica é o aumento do tamanho dos pacotes de interesse. Se o algoritmo de *hash* utilizado for o SHA-256, por exemplo, cada pacote terá $n * 32$ bytes de *overhead*. Em particular, se o valor de $n$ for próximo de 46, o *overhead* será próximo do tamanho máximo do pacote de dados (supondo pacotes de dados com no máximo 1500 bytes). Assim, quanto maior for o valor de $n$, maior o grau de paralelismo nas requisições, mas também maior o *overhead* nos pacotes de dados.

 

Uma vez que o S-SIDC requer que os consumidores conheçam o *hash* do conteúdo a ser requisitado, esta técnica não é aplicável a conteúdos dinâmicos, já que não é possível predizer o *hash* de um conteúdo que ainda não foi gerado. Para resolver este problema, uma ideia seria realizar a ligação entre um pacote de interesse e o *hash* da chave pública do publicador desejado e não com o *hash* do conteúdo. Esse mecanismo, chamado de D-SIDC pelos autores de [Gasti et al. 2012], já é implementado no CCNx. Os pacotes de interesse possuem um campo opcional onde o consumidor pode definir o *hash* da chave pública do publicador do conteúdo que ele deseja receber. Assim, os roteadores devem comparar o valor desse campo (se presente) com o *hash* da chave pública referenciada pelo pacote de dados correspondente e garantir que tal pacote só será enviado ao consumidor caso a comprovação seja válida.

Apesar do D-SIDC impedir que adversários envenenem os *caches* com conteúdos falsos, ainda é possível fazê-lo utilizando conteúdos corrompidos. Para isso, basta que os conteúdos tenham assinaturas não verificáveis, mas referenciem a chave pública indicada pelo consumidor no pacote de interesse.

Por fim, outra possibilidade é utilizar o S-SIDC e o D-SIDC em conjunto, com o objetivo de obter os benefícios de ambos. Uma maneira de fazer isso seria utilizar o D-SIDC para recuperar o primeiro *chunk* de um conteúdo grande. Se cada *chunk* contiver o valor dos *hashes* dos próximos *n chunks*, então o S-SIDC poderia ser utilizado para recuperá-los. Assim, as limitações do S-SIDC quanto a conteúdos dinâmicos e do D-SIDC quanto a proteção contra ataques de envenenamento de *cache* por conteúdos corrompidos são resolvidas.

### Verificação Probabilística de Assinaturas

Como a verificação das assinaturas de todos os conteúdos pelos roteadores é impraticável, uma abordagem interessante seria fazer com que os roteadores verificassem apenas as assinaturas de uma porção aleatória dos conteúdos armazenados em seu *cache*. Os conteúdos cujas assinaturas não fossem aprovadas na verificação seriam removidos do *cache*. Já os conteúdos com assinaturas válidas seriam marcados, de forma a evitar que outros roteadores verificassem sua assinatura novamente.

Outra abordagem é fazer com que todos os roteadores em um mesmo domínio administrativo dividissem igualmente a carga de verificação de assinaturas. Por exemplo, se existirem 4 roteadores em um domínio administrativo qualquer e 8 conteúdos cujas assinaturas precisam ser verificadas, então cada roteador deve realizar duas verificações. Para alcançar tal objetivo, [Gasti et al. 2012] apresentam um algoritmo detalhadamente.

### Alertas de Conteúdo Corrompido

O objetivo deste mecanismo é reduzir a frequência de verificação de assinaturas pelos roteadores, sem reduzir a resistência da rede contra ataques de envenenamento de *cache*. Para tanto, os roteadores devem continuar realizando a verificação probabilística das assinaturas de maneira independente, assim como explicado anteriormente. Entre-

 

tanto, mediante a falha de uma verificação, o roteador deve enviar um pacote de interesse especial por todas as suas interfaces, alertando assim seus vizinhos sobre o ocorrido. O nome de conteúdo presente no pacote de interesse de alerta deve ser parecido com `/ndn/warning/hCO`, onde `/ndn/warning/` é um prefixo reservado pela CCN e `hCO` é o *hash* do conteúdo cuja verificação da assinatura falhou. Além disso, o escopo do pacote de interesse de alerta é definido como 2, ou seja, o interesse só deve se propagar até o próximo salto. Quando um roteador receber o interesse de alerta, este deve verificar se existe algum conteúdo em *cache* cujo *hash* é igual a `hCO`. Em caso afirmativo, o conteúdo é removido do *cache*. Esse processo deve ser executado por todos os roteadores da rede.

Uma vez que os consumidores já realizam a verificação de todos os conteúdos que recebem, estes poderiam enviar *feedbacks* à rede com o objetivo de alertar os roteadores sobre conteúdos corrompidos. Entretanto, o envio de *feedbacks* apresenta os seguintes desafios: (1) não existe relação de confiança entre roteadores e consumidores, (2) consumidores são mais plausíveis de serem comprometidos do que roteadores e (3) não é possível identificar o consumidor que enviou um determinado *feedback*. Uma abordagem para proporcionar o *feedback* dos consumidores, proposta em [Gasti et al. 2012], é baseada em um valor probabilístico de confiança $T, T \in [0,1]$, associado a cada conteúdo presente no *cache* do roteador. Quando $T = 1$ significa que a assinatura do conteúdo foi verificada. Se $T \approx 0$ significa que o conteúdo deve ser removido do *cache*, no caso do mesmo estar cheio, ou deve ser selecionado para verificação da assinatura com probabilidade proporcional a $1 - T$. Aos novos pacotes de dados é atribuído o valor $T = 0,5$. O valor de $T$ é incrementado cada vez que o conteúdo é requisitado e enviado ao consumidor, e é decrementado cada vez que um *feedback* negativo é recebido.

## 3.5. Considerações Finais, Perspectivas Futuras e Problemas em Aberto

A CCN tem sido amplamente discutida por pesquisadores nos últimos anos. A adoção do paradigma orientado ao conteúdo, onde usuários requisitam conteúdos pelo nome e a rede se encarrega de encontrá-los onde quer que eles estejam, torna esta arquitetura mais adaptada ao novo perfil dos usuários e das aplicações. Para permitir uma maior eficiência na recuperação de conteúdos, os roteadores da CCN realizam *cache* dos mesmos, de maneira que requisições futuras não precisam ser encaminhadas até a fonte, reduzindo assim o tempo de resposta e a utilização da largura de banda no núcleo da rede.

Ao eliminar a associação entre os conteúdos e sua localização física, a CCN permite que a segurança dos conteúdos seja implementada de maneira mais simples. Ao invés de garantir a integridade e autenticidade do nó que armazena o conteúdo e do canal por onde este trafega, a CCN estabelece que todos os conteúdos devam ser assinados por seus publicadores. Dessa forma, para verificar se um conteúdo é autêntico e íntegro, basta obter a chave pública do publicador e verificar a assinatura do conteúdo.

Um dos principais problemas na Internet atual são os ataques DDoS. O modelo orientado à localização, no qual as redes TCP/IP estão fundamentadas, é bastante vulnerável a este tipo de ataque. Por outro lado, características como a agregação e o encaminhamento adaptativo de pacotes de interesse, tornam a CCN bastante resistente aos ataques DDoS comumente empregados contra a Internet atual.

Apesar de permitir a implementação da segurança de conteúdos de maneira mais

 

simples e de mitigar grande parte dos ataques DDoS encontrados na Internet atual, a CCN ainda permite que alguns de seus mecanismos sejam explorados por ataques projetados especialmente para esta arquitetura. Além disso, ataques largamente estudados envolvendo *caches*, como o *cache snooping*, passam a ter um maior impacto na CCN em relação a Internet.

O projeto *Named-Data Networking* - NDN, mantém o desenvolvimento da CCN e estabelece diversas áreas de pesquisa, onde uma delas é a segurança. Apesar da CCN não prover uma solução para todos os desafios em segurança de redes, ela estabelece novos mecanismos para a validação de conteúdos e a determinação de sua proveniência. Entretanto, conteúdos maliciosos ou indesejados contendo assinaturas válidas, podem ser gerados por publicadores legítimos [Zhang et al. 2010].

Para garantir a autenticidade da relação entre um publicador de conteúdos e sua chave pública, a CCN necessita que seja implementado algum mecanismo de gerenciamento de confiança. Apesar dos usuários serem livres para utilizar quaisquer modelos existentes, o projeto NDN sugere a utilização do sistema SDSI/SPKI [Zhang et al. 2010]. Neste contexto, uma questão importante é o processo de revogação de chaves. Uma vez que as chaves públicas também são tratadas como conteúdos, estas poderão ser armazenadas em *caches* na rede. Assim, quando um consumidor obtiver uma chave, este não saberá se a mesma é atual, ou se é uma cópia revogada obtida de algum *cache*.

O modo de funcionamento da CCN requer que a manutenção de estado em seus roteadores. Esta característica pode ser vista como uma vulnerabilidade explorada por atacantes para efetuar ataques DoS por inundação dos pacotes de interesses. Tais ataques, podem provocar o esgotamento dos recursos dos roteadores impedindo o atendimento de requisições de usuários legítimos.

Da mesma forma que a Internet atual, a CCN não provê um mecanismo de responsabilização para os consumidores de conteúdo. A responsabilização é definida como "a propriedade que garante que as ações de uma entidade possam ser unicamente rastreadas até esta entidade"[Shirey 2000]. Assim, apesar dos pacotes de dados serem assinados pelos publicadores, os pacotes de interesse podem ser emitidos por qualquer usuário com a garantia do anonimato. Como os pacotes de interesse podem ser usados como veículos nos ataques de negação de serviço, a fonte de tais ataques se torna ainda mais difícil de ser rastreada do que na Internet atual. Essa dificuldade poderia ser resolvida exigindo-se que os pacotes de interesse também fossem assinados. Porém, esta técnica implica em sérios problemas de privacidade. Dessa forma, cria-se uma forte tensão entre a detecção de ataques DoS e a garantia da privacidade dos usuários.

A CCN representa uma grande evolução em relação a Internet atual, principalmente no que diz respeito a segurança. Entretanto, novas tecnologias normalmente são acompanhadas por novas vulnerabilidades, que por sua vez podem originar novos ataques. Como mostrado neste minicurso, a CCN não é uma exceção e sua adoção em larga escala depende ainda da superação de diversos desafios, principalmente em segurança.

 

## Referências

[Abliz 2011] Abliz, M. (2011). *Denial of Service in Computer Networks: A Survey of Attacks and Defense Mechanisms*. LAP Lambert Academic Publishing, Germany.

[Arianfar et al. 2011] Arianfar, S., Koponen, T., Raghavan, B. e Shenker, S. (2011). On Preserving Privacy in Content-Oriented Networks. Em *ACM SIGCOMM Workshop on Information-Centric Networking*, páginas 19–24.

[Baugher et al. 2012] Baugher, M., Davie, B., Narayanan, A. e Oran, D. (2012). Self-Verifying Names for Read-Only Named Data. Em *INFOCOM Workshops'12*, páginas 274–279.

[Bishop 2003] Bishop, M. (2003). *Computer Security: Art and Science*. Addison-Wesley.

[Brito et al. 2012] Brito, G. M., Velloso, P. B. e Moraes, I. M. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet. Em *Minicurso - Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos - SBRC*, páginas 211–264.

[Che et al. 2002] Che, H., Tung, Y. e Wang, Z. (2002). Hierarchical web caching systems: modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305 – 1314.

[DiBenedetto et al. 2012] DiBenedetto, S., Gasti, P., Tsudik, G. e Uzun, E. (2012). ANDaNA: Anonymous Named Data Networking Application. páginas 251–260.

[Dingledine et al. 2004] Dingledine, R., Mathewsonn, N. e Syverson, P. (2004). Tor: The second-generation onion router. Em *The 13th USENIX Security Symposium*, páginas 21–21.

[Felten e Schneider 2000] Felten, E. W. e Schneider, M. A. (2000). Timing attacks on web privacy. Em *ACM Conference on Computer and Communications Security*, páginas 25–32.

[Fu et al. 2002] Fu, K., Kaashoek, M. F. e Mazières, D. (2002). Fast and secure distributed read-only file system. *ACM Trans. Comput. Syst.*, 20(1):1–24.

[Gasti et al. 2012] Gasti, P., Tsudik, G., Uzun, E. e Zhang, L. (2012). DoS & DDoS in Named-Data Networking. Em *Em submissão*.

[Ghodsi et al. 2011a] Ghodsi, A., Koponen, T., Raghavan, B., Shenker, S., Singla, A. e Wilcox, J. (2011a). Information-Centric Networking: Seeing the Forest for the Trees. Em *ACM Workshop on Hot Topics in Networks - HotNets*, páginas 11–16.

[Ghodsi et al. 2011b] Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P. e Shenker, S. (2011b). Naming in Content-Oriented Architectures. Em *ACM SIGCOMM workshop on Information-centric networking*, páginas 1–6.

[Ioannidis e Bellovin 2001] Ioannidis, J. e Bellovin, S. M. (2001). Pushback: Router-Based Defense Against DDoS Attacks.

 

[Jacobson et al. 2009] Jacobson, V., Smetters, D. K., Thornton, J. D. e Plass, M. F. (2009). Networking named content. Em *International Conference on emerging Networking Experiments and Technologies - CoNEXT*.

[Jacobson et al. 2012] Jacobson, V., Thornton, J. D., Plass, M., Briggs, N., Braynard, R. e Smetters, D. K. (2012). Networking Named Content. *Communications of the ACM*, 55(1):117–124.

[Koponen et al. 2007] Koponen, T., Shenker, S., Stoica, I., Chawla, M., Chun, B., Ermolinsky, A. e Kim, K. (2007). A data-oriented (and beyond) network architecture. Em *ACM SIGCOMM*, páginas 181–192.

[Kurose 2012] Kurose, J. (2012). Content-centric networking: technical perspective. *Communications of the ACM*, 55(1):116–116.

[Labovitz et al. 2010] Labovitz, C., Iekel-Johnson, S., MacPherson, D., Oberheide, J., Jahanian, F., Kalyanaraman, S., Padmanabhan, V. N., Ramakrishnan, K. K., Shorey, R. e Voelker, G. M. (2010). Internet inter-domain traffic. Em *ACM SIGCOMM*, páginas 75–86.

[Lagutin et al. 2010] Lagutin, D., Visala, K. e Tarkoma, S. (2010). Publish/Subscribe for Internet: PSIRP Perspective. Em *Towards the Future Internet - Emerging Trends from European Research*, chapter 8, páginas 75–84. IOS Press.

[Laufer et al. 2005] Laufer, R. P., Moraes, I. M., Velloso, P. B., Bicudo, M. D. D., Campista, M. E. M., Cunha, D. O., Costa, L. H. M. K. e Duarte, O. C. M. B. (2005). Negação de Serviço: Ataques e Contramedidas. Em *Minicurso - Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg*, páginas 1–63.

[Myers et al. 1999] Myers, M., Ankney, R., Malpani, A., Galperin, S. e Adams, C. (1999). X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. IETF Network Working Group RFC 2560.

[Neuman et al. 1993] Neuman, C., Kohl, J., Yu, T., Hartman, S. e Raeburn, K. (1993). The Kerberos Network Authentication Service (V5). Relatório técnico.

[Popescu et al. 2005] Popescu, B. C., Steen, M. v., Crispo, B., Tanenbaum, A. S., Sacha, J. e Kuz, I. (2005). Securely Replicated Web Documents. Em *IEEE International Parallel and Distributed Processing Symposium - IPDPS*, páginas 102–104.

[Pournaghshband e Natarajan ] Pournaghshband, V. e Natarajan, K. Em *International Conference on Security and Management - SAM*.

[Rd e Jones ] Rd, D. E. E. e Jones, P. E.

[Schwetzingen 2010] Schwetzingen, T. L. (2010). Security & Scalability of Content-Centric Networking. Tese de mestrado, Technische Universitat Darmstadt.

[Shirey 2000] Shirey, R. (2000). Internet Security Glossary.

 

[Smetters e Jacobson 2009] Smetters, D. e Jacobson, V. (2009). Securing Network Content. Relatório Técnico TR-2009-1, Xerox Palo Alto Research Center - PARC.

[Stallings 2006] Stallings, W. (2006). *Cryptography and Network Security - Principles and Practice, 4th Edition*. Prentice Hall.

[Telegraph e Committee 1991] Telegraph, I. e Committee, T. C. (1991). *CCITT Recommendation X.800: Data Communication Networks: Open Systems Interconnection (OSI); Security, Structure and Applications : Security Architecture for Open Systems Interconnection for CCITT Applications*. International Telecommunication Union.

[Wilcox-O'Hearn 2003] Wilcox-O'Hearn, Z. (2003). Names: Decentralized, secure, human-meaningful: Choose two. `https://lafsgateway.zooko.com/uri/URI:DIR2-RO:d23ekhh2b4xashf53ycrfoynkq:y4vpazbrt2beddyhgwcch4sduhnmmefdotlyelojxg4tyzllhb4a/distnames.htm`.

[Wählisch et al. 2012] Wählisch, M., Schmidt, T. C. e Vahlenkamp, M. (2012). Backscatter from the Data Plane - Threats to Stability and Security in Information-Centric Networking. Em *http://arxiv.org/abs/1205.4778*.

[Yi et al. 2012] Yi, C., Afanasyev, A., Wang, L., Zhang, B. e Zhang, L. (2012). Adaptive Forwarding in Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 42(3):62–67.

[Zhang et al. 2010] Zhang, L., Estrin, D., Bruke, J., Jacobson, V., Thornton, J., Smetters, D., Zhang, B., Tsudik, G., Claffy, K., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P. e Yeh, E. (2010). Named Data Networking (NDN) Project. Relatório Técnico NDN-0001, NDN.

[Zimmermann 1995] Zimmermann, P. R. (1995). *The official PGP user's guide*. MIT Press.

 