UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

Análise de Ataques e Mecanismos de Segurança em Redes Ad Hoc

Autora:

Natalia Castro Fernandes

Orientador:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Examinadores:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Marcelo Gonçalves Rubinstein, D.Sc.

Prof. Marcelo Luiz Drumond Lanza, M.Sc.

DEL
Dezembro de 2006

À minha família.

Agradecimentos

Agradeço, primeiramente, a Deus, por essa oportunidade. Agradeço também a minha família, por todo apoio e compreensão nestes cinco anos de faculdade. Não poderia esquecer dos amigos, que tornaram todo esse período inesquecível.

Aos professores, Otto, meu orientador, e Luís Henrique, pela paciência e pelos conselhos durante a execução deste trabalho, e a todos os meus amigos do Grupo de Teleinformática e Automação, o meu mais sincero obrigado, por toda a ajuda, disponibilidade e bom humor.

Aos professores do Departamento de Eletrônica e de Computação, agradeço pela amizade e por todo conhecimento de eletrônica e de vida que me foram passados.

Por fim, agradeço a todos que me incentivaram de forma direta ou indireta, me ajudando a completar a minha formação profissional.

Resumo do Projeto Final

Análise de Ataques e Mecanismos de Segurança em Redes Ad Hoc

Natalia Castro Fernandes

Dezembro de 2006

Orientador: Otto Carlos Muniz Bandeira Duarte
Departamento: Engenharia Eletrônica e de Computação

As redes ad hoc sem fio possuem vulnerabilidades específicas associadas principalmente à transmissão pelo ar como meio de comunicação, à ausência de infra-estrutura e ao encaminhamento colaborativo das mensagens. Nas redes ad hoc, além dos ataques convencionais às redes sem fio, o roteamento colaborativo possibilita novas ameaças de segurança e a ausência de infra-estrutura dificulta a criação de mecanismos de defesa simples e eficientes.

Este projeto apresenta os principais ataques às redes ad hoc e analisa os principais mecanismos de segurança utilizados para a proteção aos ataques, assim como os principais protocolos seguros específicos para redes ad hoc que foram propostos na literatura. Foi analisado o impacto de cada ataque sobre os protocolos *Optimized Link State Routing Protocol*(OLSR) e *Ad hoc On-demand Distance Vector Protocol*(AODV), por estes serem os mais utilizados atualmente. Além disso, analisou-se o *Secure Ad hoc On-demand Distance Vector Protocol*(SAODV) e o *Secure Optimized Link State Routing Protocol*(SOLSR), verificando a eficácia dos mecanismos de segurança propostos por eles, e a sobrecarga de processamento e dados gerado por esses protocolos devido às medidas de segurança. Baseado nesses resultados, foi possível determinar a viabilidade de instalação de uma rede ad hoc segura para ser disponibilizada ao público no Bloco H do Centro de Tecnologia.

Outra questão abordada no projeto foi a distribuição de chaves em redes ad hoc,

dado que essa rede tem como característica a ausência de infra-estrutura e a topologia dinâmica, o que dificulta a oferta de serviços na rede. Por esta razão, foi feito um estudo sobre criptografia de limiar aplicada a redes ad hoc, no intuito de promover disponibilidade do serviço e segurança de forma autônoma.

Palavras-Chave

Redes de Computadores

Redes Ad Hoc

Redes Sem-Fio

Segurança

Roteamento

Internet

Lista de Acrônimos

AC :	Autoridade Certificadora;
ACK :	*Acknowledgment*;
AES :	*Advanced Encryption Standard*;
AODV :	*Ad hoc On-demand Distance Vector Protocol*;
AODV-UU :	*Ad hoc On-Demand Distance Vector - Uppsala University*;
ARAN :	*Authenticated Routing for Ad-Hoc Networks*;
A-SAODV :	*Adaptative Secure Ad hoc On-Demand Distance Vector*;
CEFRIEL :	*ICT Center of Excellence For Research, Innovation, Education and industrial Labs partnership* ;
CPU :	*Central Processing Unit*;
CSMA :	*Carrier-Sense Multiple Access*;
DES :	*Data Encryption Standard*;
DSS :	*Digital Signature Standard*;
ETSI :	*European Telecommunications Standards Institute*;
GTA :	Grupo de Teleinformática e Automação;
IEEE :	*Institute of Electrical and Electronics Engineers*;
INRIA :	*Institut National de Recherche en Informatique et en Automatique*;
IP :	*Internet Protocol*;
IPSec :	*IP Security*;
KDE :	*K Desktop Environment*;
MAC :	*Medium Access Control*;
MANET :	*Mobile Ad hoc NETwork*;

MD5 : *Message-Digest algorithm 5*;

MID : *Multiple Interface Declaration*;

MPR : *MultiPoint Relay*;

OLSR : *Optimized Link State Routing Protocol*;

OSI : *Open Systems Interconnection*;

PKI : *Public-Key Infrastructure*;

RAM : *Randon Access Memory*;

RREP : *Route Reply*;

RREP-ACK : *Route Reply Acknowledgment*;

RREQ : *Route Request*;

RRERR : *Route Error*;

RSA : *Rivest-Shamir-Adleman*;

RTS/CTS : *Request to Send/Clear to Send*;

SAODV : *Secure Ad hoc On-Demand Distance Vector Protocol*;

SHA-1 : *Secure Hash Standard - 1*;

SOLSR : *Secure Optimized Link State Routing protocol*;

SRP : *Secure Routing Protocol*;

TC : *Topology Control*;

TDMA : *Time Division Multiple Access*;

TTL : *Time-to-Live.*

Sumário

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Capítulo I

Introdução

O cenário atual da Internet e das redes locais permitiu o desenvolvimento de redes móveis e a sua popularização. As tecnologias para redes móveis hoje são inúmeras e atingem grande parte da população planetária, principalmente devido às redes de celulares. O avanço da Internet e a procura por manter-se sempre conectado estão levando a investimentos para a instalação de redes sem-fio em ambientes empresariais e industriais, redes domésticas, de sensores e comunitárias. Os investimentos previstos para os próximos anos atingem grandes cifras e projeta-se novas aplicações como o controle automático do estoque doméstico, assistência e monitoração remota de funções vitais de idosos e convalescentes, jogos, lazer, entre outros.

O conhecimento das tecnologias envolvendo as redes de nova geração são de interesse estratégico, já que o acesso e a veiculação da informação e a difusão do conhecimento tornaram-se fatores primordiais na economia e na sociedade moderna. Estuda-se hoje o modelo que deve ser aplicado a Internet do futuro, que deve ser escalável, autônoma, prover um ambiente ubíquo e ser segura. Nesta rede, homens e também dispositivos (coisas) com capacidade de processamento e comunicação poderão se conectar a Internet em qualquer lugar e a qualquer momento, justificando sua denominação Internet de Coisas (*Internet of Things*).

Atualmente, muitos esforços já foram realizados para permitir a ampla utilização de

redes sem fio em todo o mundo sob diversos cenários. Entre os principais casos de sucesso, pode-se destacar a utilização de redes ad hoc para retomar a comunicação após o desastre causado pelo furacão Katrina nos EUA, ou ainda a utilização de redes sem fio infra-estruturadas com o projeto de *hotspot* nacional na Austrália, através do qual várias áreas do *Adelaide City Council*, aeroportos e lojas já possuem Internet sem fio disponível. Na área de redes em malha sem fio, que são redes sem-fio aonde alguns nós são previamente selecionados para formar um *backbone*, já existem dezenas de universidades e centros de pesquisa que montaram suas próprias redes.

No Brasil, existe uma crescente demanda por redes sem fio que permitam acesso à Internet nas universidades, ruas, centros comercias, aeroportos e, inclusive nas comunidades carentes, onde o seu uso é atrativo devido ao baixo custo de infra-estrutura e instalação.

Apesar de todas as vantagens das redes sem-fio, o seu uso pode ficar restrito na ausência de mecanismos de segurança que permitam o seu uso de forma adequada. Nas redes cabeadas essa é uma grande preocupação tanto com relação à privacidade dos dados como na prevenção contra ataques de negação de serviço. No entanto, a maioria das técnicas já existentes para essas redes não se aplicam a redes sem fio, tanto por o meio ser o ar quanto por características próprias da tecnologia, como o sistema de roteamento. Assim, para tornar possível o uso das redes sem fio, é necessária uma atenção especial na aplicação e desenvolvimento de técnicas e políticas de segurança próprias a essas redes.

I.1 Redes Ad Hoc

As redes ad hoc sem-fio, também conhecidas como *Mobile Ad hoc NETworks - MANETs*, são redes sem infra-estrutura e que possuem roteamento colaborativo. A ausência de infra-estrutura significa que essas redes não possuem pontos de acesso ou estações-base, como acontece, por exemplo, nas redes de celulares. O roteamento colaborativo permite que nós que não estejam no mesmo alcance de transmissão de rádio se comuniquem através de múltiplos saltos, através da colaboração dos nós intermediários. Outra

característica importante das redes ad hoc sem-fio é que a topologia da rede pode mudar dinamicamente devido à mobilidade dos nós. Nas Figuras I.1 e I.2 estão representadas respectivamente uma rede com ponto de acesso, na qual os nós precisam estar no alcance do ponto de acesso para se comunicar, e uma rede ad hoc,na qual os nós conversam através de múltiplos saltos.

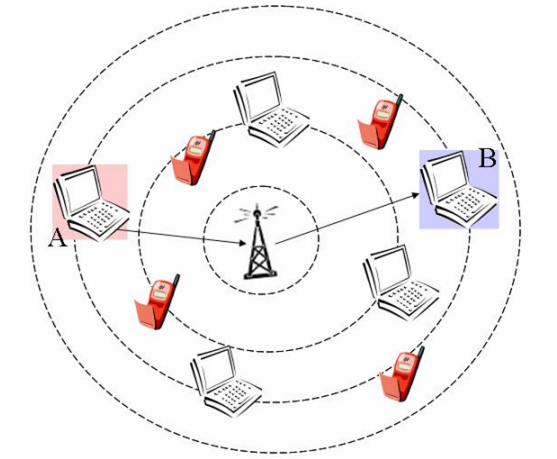

Figura I.1: Exemplo de rede infra-estruturada.

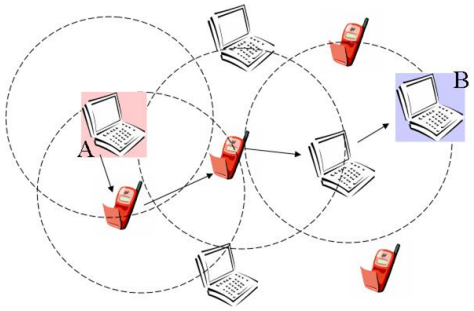

Figura I.2: Exemplo de rede ad hoc.

As redes ad hoc sem fio possuem como grande vantagem o baixo custo de instalação e facilidade de configuração, o que justifica o seu uso, por exemplo, no ambiente universitário. Cabe observar que uma rede ad hoc pura não poderia dar acesso à Internet, mas para aumentar o seu uso, vários protocolos de roteamento permitem que qualquer nó seja *gateway*, dando acesso à Internet a toda a rede.

Devido a características como o meio de comunicação sem fio, a ausência de infra-estrutura e o roteamento colaborativo em múltiplos saltos, as redes ad hoc se tornam

alvos potenciais de diversos tipos de ataques. Assim, a segurança é um dos seus pontos cruciais.

A utilização do ar como meio de transmissão torna a rede susceptível a diversos ataques, que vão desde uma simples espionagem das mensagens até interferências com a criação, modificação e destruição das mensagens em trânsito. As redes cabeadas são consideradas mais seguras, pois um atacante tem maior dificuldade para obter acesso ao meio físico e também para transpor as barreiras formadas pelos *firewalls*. Os ataques às redes sem fio podem vir de várias direções e alvejar qualquer nó da rede, bastando que o nó atacado esteja no alcance da transmissão do nó atacante. Dessa maneira, é possível que um nó malicioso tenha acesso a informações sigilosas, possa alterar mensagens em trânsito ou ainda tentar se passar por outros nós da rede. Portanto, o preço que se paga pelas facilidades oferecidas pela comunicação sem fio é a ausência de uma barreira de defesa clara. Assim, cada nó da rede deve estar preparado para lidar direta ou indiretamente com ações maliciosas.

Devido à ausência de infra-estrutura, as redes ad hoc exigem colaboração distribuída dos nós da rede para o encaminhamento das mensagens. Nas redes ad hoc, todos os nós participam do protocolo de roteamento, pois também desempenham a função de roteador. Além disso, estes nós roteadores estão sob o controle dos usuários da rede, e não de administradores. Isso possibilita a criação de novos ataques que visam as vulnerabilidades dos algoritmos cooperativos, o que significa que as principais particularidades das redes ad hoc estão na camada de rede. Desta forma, os protocolos de roteamento das redes ad hoc devem ser robustos a novos tipos de ataques.

Outro aspecto importante a ser considerado nas redes ad hoc é a ausência de centralização e de infra-estrutura. Não existem dispositivos dedicados a tarefas específicas da rede como, por exemplo, realizar a autenticação ou distribuir endereços dinamicamente. Apesar de a descentralização ter como vantagem a robustez, devido à inexistência de pontos únicos de falha, a ausência de infra-estrutura dificulta a aplicação das técnicas convencionais de autorização de acesso e de distribuição de chaves. Isto dificulta a tarefa de distinguir os nós confiáveis dos nós não-confiáveis, pois nenhuma associação segura

prévia pode ser assumida.

A mobilidade introduz outros obstáculos importantes à implementação de mecanismos de segurança, devido às constantes alterações na topologia da rede. Esta dinamicidade implica em novos nós que se tornam vizinhos e antigos nós que deixam de ser vizinhos. A mobilidade pode até causar particionamentos na rede. Outro problema que surge é que com os recursos móveis, não há como saber acessar diretamente servidores, e nem a divisão clássica de redes e sub-redes se aplica corretamente, pois como o nó é móvel, ao mudar de posição, ele poderia sair da área de acesso da sua sub-rede e perder conexão. Assim, os mecanismos de segurança devem se adaptar dinamicamente às mudanças na topologia da rede e ao movimento dos nós entrando e saindo da rede. Além disso, deve-se ressaltar que as redes ad hoc móveis são, em geral, compostas por dispositivos portáteis, possuindo, assim, restrições de energia, processamento e memória. Com isso, as MANETs estão sujeitas a diferentes ataques de negação de serviço que visam esgotar os recursos dos nós a fim de prejudicar o funcionamento da rede.

Em suma, as redes ad hoc móveis possuem vulnerabilidades justamente nas suas principais características, como a utilização do ar como meio físico, as alterações dinâmicas da topologia de rede, a cooperação entre os nós, a distribuição de tarefas, a ausência de um ponto central de monitoramento e gerenciamento e a falta de uma linha de defesa clara. Essas vulnerabilidades são inerentes às MANETs, havendo portanto a necessidade de se criar mecanismos de defesa específicos. É importante notar que a solução desses problemas não resolve algumas das questões clássicas de segurança, como proteção contra vírus ou mal-uso das aplicações, mas apenas colocam as redes ad hoc com o mesmo grau de segurança que uma rede cabeada. Assim, são ainda necessários todos os métodos de segurança da camada de aplicação em funcionamento para garantir uma rede realmente segura.

I.2 Organização do Projeto

Neste projeto, primeiramente, serão expostas as principais características do roteamento em redes ad hoc no Capítulo II, tomando como exemplo os seus dois principais protocolos de roteamento, o OLSR e o AODV. O Capítulo III descreve as principais formas de ataques às redes ad hoc móveis e algumas soluções para esses ataques. O Capítulo IV apresenta alguns conceitos básicos de segurança e os principais mecanismos para fazer distribuição de chaves segura em redes ad hoc, enquanto o Capítulo V apresenta os protocolos especialmente projetados para prover segurança em redes ad hoc analisados neste projeto, avalia o seu desempenho, quando comparados aos inseguros, e apresenta as conclusões finais do trabalho.

Os Capítulos I, III e IV deste projeto foram, em parte, adaptados do minicurso: Fernandes, N. C., Moreira, M. D. D., Velloso, P. B., Costa, L. H. M. K., and Duarte, O. C. M. B. - "Ataques e Mecanismos de Segurança em Redes Ad Hoc", Minicursos do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg'2006, Santos, SP, Brazil, pp. 49-102, August 2006, do qual Natalia é co-autora.

Capítulo II

Roteamento em Redes Ad Hoc

Redes ad hoc possuem uma tecnologia relativamente nova, e dadas as inúmeras dificuldades de roteamento, muitos protocolos foram propostos, mas sem que se observasse a questão de segurança. Assim, para entender os principais ataques e as melhorias propostas, é necessário primeiramente analisar o funcionamento dos primeiros protocolos propostos, que ainda são os mais utilizados.

Na classificação dos protocolos de roteamento existem dois grupos principais: o dos protocolos pró-ativos e o dos reativos. Protocolos pró-ativos mantêm rotas para todos os destinos da rede, enquanto que protocolos reativos só buscam as rotas quando elas são requisitadas. Dentre os pró-ativos, destaca-se o *Optimized Link State Routing Protocol* (OLSR) [1], que mantém um mapa da topologia sempre atualizado e possui rota para qualquer destino na rede. Para o cálculo das rotas ele utiliza o algoritmo de Dijkstra [2]. No grupo dos reativos, um dos principais exemplos é o *Ad hoc On-demand Distance Vector Protocol* (AODV) [3], que calcula o melhor caminho utilizando o algoritmo de Bellman-Ford distribuído [2].

II.1 *Optimized Link State Routing Protocol* (OLSR)

O OLSR [4] é um protocolo pró-ativo para redes ad hoc móveis baseado em estado de enlace. Seu funcionamento básico é feito por um monitoramento da rede para saber quais são os membros ativos e então calcular a tabela de roteamento. Assim, ele tem uma resposta rápida a requisições de rota, levando vantagem sobre os protocolos reativos. Por outro lado, suas desvantagens estão na ocupação da rede com constantes mensagens para descobrir os membros da rede, e na ocupação constante da CPU e da memória para fazer a atualização da tabela de roteamento.

O OLSR não gera nenhum tráfego além do esperado com as mensagens periódicas em resposta a falhas de enlace, devido a seu controle já tratar esse problema.

O protocolo mantém rotas para todos os nós da rede e, assim, é mais adequado a redes aonde os nós se comunicam constantemente, com pares de comunicação diferentes. Além disso, o OLSR se adapta melhor a ambientes de redes de grande porte densas, devido a sua pró-atividade e o uso do mecanismo de controle de inundação. O OLSR funciona de forma totalmente distribuída, não dependendo de pontos centrais para a troca de informações.

Como as mensagens de controle são trocadas periodicamente, não existe necessidade explícita de um controle de transmissão confiável. O protocolo é robusto mesmo com a perda de algumas dessas mensagens, o que é comum nos enlaces de rádio. Além disso, elas possuem um número de seqüência que garante que sempre será utilizada a informação mais nova sobre o enlace.

Com relação à mobilidade, o OLSR costuma apresentar resultados melhores que os protocolos reativos, pois sempre a informação mais recente é utilizada para o roteamento. Assim, dependendo da velocidade do nó e da freqüência de emissão de mensagens de controle, os vizinhos serão capazes de seguir o nó e retransmitir a mensagem.

Para fazer um controle de inundação, no OLSR, cada nó seleciona entre seus vizinhos os *multipoint relays* (MPR). Esses são nós especiais que servem para controlar a inundação da rede. A seleção destes nós especiais segue um algoritmo que permite a escolha de

um grupo de nós de forma que, se apenas esse grupo retransmitir, toda a rede deve ser alcançada. Dessa forma, é feito um mecanismo eficiente de controle de tráfego na rede, reduzindo-se o número de pacotes que circulam por ela. Para ser escolhido como MPR, o nó necessita ter um enlace bidirecional com o nó que o escolheu, pois todas as rotas da rede serão traçadas pelos MPRs.

II.1.1 *Multipoint Relays*

Esse é um mecanismo criado para evitar transmissões redundantes nas inundações, uma vez que todos os nós deveriam receber o mesmo pacote de cada um de seus vizinhos. A idéia do MPR consiste de evitar inundações localmente, permitindo um bom tráfego global. De fato, essa técnica melhora o uso da banda por eliminar transmissões desnecessárias na rede e também por determinar qual o menor caminho a se seguir no roteamento.

Os MPRs não foram propostos originalmente no OLSR, mas têm uma origem mais antiga, tendo sido utilizados no padrão de Hiperlan tipo 1 da ETSI (*European Telecommunications Standards Institute*) [5], sendo implementados na camada MAC.

Para o funcionamento dessa técnica, deve-se levar em consideração que esse é um algoritmo distribuído, que troca informações através de mensagens, e que cada nó apenas tem conhecimento dos nós vizinhos e dos vizinhos desses, que seriam seus vizinhos por dois saltos, como exemplificado na rede da Figura II.1. Nessa, as linhas representam a alcançabilidade do nó. Assim, existe um conhecimento da rede local, e sempre que um pacote for recebido, o nó precisa decidir se deve reenviá-lo ou não.

Sempre que um nó deseja enviar um pacote por inundação, ele transmite o pacote no ar, e apenas os seus vizinhos previamente selecionados como MPR devem retransmitir o pacote. Como o meio é o ar, após a primeira retransmissão, todos os nós no alcance também receberão esse pacote, mas apenas os que forem MPRs dos MPRs do nó inicial devem retransmitir a mensagem. Esse processo continua até que toda a rede seja atingida.

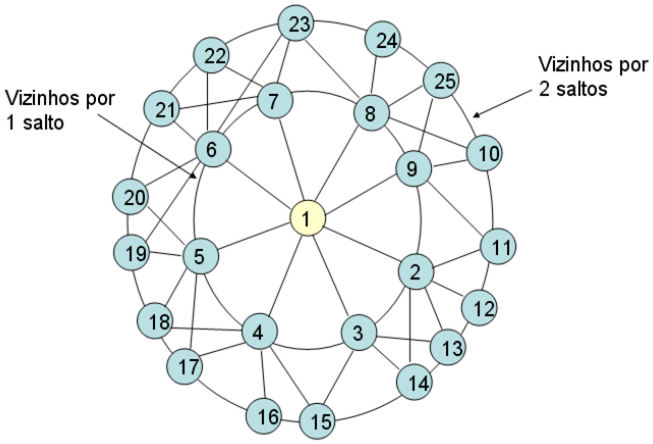

Figura II.1: Exemplo de rede ad hoc.

A seleção dos MPRs é feita por cada nó e garante que todos os vizinhos por dois saltos tenham enlace bidirecional com o nó emissor através do conjunto de MPRs selecionado. O algoritmo de seleção dos MPRs não precisa ser ótimo, desde que possa escolher um conjunto de MPRs que consiga fazer um controle de inundação.

Na inicialização da rede, todos os nós são considerados MPR's e a partir da troca de mensagens de controle é que se determina um conjunto menor de nós para exercer essa função. O conjunto de MPRs é recalculado quando é detectada uma mudança na vizinhança composta por nós distantes da origem por um ou dois saltos, seja pela falha de um enlace ou pela entrada de um novo vizinho.

O algoritmo para o cálculo do MPR é heurístico. Para realizá-lo, define-se:

N = Conjunto dos vizinhos por um salto.

N_2 = Conjunto dos vizinhos por dois saltos, excluindo-se os membros de N e o próprio nó de origem.

MPR = Conjunto de vizinhos selecionados como MPR.

$D(x)$ = Número de vizinhos com enlace simétricos de x excluindo-se todos os membros de N e o nó de origem, sendo x um membro de N.

$A(x)$ = Número de vizinhos de x em N_2, sendo x um membro de N.

Simplificadamente, seu funcionamento é dado pelos seguintes passos:

1. Preencher os conjuntos N e N_2 e deixar vazio o conjunto MPR.
2. Calcular $D(x)$ para todo x que pertença a N.
3. Selecionar em N todos os nós que sejam os únicos a atingir algum membro em N_2 e transferí-los para o conjunto MPR.
4. Excluir de N_2 todos os nós vizinhos dos nós em MPR.
5. Calcular a alcançabilidade, que representa o número de nós em N_2 que cada nó de N cobre.
6. Escolher o nó em N que possui a maior alcançabilidade entre os vizinhos em N_2. Em caso de empate, escolher o que possui o maior $D(x)$. Transferir o membro de N escolhido para MPR.
7. Excluir de N_2 todos os nós vizinhos do novo membro em MPR.
8. Se N_2 não estiver vazio, voltar ao passo 5.
9. Se N_2 estiver vazio, fim do algoritmo.

Como exemplo, tem-se a topologia de rede da Figura II.2a. Nela, todos os enlaces simétricos estão representados por linhas. Inicialmente, os conjuntos seriam dados por:

$N_2 = \{10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25\}$

$N = \{2, 3, 4, 5, 6, 7, 8, 9\}$

$D(x) = \{4, 3, 4, 4, 5, 3, 4, 3\}, x \in N$

$MPR = \{\}$

Pela observação da rede, é possível perceber que existem dois nós, sendo eles o 4 e o 8, que são os únicos a atingir os vizinhos 16 e 24 respectivamente. Assim, eles devem ser escolhidos como MPR, e os conjuntos podem ser atualizados para:

$MPR = \{4, 8\}$

$N = \{2, 3, 5, 6, 7, 9\}$

$N_2 = \{11, 12, 13, 14, 19, 20, 21, 22\}$

A representação gráfica dessa fase pode ser vista na Figura II.2b. Nesta figura, os nós 8 e 4 foram escolhidos como MPR por serem os únicos vizinhos de nós a dois saltos de distância da origem, e os nós que são alcançados por 8 e 4 também foram destacados. A partir desses valores de MPR, N e N_2, deve-se calcular a alcançabilidade A(x) para selecionar o novo MPR.

$A(x) = \{4, 2, 2, 4, 2, 1\}$

Por esse conjunto, os nós 2 e 6 são os candidatos a MPR. Como D(6) $>$ D(2), então 6 é o novo MPR. Os conjuntos passam a ser:

$MPR = \{4, 8, 6\}$

$N = \{2, 3, 5, 7, 9\}$

$N_2 = \{11, 12, 13, 14\}$

Na Figura II.2c pode ser vista essa representação, que permite calcular a nova alcançabilidade.

$A(x) = \{4, 2, 0, 0, 1\}$

Com base nesse novo conjunto, escolhe-se o nó 2 como MPR, resultando na Figura II.2d e nos conjuntos a seguir:

$MPR = \{4, 8, 6, 2\}$

$N = \{3, 5, 7, 9\}$

$N_2 = \{\}$

Assim, o algoritmo termina e é possível alcançar toda a rede a partir do nó 1 com base nos nós MPRs escolhidos, como se pode observar na Figura II.2d.

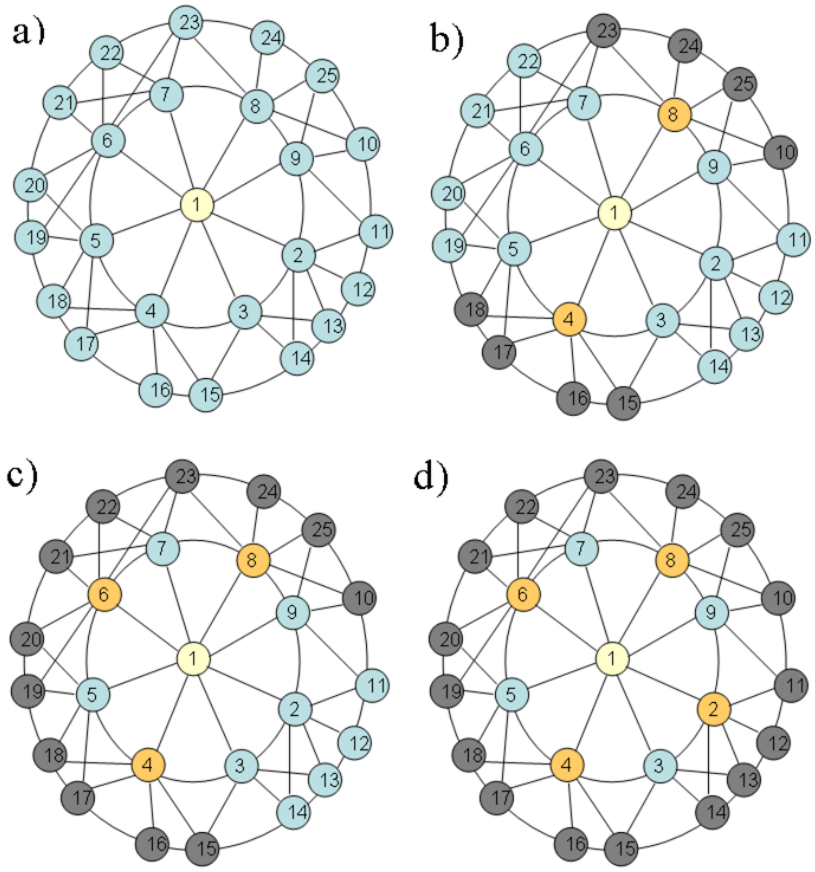

Figura II.2: Exemplo de escolha de MPRs.

Para um nó repassar um pacote, ele também precisa seguir algumas regras, sendo elas:

- Somente MPR's do último nó emissor repassam a mensagem, como pode ser visto na Figura II.3. Assim se realiza um controle de inundação eficiente.
- A mensagem só é repassada se ela ainda não tiver sido recebida. Assim, cópias da mesma mensagem não são repassadas e o caminho mais rápido do emissor até o receptor é garantido.

As rotas para todos os destinos são calculadas sempre passando pelos MPR's. Para tanto, é necessário que cada nó transmita periodicamente a informação sobre os nós que ele selecionou como MPR, dentro da sua lista de vizinhos. Sempre que um nó recebe uma dessas mensagens deve recalcular a sua tabela de roteamento. Dessa forma, as rotas no

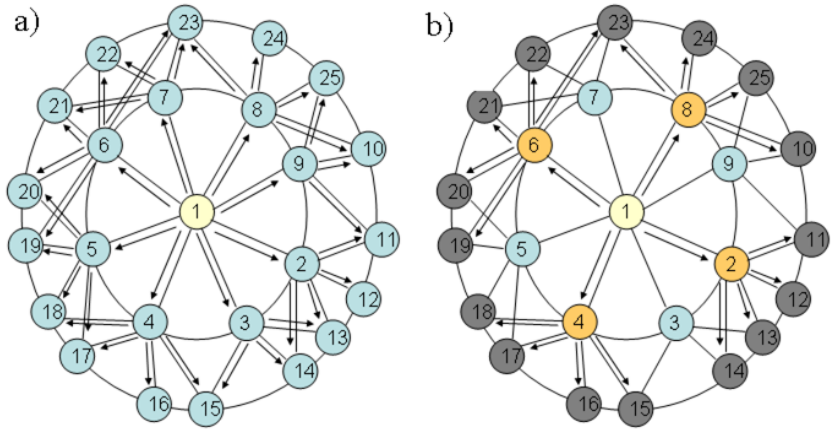

Figura II.3: a) Inundação sem MPR; b) Inundação com MPR's selecionados.

OLSR são uma seqüência de saltos entre a fonte e o destino. Esse tipo de escolha garante que todas as rotas serão bidirecionais, já que só podem ser escolhidos como MPR's nós com enlaces entre fonte e vizinhos por dois saltos com essa característica.

As informações sobre os MPR's do nó são guardadas em uma tabela própria. Cada conjunto de MPR's recebe um número de seqüência, para informar qual é o conjunto de MPR's mais novo. Sempre que o conjunto é recalculado, esse número de seqüência é incrementado.

II.1.2 Mensagens

O OLSR possui três tipos de mensagens que devem estar presentes em todas as suas implementações, sendo elas HELLO, TC e MID. As mensagens de HELLO são responsáveis pela detecção do tipo de enlace, detecção de vizinhança e sinalização de *Multipoint Relays* (MPRs) . A mensagem TC (*Topology Control*) declara a topologia, anunciando os estados de enlace. Por fim, as mensagens de MID (*Multiple Interface Declaration*) declaram a existência de múltiplas interfaces no nó.

Outros tipos de mensagem também são permitidos, seja para funções de conservação de energia, roteamento *multicast*, suporte para enlaces unidirecionais, ou determinação automática de endereços, entre outros.

- Mensagens de HELLO

As mensagens de HELLO permitem a detecção dos vizinhos e dos enlaces, o que é necessário para a seleção dos MPR's.

Cada nó deve detectar os vizinhos com os quais possui um enlace bidirecional, pois a instabilidade do meio pode acarretar em alguns enlaces unidirecionais. Assim, todos os enlaces devem ser testados nas duas direções para serem considerados válidos. Para realizar essa tarefa são utilizadas as mensagens de HELLO, que contêm as informações sobre os vizinhos, os nós que já foram ouvidos e o estado dos enlaces. Essas mensagens são enviadas em *broadcast* periodicamente, sendo recebidas por todos os vizinhos de um salto.

A lista com os vizinhos que já foram ouvidos é que permite o reconhecimento da bidirecionalidade do enlace. Se o nó A escuta um HELLO do nó B, o nó A insere B em sua lista de nós escutados. Se B escutar o HELLO de A, verá que A o escutou e que ele é capaz de escutar A e, assim, o enlace entre os dois é bidirecional. B irá, então, inserir A na sua lista de vizinhos, para que A também perceba que o enlace é bidirecional. Esse processo está representado na Figura II.4. Cabe mencionar que não é necessário anunciar sempre todos os vizinhos, pois isso poderia gerar mensagens muito grandes. Basta que os vizinhos sejam anunciados dentro de um período máximo pré-determinado de atualização.

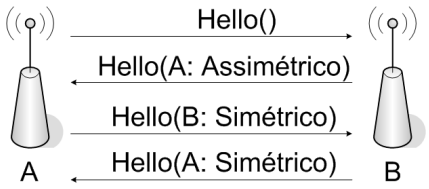

Figura II.4: Troca de dados entre A e B para a detecção de vizinhos com enlace bidirecional.

As mensagens de HELLO servem ainda para o cálculo e anúncio dos MPR's e a determinação das listas de vizinhos por um e dois saltos, o que é possível de se descobrir pelos nós que são escutados e os que são anunciados no pacote, respectivamente.

- Mensagens de TC

As mensagens de TC (*Topology Control*) tem como objetivo transmitir informações sobre a topologia que permitam que a tabela de roteamento seja calculada, e devem ser transmitidas por inundação periodicamente. O seu conteúdo é uma lista de nós que selecionaram o nó emissor da TC como MPR. Isso significa que é possível que alguns nós nunca enviem TC's, pois não foram selecionados por nenhum nó como MPR. As mensagens de TC, assim como as de HELLO, não precisam trazer a listagem de nós completa sempre, desde que dentro de um período de validade da rota esses dados sejam reenviados. Caso o conjunto de MPR's mude, a TC deve ser enviada antes do período normal de envio dessa mensagem.

Cada nó na rede guarda uma Tabela de Topologia na qual ele armazena todos os dados obtidos com as TC's. Essa tabela, que é a base para o cálculo das rotas a serem utilizadas, possui como campos o endereço de um possível nó de destino, que são todos os nós listados nas TC's, o endereço do último nó para chegar naquele destino, que é o endereço do emissor da TC, o número de seqüência associado ao conjunto de MPR's e um período de expiração da rota.

- Mensagens de MID

As mensagens de MID são geradas para a declaração de múltiplas interfaces do nó que estejam utilizando o OLSR, caso elas existam. Por essa razão, nem todos os nós emitem MID's na rede. Com essas mensagens é possível associar as diversas interfaces que pertencem ao mesmo nó para o cálculo de rotas. Como as mensagens de HELLO e TC, as MID's não precisam trazer a lista completa das interfaces, desde que os dados sejam retransmitidos dentro de um período máximo de atualização.

As MID's são inundadas em toda a rede, para que o cálculo do roteamento seja correto em todos os nós.

II.1.3 Cálculo das Rotas

A partir dos dados da Tabela de Topologia, é possível calcular a Tabela de Roteamento. O cálculo é feito do destino para a origem, pois a informação obtida com a Tabela de Topologia só inclui o último nó para chegar naquele destino. O algoritmo utilizado para o cálculo do menor caminho é o Dijkstra.

O cálculo da tabela é refeito sempre que uma nova mudança é anunciada nas mensagens de TC, o que pode ser originado por mudanças nas vizinhanças por um ou dois saltos de qualquer nó da rede. Cabe observar que a modificação da tabela de roteamento não gera mensagens de controle para serem transmitidas pela rede, já que todos os nós recebem as mensagens de TC e o cálculo da tabela é local.

Como resultado deste processo, cada nó obtém sua tabela, com rotas para todos os nós da rede. Para encaminhar pacotes, ele consulta a tabela, onde encontrará o próximo nó para o qual ele deve encaminhar. Não é necessário inserir todo o caminho escolhido na mensagem, pois, como os pacotes de TC e MID são enviados por inundação, existe um sincronismo entre as bases de dados dos nós. Assim, ao receber uma mensagem, o nó já sabe para quem deve repassar, sempre através do menor caminho bidirecional. Esta propriedade é garantida devido ao fato de que nas mensagens de TC só existem dados sobre os enlaces MPR's, e o cálculo das rotas é feito sobre a Tabela de Topologia.

Portanto, após esse processo, todos os nós possuem rotas para todos os outros nós da rede de forma pró-ativa. Sempre que se desejar enviar um pacote, esse será encaminhado diretamente, sem a necessidade de buscas por rotas, como é o caso dos protocolos reativos, garantindo rapidez de resposta.

II.2 *Ad hoc On-demand Distance Vector Protocol* (AODV)

O AODV é um protocolo de roteamento ad hoc reativo, ou seja, que procura as rotas quando elas são necessárias, e as mantém armazenadas apenas enquanto ainda são necessárias. Para evitar o problema de *loops* na rede, o AODV utiliza números de seqüência que garantem que a rota que está sendo utilizada é a mais nova que existe.

O AODV possui três tipos de mensagens básicas, sendo elas o *Route Request* (RREQ), o *Route Reply* (RREP) e o *Route Error* (RERR). Com essas mensagens, ele realiza o roteamento através da busca por destinos desconhecidos com o RREQ e o RREP e comunicação de erros com o RERR. Apesar de possuir uma resposta mais lenta, pois é necessário buscar a rota no momento que ela é requisitada, sua vantagem é a diminuição do *overhead* de mensagens de controle na rede. Além disso, o AODV necessita de menos memória que o OLSR, pois ele só armazena as rotas que estão sendo utilizadas, e não a topologia completa da rede.

II.2.1 Descoberta de Rotas

Sempre que um nó deseja descobrir a rota para um determinado nó, ele inunda a rede com um RREQ. Cada nó que recebe um RREQ o reencaminha e guarda uma rota reversa para a origem. O AODV supõe que todos os enlaces são simétricos, o que pode representar uma fonte de erros em redes sem-fio, onde a transmissão por rádio, devido a interferências do ambiente, pode levar à existência de enlaces unidirecionais. Cabe ressaltar que apenas o primeiro RREQ é encaminhado, e que as demais cópias desse pedido são descartadas.

Quando o nó destino recebe o primeiro RREQ, ele estabelece a rota reversa para o nó que requisitou a rota e envia um pacote de RREP. O RREP, ao contrário do RREQ, é transmitido em *unicast*, e não por inundação, pois se supõe que a rota reversa, criada pela passagem do RREQ, é funcional. Assim, o nó encaminha o RREP para o nó que enviou-lhe o RREQ, e assim por diante, até que se atinja o nó de origem do RREQ. Os

nós intermediários, ao receberem o RREP, devem consultar sua tabela de roteamento e encaminhar o pacote para o nó indicado, até que se atinja o nó de origem do RREQ. Caso o nó de destino do RREQ posteriormente receba um outro RREQ da mesma fonte, ele observa o número de seqüência do pedido. Caso o número de seqüência seja superior ao do último pedido, ele responde com um novo RREP, pois isso indica que a primeira rota foi perdida e é necessário buscar uma nova. Caso o número de seqüência seja o mesmo de uma mensagem já recebida, ele verifica o número de saltos. Caso o número de saltos seja maior ou igual ao número de saltos anterior, a mensagem é descartada. Caso contrário, a nova mensagem representa uma rota menor e deve ser utilizada como a melhor rota. Assim, o nó destino do RREQ deixa de utilizar a rota estabelecida com o primeiro RREQ e passa a utilizar a nova rota.

Um exemplo desse procedimento está na Figura II.5. Nesta, o nó F representa a fonte e D o destino que se deseja encontrar. Na Figura II.5a, o nó F envia em broadcast o pacote de RREQ, representado pelas setas cheias. O único nó em seu alcance é o nó 1, que ao receber o pacote, o reencaminha, atingindo os nós 2, 8, 7 e F, como pode ser visto na Figura II.5b. Como o nó F é o próprio emissor da mensagem, ele descarta o pacote que recebe de 1, o que está representado pela seta pontilhada. Em seguida, os nós 2, 8 e 7 concorrem pelo meio para decidir quem deve ser o primeiro a enviar o pacote, o que pode ser feito pelo mecanismo RTS/CTS (*Request to Send/ Clear to Send*) definido pela norma IEEE 802.11. Supondo que o nó 2 consiga o acesso ao meio primeiro, como na Figura II.5c, ele encaminha RREQ. O nó 1 e o nó 8 descartarão essa mensagem, pois já a receberam anteriormente. Sequencialmente, pode ser visto em d, e, f, g e h que os nós 7, 8, 3, 6 e 5 reencaminham o RREQ, até o momento que o destino D recebe o RREQ. Ele então armazena a rota vinda pelo nó 5, com 5 saltos, como a melhor opção de rota entre F e D. Em i, o nó D e o nó 4 competem pelo acesso ao meio. O nó D deseja enviar o RREP e o nó 4 o seu RREQ. Supondo que o nó 4 ganhe o acesso ao meio primeiro, ele irá encaminhar o RREQ. Quando D receber, ele verá que o número de seqüência é o mesmo, mas que o número de saltos foi reduzido para 4, indicando que esta nova rota é melhor que a anterior. Assim, o nó D enviará no seu RREP a indicação da rota que passa por 4 e não por 5. Caso D tivesse tido acesso ao meio antes de 4, ele teria enviado o RREP, e,

em seguida, ao receber o novo RREQ, ele criaria um novo RREP e o enviaria pela rota menor.

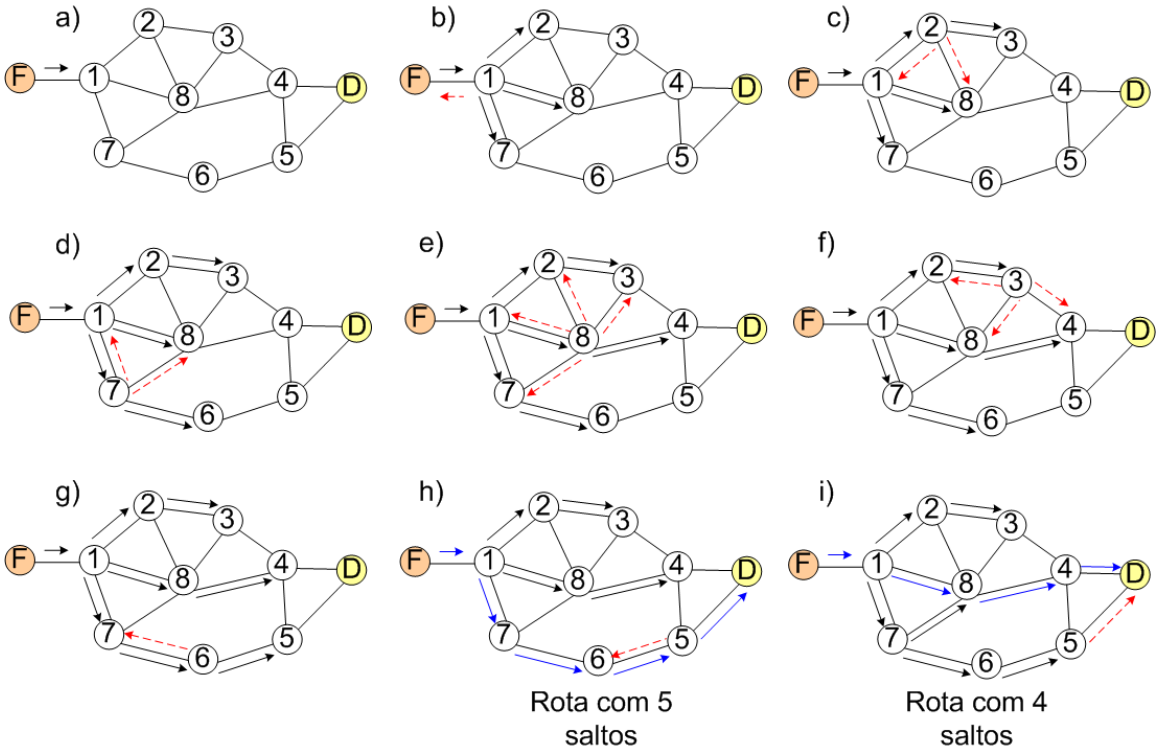

Figura II.5: Processo de descoberta da rota do AODV.

O AODV permite, ainda, que os nós intermediários respondam aos RREQ's, no caso de conhecerem uma rota para o destino. Assim, ao invés de o nó encaminhar o RREQ, ele o responde com as suas informações sobre o nó de destino. Isso permite que a rota seja obtida de forma mais rápida, evitando atrasos de transmissão. Nesses casos de resposta intermediária, existe a opção de se utilizar o *gratuitous* RREP. Este acontece quando o RREQ vem com o *flag* G setado, e implica na emissão de um RREP também para o destino, para comunicá-lo de que uma resposta já foi enviada para a origem e para o destino obter a rota reversa para a origem. Enquanto a rota permanecer ativa, ou seja, enquanto existirem pacotes sendo transmitidos através dela, ela será mantida. Quando a fonte deixa de enviar pacotes por aquela rota, é esperado um período curto pré-determinado, e após esse tempo a rota é excluída da tabela.

II.2.2 Mensagens de Erro

Uma vez que a rede considerada é uma rede móvel, ela está exposta a problemas como quebras de enlace, pois um nó, ao se mover, pode sair do alcance dos seus nós vizinhos. Isto pode ocorrer tanto durante um momento que aquele enlace não esteja sendo usado, quanto em momentos que o enlace está sendo utilizado para a transmissão de dados. Quando o segundo caso ocorre, é necessário que a fonte seja notificada que aquela rota não está mais disponível e que é necessário descobrir novas rotas. Assim, o nó que detectou a quebra de enlace envia um RERR (*Route Error*) para a fonte, comunicando os destinos que não estão mais disponíveis. Caso a fonte ainda precise enviar dados para o destino, ela necessitará, primeiro, realizar um novo processo de descoberta de rota, inundando a rede com um RREQ.

II.2.3 Outras mensagens

Route Reply Acknowledgment (RREP-ACK)

O RREP-ACK é utilizado em situações especiais. Ele deve ser enviado em resposta a um RREP com o flag A setado. Essa situação acontece tipicamente quando o RREQ, que é enviado por *broadcast*, tem um alcance diferente do RREP, que é enviado em *unicast*.

Detecção de Vizinhos com HELLO

Essa é uma mensagem opcional que permite a identificação da conectividade local. A detecção dos vizinhos pode ser feita tanto com mensagens de nível dois quanto utilizando o RREP com $TTL = 1$ como HELLO. Nesse RREP, o IP de destino é preenchido com o IP do emissor e o número de seqüência atual do nó também é inserido no pacote.

Esse tipo de reconhecimento pode ser utilizado para determinar se os enlaces com os vizinhos ainda estão válidos. Se após um período pré-determinado para essa função o nó não receber nenhum HELLO, ele pode considerar aquele enlace como perdido e as rotas

que o utilizam como inválidas.

II.2.4 Encaminhamento de Pacotes

O AODV obtém suas rotas pelo processo já descrito, utilizando as mensagens RREQ e RREP. Após esse processo, ele armazena em sua tabela de roteamento o nó de destino e o próximo salto para o qual deve encaminhar a mensagem. Uma vez que nenhum nó possui o mapa da topologia completo, é necessário que as informações guardadas na tabela estejam sempre o mais atualizadas possível em todos os nós. Esse controle é feito pelos números de seqüência e evita que ocorram *loops* na rede.

Capítulo III

Principais Formas de Ataques e Algumas Soluções

Os ataques a redes ad hoc sem fio se servem de vulnerabilidades provindas das principais características destas redes: a ausência de uma infra-estrutura e de centralização, a utilização do ar como meio físico e o roteamento cooperativo. A falta de infra-estrutura e de centralização dificulta enormemente a utilização de mecanismos eficientes e convencionalmente utilizados para prover autenticação, autorização, confidencialidade e integridade. A transmissão das informações pelo ar através de ondas de rádio-freqüência facilita a escuta, as interferências e a injeção de pacotes falsos, o que é mais difícil de ser realizado em meios cabeados. O roteamento colaborativo é responsável por diversos ataques específicos de redes ad hoc.

Os ataques a redes ad hoc móveis podem ser divididos em passivos ou ativos [6]. Os ataques passivos não afetam a operação da rede, sendo caracterizados pela espionagem dos dados sem alterá-los. Por outro lado, os ataques ativos são aqueles em que o atacante cria, altera, descarta ou inviabiliza o uso dos dados em trânsito. Os ataques ativos são os mais numerosos, podendo atuar em diferentes camadas do modelo OSI. Outra classificação pode ser feita com relação aos atacantes, que podem ser classificados como internos ou externos. Atacantes internos são aqueles que conseguem de alguma forma se passar por membros

da rede, enquanto que os externos são aqueles que influenciam, mas não participam da rede. De fato, a eficiência e as possibilidades de ataques variam de acordo com o acesso que o atacante tem à rede. Se de alguma forma o atacante conseguir obter chaves ou for incluído na lista de vizinhos válidos, passando a ser um atacante interno, pode causar mais problemas, pois pode emitir mensagens de controle. Neste projeto, abordam-se apenas os ataques no nível de roteamento.

III.1 Ataques Passivos

Nos ataques passivos, o atacante não interfere no funcionamento da rede, mas escuta mensagens e analisa o tráfego. O atacante tem acesso às informações contidas nas mensagens, porém não as altera ou destrói. Os ataques passivos são de difícil detecção por não influírem no comportamento da rede.

Escuta Clandestina ou Espionagem (*Eavesdropping*[1])

A Escuta Clandestina, ou Espionagem, caracteriza-se pela escuta de mensagens que trafegam na rede. É considerado um ataque passivo porque o atacante não participa da topologia da rede e não modifica os dados, aproveitando-se apenas do meio inseguro para roubar informações.

Quando o atacante utiliza o tráfego observado para aprender a localização dos recursos críticos da rede, o ataque é chamado de Revelação de Informações Críticas (*Homing/Information Disclosure*) [8]. Uma vez que esses pontos críticos são encontrados, essas informações são passadas para outros nós maliciosos que podem realizar ataques ativos. Protocolos de roteamento que utilizam encaminhamento geográfico são ainda mais expostos a esse ataque, pois a posição exata dos nós críticos é passada para os atacantes ativos, facilitando a localização e ataque ao nó.

A criptografia para prover sigilo das informações é muito utilizada nas camadas de

[1] A origem do termo *Eavesdropping* vem da expressão *hide out in the eavesdrop of a house* [7], que tem como similar em português o ouvir atrás da porta.

protocolos mais altas, pois acarreta custo computacional apenas nas extremidades da comunicação, na máquina do usuário. A proteção contra espionagem nas camadas baixas, como na camada de rede, não costuma ser utilizada, pois requer operações de criptografia/decriptografia nó a nó para cada mensagem, o que gera um custo computacional muito alto.

III.2 Ataques Ativos

Os ataques ativos, em sua maior parte, têm como alvo a vulnerabilidade de alguma camada específica do modelo OSI, e aqui serão tratados apenas os ataques à camada de roteamento.

III.2.1 Camada Rede

Devido tanto às características críticas da rede, quanto às vulnerabilidades dos protocolos de roteamento, é na camada rede que ocorrem a maior parte dos ataques a redes ad hoc. Muitos dos ataques relativos a essa camada possuem soluções eficientes, com métodos preventivos capazes de reduzir bastante a interferência do atacante na rede.

Alteração de Mensagens de Roteamento ou Ataque Bizantino

O nome Ataque Bizantino tem como referência a história de como os generais bizantinos organizavam suas guerras. Uma vez que o exército era pequeno, os generais sempre se comunicavam dois a dois oralmente. Cada um deveria se posicionar em volta da cidade e observar os acontecimentos. Através da troca de informações, todos deveriam optar por atacar ou bater em retirada [9]. A existência de generais traidores, que omitiam ou passavam informações falsas poderia levar a uma tomada de atitude errada pelos outros generais. O paralelo do ataque Bizantino para ataques da camada de rede é o fato dos nós sempre confiarem na informação passada pelos outros nós e, caso existam nós maliciosos enviando informações falsas, a rede acaba tomando decisões de roteamento incorretas,

que podem ir desde a utilização de caminhos mais longos até contagens para o infinito.

Para gerar problemas como *loops* de roteamento, pacotes de roteamento falsos, escolha de caminhos não-ótimos, entre outros, neste ataque, um nó sozinho ou diversos nós em conluio utilizam maliciosamente as mensagens de controle dos protocolos em uso. Murthy [6] cita esse problema restringindo-o apenas aos problemas de roteamento, embora esse tipo de abordagem maliciosa possa atingir diversas camadas. O Ataque Bizantino é de difícil detecção, pois para os nós não-maliciosos, o funcionamento está correto, embora, de fato, esteja apresentando anomalias do tipo pacotes falsos, alterados e descartados.

O Ataque Bizantino pode ser efetuado de diferentes formas nos protocolos OLSR e AODV, causando problemas de roteamento como a escolha de uma rota falsa ou mais longa. Um exemplo é o caso da topologia da Figura III.1 com o AODV. De acordo com o exemplo da Figura II.5, a melhor rota entre F e D é a F-1-8-4-D. Caso o nó 5 seja malicioso, ele pode enviar as suas mensagens de RREQ e RREP modificando o número de saltos, de forma a transformar a sua rota mais atrativa, como pode ser visto na Figura III.2, na qual as setas cheias representam os RREQs aceitos e as pontilhadas os RREQs descartados. Nesta figura, F inicia um processo de descoberta de rota, enviando um RREQ para seus vizinhos. O nó 1 escuta a mensagem e a encaminha, processo que se segue de acordo com a ordem que os nós têm acesso ao meio. Neste exemplo, a ordem suposta foi F-1-2-7-8-3-6-5-4. O nó 5, que é o nó malicioso, ao receber de 6 um anúncio de três saltos, ao invés de repassar quatro saltos, repassa um salto, levando o destino a crer que a rota possui apenas dois saltos. Desta forma, quando o RREQ do nó 4 chega anunciando uma rota de quatro saltos, que seria a melhor opção para essa rede, o destino a descarta, pois já possui uma rota com dois saltos. Assim, o nó 5 conseguiu realizar o seu objetivo de causar a escolha de uma rota não-ótima e ainda atrair o tráfego para si. Outro ataque é aumentar o número de seqüência do RREP, de forma a obrigar o nó de origem a descartar todas as mensagens não-maliciosas que cheguem em resposta ao seu RREQ, pois elas parecerão desatualizadas. Dependendo de quanto se aumente o número de seqüência, mesmo respostas posteriores a pedidos de rota seriam descartadas, até que o nó destino alcançasse aquele número de seqüência. Tal processo está exemplificado na

Figura III.3, na qual as setas cheias representam os RREPs aceitos e as pontilhadas os RREPs descartados. Nesta, o nó 5 continua sendo o nó malicioso, embora não tenha realizado infrações durante o encaminhamento do RREQ. Supondo que o RREQ advindo do nó 5 tenha chegado antes do RREQ do 4, o destino poderia emitir um RREP para o 5 antes de receber o RREQ do 4. Ao receber o RREP, o nó 5 o repassa alterando o número de seqüência de D de 2 para 10, como mostrado na Figura III.3b. Em seguida, os nós encaminham o RREP modificado até o nó F. Uma vez que a melhor rota é D-4-8-1-F, um novo RREP será emitido por D, comunicando F que esta é a melhor escolha. No entanto, como o número de seqüência de D no RREP não-alterado é 2, o nó F descarta a mensagem, pois acredita que a informação está desatualizada.

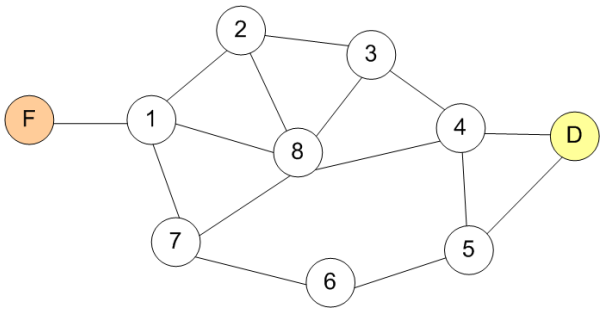

Figura III.1: Exemplo de topologia, onde o nó F representa a fonte e o nó D, o destino.

No OLSR o Ataque Bizantino também pode ser realizado de muitas formas. Como exemplo, temos a emissão de HELLOs falsos, de forma a criar um enlace de *Multipoint Relay* falso. Esse ataque está exemplificado na Figura III.4. Na Figura III.4a, o nó 1 e 4 enviam seus HELLOs. Em b, o nó 8 modifica os Hellos e os envia como se fosse 1 e 4. Em c, os nós 1 e 4 acreditam ser vizinhos e criam um enlace MPR entre eles falso, o que prejudica todo o roteamento após o cálculo das rotas.

As soluções para o ataque bizantino podem ser a assinatura digital, o uso de múltiplos caminhos e, ainda, a autorização, mecanismos que são descritos na Seção III.3.

Estouro (*Overflow*) da Tabela de Roteamento

Este ataque se baseia no fato de os protocolos de roteamento ad hoc pró-ativos armazenarem todas as rotas anunciadas pelos seus vizinhos. Nestes protocolos, o nó armazena

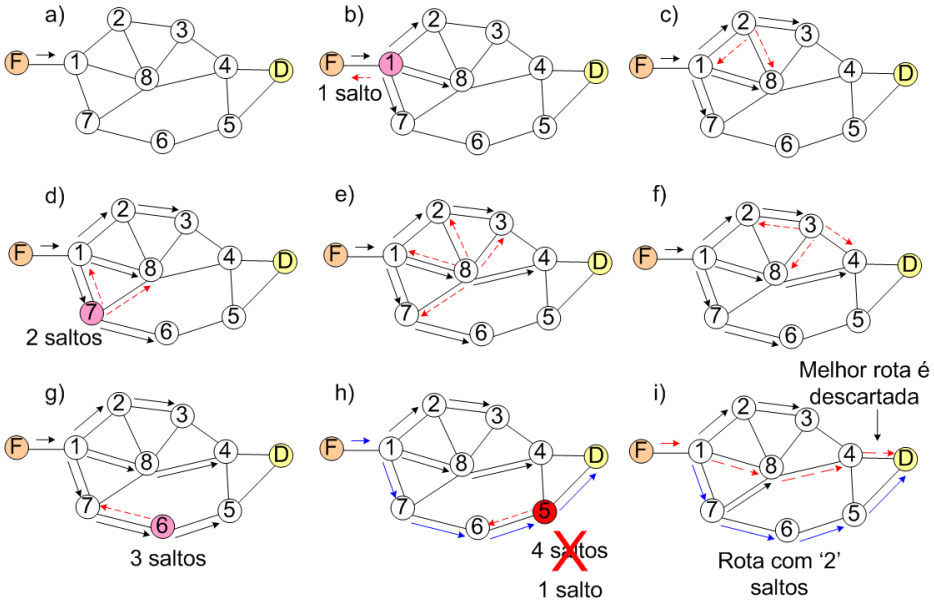

Figura III.2: Exemplo de ataque no qual nó malicioso troca o número de saltos durante o processo de descoberta de rotas.

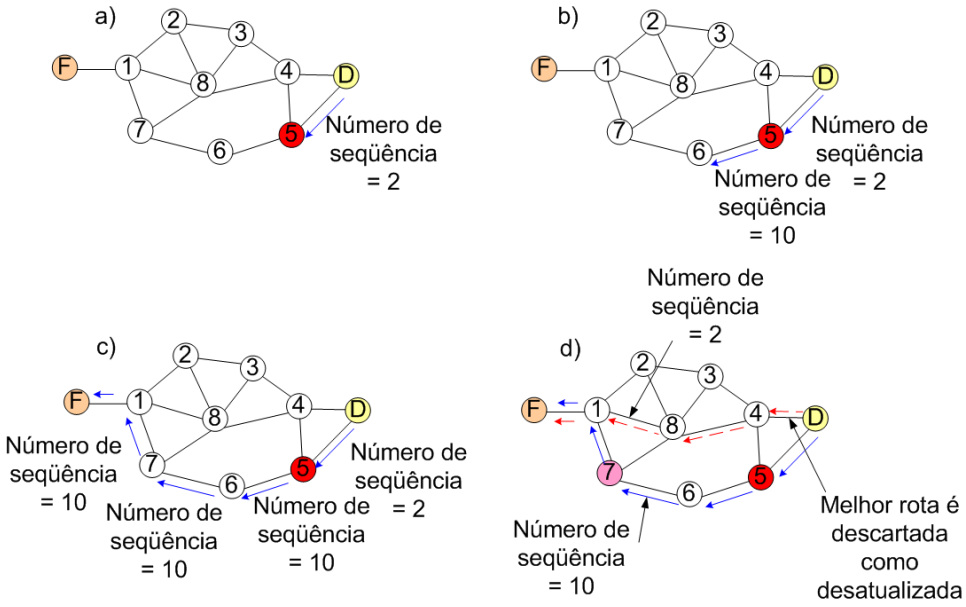

Figura III.3: Exemplo de ataque no qual nó malicioso troca o número de seqüência do destino durante o processo de descoberta de rotas.

em sua tabela de roteamento todas as mensagens de rota que recebe periodicamente, como acontece com o OLSR com suas mensagens de HELLO. A estratégia deste ataque

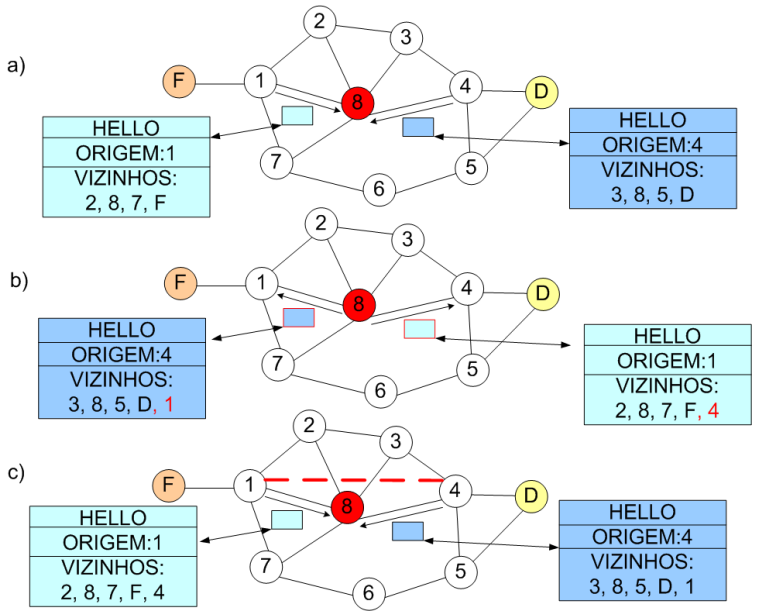

Figura III.4: Exemplo de ataque bizantino no OLSR.

é anunciar diversas rotas para nós inexistentes, de modo a aumentar progressivamente o tamanho da tabela de roteamento, até que ela estoure e o nó não possa mais armazenar as rotas reais. Os protocolos reativos que armazenam diversas rotas para um mesmo destino também estão expostos a esse tipo de ataque, pois o nó malicioso poderia enviar rotas passando pelos nós inexistentes.

Esse ataque é grave no caso de redes ad hoc que possuem nós com recursos escassos, onde tanto o gasto de energia com a recepção de um número excessivo de mensagens, quanto o estouro de *buffer* são cruciais. Para prevení-lo, deve-se limitar o número máximo de rotas nas tabelas de roteamento, além de só aceitar entradas de nós autenticados.

Envenenamento de *Cache*

Muito semelhante ao ataque do estouro da tabela de roteamento, este ataque visa envenenar o *cache* de roteamento fazendo anúncios falsos de rotas para nós reais. Esse ataque se aproveita, em especial, de protocolos reativos, como o *Ad Hoc On demand Distance Vector* (AODV) [3], que estão mais susceptíveis que os protocolos pró-ativos pelo fato de anunciarem para onde desejam mandar o pacote sempre que não possuem rota para o destino, permitindo ao nó invasor anunciar a rota falsa antes de um nó

confiável. No caso dos protocolos pró-ativos, esse ataque também é possível, mas seria necessário mandar anúncios de rota falsos em todas as rodadas de atualização, para todos os possíveis destinos.

É importante ressaltar que a identificação deste tipo de ataque é complicada, principalmente devido às características de mobilidade da rede.

Não há diferença entre um anúncio de rota falso e de uma rota que deixou de existir porque algum nó intermediário mudou de posição. Ainda assim, para evitar esse tipo de comportamento dos nós maliciosos, deve-se utilizar sistemas de confiabilidade baseados em monitoramento e punição. Outra forma de identificação deste ataque é a utilização de pacotes de investigação, como será visto na Seção III.3.

Replicação de Pacotes

Este ataque possui dois objetivos principais: ocupar o meio de transmissão e levar os nós à exaustão. Para conseguir seus objetivos, o atacante envia réplicas de pacotes de roteamento antigos, impedindo a transmissão de outros nós nos momentos em que está enviando a réplica, além de gastar a bateria dos demais nós, que deverão escutar o pacote replicado e processar o seu conteúdo.

As soluções propostas para esse tipo de ataque são limitadas, pois ainda que se deduza que o mesmo nó envia mensagens de roteamento antigas, através da observação do número de seqüência, o máximo que poderia ser feito é retirar aquele nó das rotas, mas nada poderia impedí-lo de continuar a enviar as suas réplicas, assim como acontece no ataque da interferência na camada física.

Direcionamento Falso (*Misdirection*)

O direcionamento falso consiste na fabricação de mensagens visando gerar negação de serviço para um determinado nó. Assim, são enviadas mensagens de modo a direcionar tráfego para uma determinada região que se deseja atacar. Na versão da Internet desse ataque, conhecida como Ataque *Smurf*, o atacante forja pacotes *echo*, colocando como emissor o nó vítima, que irá receber inúmeros *echo-replies*.

Esse ataque pode ser realizado por mecanismos além do uso de *echos*, em especial no caso de protocolos reativos, como o AODV. Neste caso, o atacante emite RREPs falsos, colocando como origem algum outro nó. Esse ataque se caracteriza como um tipo de Identidade Falsa, que tem como objetivo fazer com que um determinado nó receba pacotes que ele não é capaz de rotear, com o fim de ocupar o meio e forçar o nó a processar os pacotes e a gerar de mensagens de erro. A outra conseqüência é que os dados não poderão ser entregues corretamente e é necessário um novo processo de busca por rotas.

Inundação de *HELLOs*

Este ataque inicialmente foi considerado para redes de sensores. De fato, ele também se aplica às redes ad hoc, desde que o atacante possua uma potência de transmissão maior que os demais nós da rede. A Inundação de *HELLOs* só se aplica a protocolos que utilizam a mensagem de *HELLO* para identificação dos vizinhos, embora não façam a verificação de bidirecionalidade do enlace, o que é o caso do AODV.

Para realizar este ataque, o nó malicioso envia *HELLOs* com alta potência, informando que o nó possui enlaces com determinados destinos. Assim, ele atinge um grande número de nós, que por terem ouvido a mensagem, o colocam na sua lista de vizinhos e podem escolhê-lo para encaminhamento de dados. No entanto, apesar dos nós ouvirem o nó malicioso, o nó malicioso não é capaz de escutá-los, de forma que vários nós da rede vão apontar suas rotas de encaminhamento para um nó inalcançável.

A inundação de *HELLO* é mais eficiente ainda quando realizada em conjunto com uma revelação de informações críticas, de forma a permitir que o nó malicioso descubra quais rotas ele deve tentar interceptar.

Esse ataque pode ser evitado pela verificação de bidirecionalidade do enlace. Cabe observar que apesar de ser uma solução simples, muitos protocolos de roteamento não a aplicam, assumindo que os enlaces são bidirecionais.

Ganância (*Greed*)

A Ganância é caracterizada quando um nó dá uma prioridade injusta às suas mensa-

gens. O nome ganância é justificado pelo fato do nó atrapalhar o funcionamento da rede por tentar ter sempre a maior fatia de tempo para transmissão, embora não prejudique o funcionamento da rede de forma explícita.

Esse ataque se aplica melhor no caso de dados, onde geralmente o nó deseja dar prioridade às suas próprias mensagens. No caso do roteamento, ele se aplica a protocolos sob demanda, pois para os nós gananciosos, os seus pedidos por rotas e de anúncio de existência (HELLO) terão sempre prioridade sobre o encaminhamento de pacotes de qualquer outro nó.

A detecção de um nó ganancioso é difícil, pois um nó que já esteja com falhas de bateria apresenta um comportamento semelhante. Assim, a melhor forma de evitar esse ataque é a utilização de redundâncias, garantindo que, ainda que pacotes sejam perdidos por demora ao enviar, uma segunda rota possa garantir a sua entrega.

Encaminhamento Seletivo ou Buraco Cinza

Uma das características principais das redes ad hoc é a confiança nos vizinhos para o encaminhamento de dados. No entanto, um vizinho malicioso pode encaminhar apenas alguns pacotes. No encaminhamento seletivo, um nó malicioso pode decidir não passar algumas ou todas as mensagens para um determinado nó alvo, ou pode optar por passar as mensagens de roteamento, mas impedir a transmissão de dados, impedindo o funcionamento da aplicação.

Um tipo especial de encaminhamento seletivo é chamado de egoísmo, no qual o nó não encaminha nenhuma mensagem dos vizinhos, passando apenas as suas próprias. O egoísmo nem sempre é classificado como um ataque, podendo ser uma escolha de um nó por um comportamento não cooperativo. Tal decisão pode ser tomada, por exemplo, em momentos nos quais o nó deseja se poupar.

O encaminhamento seletivo pode ser grave tanto nos protocolos pró-ativos quanto nos reativos. No protocolo OLSR, caso um nó tenha em seu conjunto de vizinhos apenas um nó, que é malicioso, ele ficará isolado da rede, pois o nó malicioso não encaminhará as mensagens de TC (*Topology Control*) do nó alvo e nem anunciará a existência dele em

suas próprias TCs. Este caso está representado na Figura III.5. Nesta, o nó malicioso 1 isola o nó alvo F e os demais nós que dependem dele para ter conexão. Para isso, ele não encaminha a mensagem TC do nó alvo F, como mostrado na Figura III.5a e não anuncia que o nó alvo o escolheu como MPR, como representado na Figura III.5b. No caso do AODV, o nó malicioso pode não encaminhar os RREP, impedindo a formação das rotas.

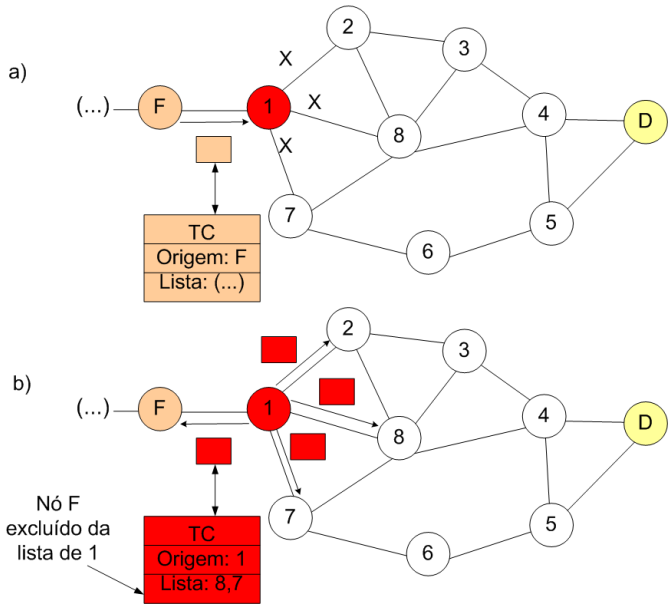

Figura III.5: Representação de encaminhamento seletivo no OLSR.

O Buraco Cinza é de difícil detecção, assim como a Ganância, pois nós com pouca energia e perdas de pacotes normais podem gerar uma situação muito semelhante.

Buraco Negro

Este é caso extremo do Encaminhamento Seletivo, onde todos os pacotes são atraídos para o nó malicioso e são descartados. Este ataque, dependendo da posição do atacante, pode ter um efeito totalmente destrutivo na rede, impedindo o seu funcionamento. Ao contrário do Encaminhamento Seletivo, a detecção do Buraco Negro é fácil, pois em pouco tempo toda uma parte da rede deixa de funcionar.

Uma segunda conseqüência do buraco negro é que, como ele atrai muito tráfego em sua direção, ele acaba consumindo os recursos dos nós à sua volta, tanto em termos do meio, que fica excessivamente ocupado, assim como em termos de recursos dos nós. No caso de exaustão desses nós, o resultado é o particionamento da rede, mesmo que o nó atacante não descarte nenhum pacote.

Assim como o Encaminhamento Seletivo, uma premissa de funcionamento do ataque é que o nó malicioso se torne atrativo durante a escolha de rotas. Para tanto, vários métodos podem ser utilizados, como o ataque da pressa e o túnel de minhoca.

A solução mais simples para os ataques de descarte de pacotes é a utilização de múltiplas rotas. Outros métodos que ajudam a detectar e prevenir esse comportamento são a investigação e a autorização, que são descritos na Seção III.3.

Ataque da Pressa (*Rushing Attack*)

Este ataque permite a formação de um buraco negro, se aplicando a protocolos de roteamento sob demanda que guardam apenas uma rota para cada destino em sua tabela. Ao receber um *Route Request* (RREQ), o atacante 'se apressa' em responder esta solicitação de rota de forma mais rápida que os demais nós da rede. Assim, para os protocolos de roteamento que apenas consideram a resposta que chegou primeiro, a resposta de rota enviada pelo nó malicioso é a que será considerada e as demais respostas provenientes dos outros vizinhos serão descartadas. Desta forma, as rotas sempre passariam pelo nó malicioso, tornando a rede vulnerável.

A detecção de tal ataque é difícil dado que, para o protocolo de roteamento, tudo está transcorrendo de forma normal. Uma solução é tentar identificar os métodos para conseguir enviar a mensagem de forma mais rápida, como por exemplo, um abuso do protocolo de enlace. De fato, existem vários métodos para acelerar o envio da mensagem que não exigem muitos recursos do nó malicioso [10]. Em geral, os protocolos MAC (*Medium Access Control*) impõem atrasos entre o momento no qual o pacote é recebido e a transmissão. Um exemplo são os protocolos MAC que utilizam o *Time Division Multiple Access* (TDMA) onde o nó precisa esperar até a sua vez de transmitir, ou ainda

os protocolos de acesso ao meio que utilizam *Carrier-Sense Multiple Access* (CSMA), onde é utilizado um *backoff* para evitar colisões. Outro espaço de tempo que também pode ser burlado por um nó malicioso é aquele que pode ser usado pelo roteamento entre a recepção de um RREQ e o encaminhamento do mesmo, para evitar colisões. Assim, um atacante que deseja enviar um pacote mais rápido que outros nós pode simplesmente ignorar um ou mais destes tempos de espera.

Uma outra forma de executar esse ataque é, por exemplo, provocar filas nas interfaces dos nós vizinhos, de forma a que o nó malicioso repasse o pacote enquanto seus vizinhos estão processando os pacotes das filas. Esse tipo de atitude do nó malicioso é mais fácil em sistemas que utilizam autenticação da mensagem, pois ele pode gerar várias mensagens com assinaturas falsas, levando os vizinhos a perder tempo durante a verificação das assinaturas. No caso de autenticação por chave pública esse problema é ainda maior, devido ao alto custo computacional para realizar a verificação. Outros métodos para transmitir os pacotes mais rápido que os vizinhos também são possíveis, como a utilização de uma potência de transmissão maior, reduzindo a quantidade de saltos atravessada.

A melhor solução para este ataque é o uso de múltiplas rotas, que garantiriam que mesmo que o atacante atraísse o trafego para si em uma das rotas as outras permaneceriam seguras.

Túnel de Minhoca (*Wormholes*)

No ataque do túnel de minhoca, dois atacantes criam um túnel de comunicação por um enlace de baixa latência, através do qual irão trocar informações da rede, replicando-as do outro lado do túnel, de forma a tornar excepcionalmente atrativo o enlace formado pelos dois. Assim, os nós maliciosos informam outros nós da rede que eles podem se comunicar com determinados destinos por apenas um salto, ao invés de utilizar os vários saltos que existem realmente entre o nó que deseja se comunicar e o nó destino.

O túnel é um canal seguro e de baixa latência entre os dois nós maliciosos, que permite que os nós vizinhos sejam incapazes de perceber que o ataque está sendo realizado. Uma forma simples de obter esse resultado é utilizar uma conexão cabeada direta entre os dois

nós, fazendo uma transmissão mais rápida que o encaminhamento por múltiplos saltos [10]. Outra possibilidade é a utilização de um enlace direcional sem fio de longa distância, através do uso de antenas diretivas, para conseguir poupar saltos e, com isso, obter uma velocidade superior a de comunicações que normalmente utilizariam mais que um salto [11]. Uma terceira forma seria utilizar um canal diferente do utilizado na comunicação com uma potência de transmissão superior, o que também permitiria a maior velocidade sem que os vizinhos notassem. Além disso, uma técnica que pode ser usada é o envio dos bits diretamente, sem aguardar a chegada do pacote completo para começar a transmissão.

É interessante notar que, no caso de o atacante construir o seu túnel de forma honesta e confiável, nenhum prejuízo direto é causado à rede. Pelo contrário, um serviço é prestado ao melhorar a eficiência da conexão da rede. No entanto, o ataque coloca os nós maliciosos em uma situação privilegiada, que os permite gerar, no momento que desejarem, diversos tipos de prejuízos à rede.

É importante observar que, para as camadas superiores, o ataque é invisível, e mesmo para a camada de roteamento é complicado perceber a presença de um Túnel de Minhoca.

Negação de serviço

O conceito de negação de serviço é muito amplo. O ataque de negação de serviço pode ser definido como qualquer ação que reduza ou elimine a capacidade da rede de realizar uma de suas funções esperadas [8]. Assim sendo, a negação de serviço não seria causada apenas por ataques, mas por qualquer evento que prejudicasse a rede, como falhas de *hardware*, defeitos de programas, exaustão de recursos intencional ou não, condições ambientais não favoráveis ou qualquer combinação desses fatores. Segundo esta definição, todos os ataques ativos podem ser classificados como negação de serviço da rede.

Uma forma mais severa deste ataque é a negação de serviço distribuída. Nesta, vários adversários estão espalhados pela rede fazendo um conluio para impedir que usuários legítimos tenham acesso aos serviços. Este ataque tem um efeito muito mais rápido sobre a rede, podendo impedir totalmente o seu funcionamento sem grandes dificuldades.

Sybil[2]

O ataque Sybil se baseia no fato de que é praticamente impossível, em sistemas computacionais distribuídos, que nós que não se conhecem apresentem identidades distintas convincentes. Sem a existência de um ponto central para controlar a associação de uma identidade a uma entidade, é sempre possível para uma entidade desconhecida apresentar múltiplas identidades. Assim, o ataque Sybil acontece quando um único *hardware* assume múltiplas identidades em uma rede [13].

Este ataque tem grande importância devido a muitas arquiteturas utilizarem réplicas de bancos de dados para impedir a violação da integridade dos dados e fragmentação de tarefas para impedir a violação da privacidade das atividades realizadas. Se algum nó malicioso assume múltiplas personalidades, o sistema pode escolher o mesmo nó para guardar todas as réplicas ou fragmentos, o que acabaria com toda a segurança adquirida com o mecanismo.

O Sybil pode ser utilizado para atacar não só sistemas de armazenamento distribuído. Uma outra possibilidade que utiliza redundância é o roteamento com múltiplos percursos. Em geral, protocolos que utilizam essa técnica buscam escolher caminhos disjuntos ou trançados para diminuir a possibilidade de existir um atacante na rota. O ataque Sybil pode ser feito de tal forma a colocar uma identidade falsa em cada rota, de forma que todos os caminhos continuem passando pelo nó malicioso. Ainda no campo de roteamento, outro possível problema que não tem relação com redundância é o ataque ao roteamento geográfico. Neste caso, o nó malicioso anunciará sempre uma de suas identidades Sybil como o nó mais próximo ao destino, fazendo com que todos os pacotes de roteamento passem por ele.

Outro ataque possível é a utilização dos nós Sybils para falsificar resultados de votações na rede. Sempre que existir algum mecanismo cooperativo para tomada de decisões na

[2] A primeira descrição do ataque Sybil foi feita em [12], para redes *peer-to-peer*. O nome do ataque foi inspirado em um caso famoso nos EUA, onde uma mulher sofria de múltiplas personalidades, num total de 16 diferentes personalidades. Sybil Isabel Dorsett foi o pseudônimo criado pela autora Flora Schreiber para proteger a identidade real da paciente, em seu livro "Sybil", Warner Books, 1973.

rede, o nó malicioso pode gerar diversas identidades para votar sempre a seu favor. Outro ataque é a alocação injusta de recursos, que pode ocorrer em redes que fazem divisão temporal para acesso ao meio. Neste caso, o nó malicioso utiliza todas as suas identidades falsas para obter um maior tempo de acesso. Por fim, uma outra utilização para os nós Sybils acontece em redes que utilizam mecanismos de confiabilidade. Em tais redes, a índole do nó é dada pela observação de suas ações. Um nó só é considerado malicioso se cometer diversas ações consideradas ruins ou se cometer uma grande ação ruim. Assim, duas estratégias podem ser utilizadas. A primeira seria o espalhamento da culpa, na qual o nó Sybil utiliza cada uma de suas identidades para fazer pequenas ações ruins, de forma que nenhuma delas possa ser considerada maliciosa. A outra estratégia seria utilizar uma identidade para realizar uma ou mais ações ruins até que ela fosse expulsa, classificada como maliciosa. Quando isso acontecesse o nó geraria uma nova identidade e a usaria para continuar atacando.

Existem diversas propostas de defesas para o Sybil, sendo a maioria delas baseadas em métodos de autenticação ou de validação de chaves distribuídas. Estes são descritos no Capítulo IV.

Identidade Falsa (*Impersonating*) e Ataque da Replicação

Estes ataques se assemelham muito ao Sybil. Nestes, nós maliciosos assumem uma ou mais identidades da rede, porém desta vez, todas as identidades são reais, e cada identidade está ligada a um ou mais hardwares diferentes, caracterizando respectivamente a Identidade Falsa e o Ataque da Replicação. A Replicação serve para inserir vários nós maliciosos, sem ter a dificuldade de se roubar várias identidades. Desta forma, os nós maliciosos replicam alguma identidade roubada e utilizam as réplicas simultaneamente dentro da rede [14]. No caso dessas réplicas serem muito numerosas, os adversários podem dominar a rede através de um conluio para ter vantagens em casos votação, ou ainda tirar vantagem apenas por estarem participando da rede. Deve-se notar que esses ataques que, em geral, acontecem após uma violação ou quebra de algoritmo criptográfico, fazem com que o atacante tenha o segredo da rede, podendo participar de todas as suas atividades como um nó legítimo. Assim, ele pode, por ter se tornado um atacante interno, executar

a maioria dos ataques já descritos, com a facilidade do conluio com as outras réplicas.

Cabe ressaltar, que apesar da gravidade do efeito causado, a Replicação é de fácil detecção, devido a uma mesma identidade se anunciar em diversos pontos da rede. Algumas propostas eficientes já foram feitas, embora elas não estejam incluídas nos protocolos mais populares. No caso da identidade falsa, se o nó legítimo tiver sido destruído, a detecção é muito mais complicada, pois a identidade é única na rede.

Uma outra variação da Identidade Falsa é o Ataque do Homem no Meio (*Man-in-the-Middle*) [6]. Neste, o nó malicioso intercepta uma comunicação, enganando os dois nós que deveriam se comunicar, como pode ser visto na Figura III.6. Uma vez que ele se passa por x para y e por y para x, ele está assumindo duas identidades reais da rede. Esse tipo de ataque só pode ocorrer em redes que não possuem um terceiro ponto para autenticar a comunicação entre os dois primeiros.

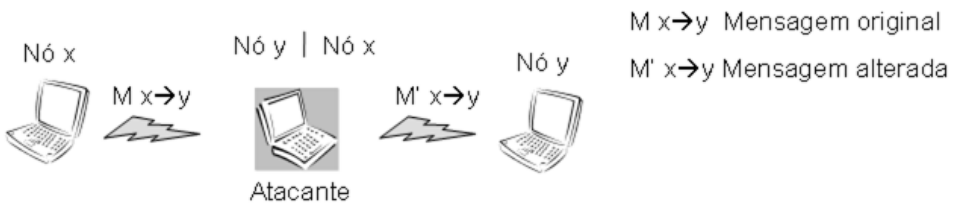

Figura III.6: Ataque do Homem no Meio.

Violação

A violação (*Device Tampering*) é um dos ataques mais preocupantes para redes onde os nós ficam desprotegidos. Ela consiste da violação física dos nós com o fim de obter informações e segredos, além de também comprometer o nó, com a inserção de códigos maliciosos ou ainda pela troca de partes do *hardware*. A maioria dos protocolos desenvolvidos para prover segurança falha em ambientes nos quais é possível ocorrer a violação. De fato, não é simples garantir a segurança de todos os nós quando se trata de redes de larga escala. Em especial pelo fato de que, em geral, existem muitas falhas de comunicação e períodos de sono, que tornam impraticável distinguir uma falha de um nó propositalmente desligado ou destruído. A proposta de defesa para esse ataque é a resistência à violação,

que diz respeito à dificuldade imposta por um mecanismo de segurança à violação da informação, seja por software ou por hardware. O conceito pode ser estendido à capacidade da rede em resistir a ataques onde um usuário não autorizado se apossou de um dos nós da rede.

Desta forma, as técnicas de resistência à violação podem ser baseadas em hardware, em software, ou em ambos. Uma forma de aumentar a resistência à violação baseada em software é a utilização de associações de segurança temporárias. Por exemplo, toda vez que um nó for enviar uma requisição de rota em um protocolo de roteamento reativo, o sistema operacional pode exigir que o usuário entre com uma senha, ou forneça sua impressão digital para se autenticar. Se o equipamento for perdido ou roubado, o usuário não autorizado não conseguirá gerar pedidos de rota, e quando esta tentativa for feita, o sistema operacional do nó pode tomar alguma atitude adicional, como, por exemplo destruir qualquer chave armazenada no sistema, aumentando sua resistência à violação [15]. Esse tipo de medida tem o problema de ser considerada como inconveniente pela maioria dos usuários.

A utilização de hardware resistente à violação possui, naturalmente, um problema de custo-benefício. Quanto mais seguro o hardware, mais caro [16]. Além disso, como na segurança do software, não existe sistema de hardware inviolável. Existem alguns mecanismos de redes ad hoc que supõem a utilização de hardware resistente à violação. Um exemplo é o mecanismo de estímulo ao encaminhamento de pacotes, proposto em [17]. A idéia é que os nós de uma rede ad hoc tenham motivos para se comportar de forma cooperativa encaminhando os pacotes uns dos outros, embora isto signifique consumir energia. O sistema se baseia em créditos, representados por uma moeda virtual, o *nuglet*. Os *nuglets* são creditados ou debitados aos nós que encaminham ou enviam seus próprios pacotes, respectivamente. O controle dos *nuglets* de um nó é feito através de um contador implementado em hardware que, obviamente, deve ser resistente à violação.

III.3 Alguns Mecanismos de Segurança

A utilização dos mecanismos de segurança deve levar em consideração a relação entre custos e benefícios de cada solução. Nesta seção são descritos alguns métodos simples para aumentar a segurança que podem ser implementados no projeto de protocolos de roteamento. Entre estes métodos estão a utilização de redundância, a verificação de bidirecionalidade do enlace, a autorização e a investigação.

Redundância

A redundância pode ser utilizada para prevenir diversos ataques em redes ad hoc, devido a sua característica de possibilitar alternativas seguras não comprometidas pelo atacante, dando maior robustez à rede. O único ataque que pode inviabilizar a robustez gerada por redundância é o Sybil.

A utilização de múltiplas rotas é uma forma de se obter redundâncias no encaminhamento de dados. Embora as múltiplas rotas sejam usadas para balanceamento de carga e prevenção contra congestionamentos, elas também podem prover robustez contra ataques, através do envio repetido dos dados pelos múltiplos caminhos. Assim, ainda que existam nós maliciosos em um caminho, descartando ou alterando pacotes, a mensagem real consegue ser entregue através de outra rota que não inclua esse nó. Para tanto, a escolha das rotas redundantes deve ser cuidadosa, escolhendo rotas disjuntas, ou seja, totalmente independentes, de forma a garantir que mesmo que um nó malicioso esteja em uma determinada rota, ele não estará nas outras. O grande problema da utilização de rotas disjuntas é a dificuldade de encontrá-las em redes ad hoc, quando a conectividade é baixa. Para resolver esta questão existem duas propostas. A primeira é, ao invés de buscar rotas disjuntas, buscar os nós que são confiáveis na rede e tentar traçar múltiplas rotas passando por esses nós, independente de elas se sobreporem ou não [18]. A segunda proposta é a de utilizar caminhos trançados [19], que podem ter nós em comum, mas não possuem enlaces em comum, o que pode prover proteção probabilística contra o encaminhamento seletivo [20].

Apesar das vantagens trazidas pelo uso de múltiplos caminhos, é preciso observar também os contrapontos dessa medida. O uso de múltiplos caminhos pode sobrecarregar a rede, pois o tráfego total irá ser multiplicado pelo número de rotas entre pares, o que restringe o uso dessa técnica a redes que não trabalham próximas à saturação.

A utilização de armazenamento distribuído na rede é outro tipo de redundância [13]. Este mecanismo, quando utilizado para prover segurança, consiste em distribuir várias réplicas de uma base de dados por diversos nós, de forma que a falha de um nó não comprometa a disponibilidade das informações.

Verificação de Bidirecionalidade do Enlace

Esta verificação está diretamente ligada à forma como o protocolo de roteamento determina quais são os seus vizinhos. Apesar de freqüentemente o roteamento ser executado acima da camada de enlace IEEE 802.11, nem sempre a informação sobre vizinhos gerada por ele através do ACK de enlace é utilizada. É comum que protocolos de roteamento assumam que todos os enlaces são bidirecionais, o que não é verdade em redes sem fio. Assim, eles determinam que os vizinhos de um nó sejam todos os nós que ele é capaz de ouvir. Isso gera vulnerabilidades que permitem, por exemplo, o ataque da Inundação de HELLOs.

A verificação do estado do enlace é simples e representa baixa sobrecarga ao funcionamento da rede. Basta que todos os nós enviem mensagens periodicamente, anunciando quem são os nós que eles são capazes de escutar. Qualquer nó que escute uma mensagem em que ele esteja listado como um possível vizinho pode considerar o emissor da mensagem como seu vizinho real. Esse tipo de mecanismo é executado nas mensagens de HELLO do OLSR, embora o AODV não o faça nos seus HELLOs.

Autorização

A Autorização [8] é uma defesa contra os ataques do Direcionamento Falso, Encaminhamento Seletivo e Buraco Negro, que se baseia em escolher nós que serão os únicos autorizados a encaminhar informações de roteamento. Para tanto, é necessária a existência de autoridades certificadoras que autentiquem os nós autorizados. Assim, qualquer

pacote de roteamento que chegue por meio de um nó não-autorizado deve ser imediatamente descartado.

Para o bom desempenho da Autorização é necessário um bom sistema de confiança, para que apenas nós que possam ser considerados de boa índole sejam escolhidos para essa função.

Investigação

A Investigação (*Probe*) é um teste que qualquer nó pode realizar para avaliar a conectividade da rede. Em particular em redes onde se tem o conhecimento da estrutura física, é possível enviar pacotes que atravessem o diâmetro da rede, funcionando como sondas. Essas sondas realizam uma investigação sobre a rede, avaliando a existência, ou não, de áreas desconectadas. As sondas não são capazes de distinguir regiões de ataques de regiões de falhas, mas podem determinar as rotas a serem utilizadas.

É importante notar que um pacote de sonda deve ser indistinguível de pacotes normais, para que os nós que descartam pacotes maliciosamente não o encaminhem apenas para reforçar a validade da rota que o inclui. Para um bom desempenho, esses pacotes devem ser enviados periodicamente, tanto para manter um mapa atualizado das rotas disponíveis, quanto para aumentar a probabilidade de descobrir quais são os nós que realizam ataques de encaminhamento seletivo, de egoísmo ou de ganância.

Capítulo IV

Distribuição de Chaves

O principal objetivo do gerenciamento de chaves é compartilhar e manter a validade das chaves através de trocas periódicas com um grupo de participantes ou ainda prover meios para a autenticação de usuários. Em geral, esse tipo de mecanismo é necessário nos protocolos de roteamento seguros, embora não faça parte integrante do protocolo em si. Assim, por ser considerado em separado, é necessário que se desenvolva e utilize módulos para realizar essa função.

A autenticação é complexa em redes ad hoc, porque se assume que a rede não tem nenhum tipo de infra-estrutura. No entanto, quando se deseja dar acesso à Internet e controlar o acesso de usuários, é necessária a inserção de alguns pontos de controle na rede, como um ou mais *gateways*, além do servidor de autenticação.

Nessa seção serão avaliadas algumas das propostas feitas para autenticação em redes ad hoc, considerando-se a sobrecarga que elas impõem à rede e o quanto elas exigem de intervenção humana, indesejável em redes ad hoc. Primeiramente serão expostos alguns fundamentos de segurança em rede, conceitos de criptografia e de infra-estrutura de chaves públicas. Em seguida, são expostas as propostas para redes ad hoc.

IV.1 Fundamentos de Segurança em Redes

Para o projeto de protocolos seguros, é necessário definir os objetivos que os mecanismos de segurança a serem implementados na rede devem buscar. Os requisitos de segurança clássicos que devem ser observados são a autenticação, a confidencialidade, a integridade, o não-repúdio e a disponibilidade. A autenticação garante que uma dada entidade é realmente quem ela diz ser, enquanto que o não-repúdio impede que o emissor de uma mensagem negue a sua autoria. A confidencialidade garante o sigilo das informações trocadas por dois nós e a integridade permite afirmar que as informações recebidas por um nó não foram alteradas durante o trânsito ao longo da rede. A disponibilidade trata de garantir que os recursos da rede estarão disponíveis quando forem necessários.

A criptografia é uma ferramenta fundamental para prover segurança, pois por meio dela, é possível atender a maioria dos requisitos clássicos, exceto a disponibilidade. A maioria dos ataques a redes poderia ser solucionada pela utilização de um mecanismo criptográfico.

IV.1.1 Criptografia

Tradicionalmente, a criptografia é separada em dois ramos: simétrica e assimétrica. A criptografia simétrica é caracterizada pela existência de um segredo, chamado de chave secreta, compartilhado entre os nós que desejam se comunicar. Esta chave é utilizada em operações que alteram os dados a transportar, enviando um texto criptografado ao invés de um texto em aberto. As principais operações realizadas pelos algoritmos simétricos são o ou-exclusivo, a troca de colunas, a troca de linhas, a permutação, a rotação e a expansão, que são operações de baixo custo computacional. Apesar de serem simples, as combinações dessas operações devem ser capazes de tornar difícil a descoberta da mensagem para quem não possui a chave secreta. Por essa razão, a eficiência desses algoritmos é medida pelo seu custo computacional e pela capacidade de modificar a saída dada uma pequena mudança na entrada. Os algoritmos simétricos mais conhecidos são o

Data Encryption Standard(DES) [21] e o *Advanced Encryption Standard*(AES) [22].

Na criptografia assimétrica existem duas chaves, a chave pública e privada. A chave pública deve ser distribuída aos membros da rede, enquanto que a privada deve ser mantida em segredo pelo nó. Esse tipo de criptografia possui maior custo computacional que a simétrica, por fazer uso de operações como a exponenciação. O objetivo principal é que, a partir de uma das chaves, não seja possível encontrar a outra, o que é obtido quando se usa para o cálculo funções que são simples de calcular, mas quase impossíveis de se reverter computacionalmente. Outras funcionalidades, como a distribuição de chaves de forma segura e a assinatura de mensagens são possíveis com o uso de criptografia assimétrica. Os algoritmos mais conhecidos são o RSA [23], o Diffie-Hellman [24], que utilizam números primos entre si muito grandes para gerar as suas chaves, e mais recentemente a Criptografia de Curva Elíptica (*Elliptic Curve Cryptography* (ECC)) [25], considerada a mais segura dado um mesmo tamanho de chave.

Em resumo, a criptografia simétrica é mais simples mas tem o problema de como compartilhar um segredo através de um meio inseguro enquanto a criptografia assimétrica não possui o problema de ter que divulgar um segredo mas requer um gasto computacional muito mais intenso. Assim, em redes cabeadas, é comum utilizar as características dos dois tipos de criptografia para garantir uma comunicação segura, o que é conhecido como criptografia híbrida. Primeiramente, é trocado um segredo entre os nós por intermédio das chaves públicas. Este segredo servirá como chave secreta para criptografar a comunicação posterior usando criptografia simétrica, de menor custo computacional. A Figura IV.1 mostra como funciona a criptografia híbrida. Primeiramente, os nós A e B trocam suas chaves públicas. Em seguida, o nó A gera uma chave secreta, a criptografa com chave pública de B e a envia. O nó B, então, decriptografa a mensagem com sua chave privada e gera uma mensagem contendo a chave secreta, criptografada com a chave pública de A, para confirmar que conseguiu obter o segredo. Após esses passos, a comunicação entre A e B é criptografada com a chave secreta trocada. É importante notar que um esquema como esse não é suficiente para garantir a autenticação e confiabilidade das mensagens, pois qualquer nó, agindo de forma maliciosa, poderia divulgar uma chave pública se dizendo

ser um outro nó, ou seja, usar uma identidade falsa. Assim, não basta a criptografia para manter o sigilo das mensagens, é necessário também a autenticação, ou seja, a garantia que um nó é realmente o nó que diz ser.

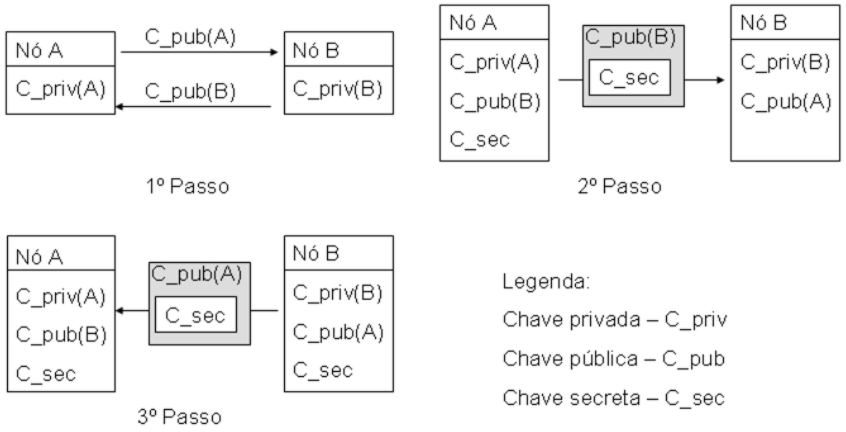

Figura IV.1: Criptografia híbrida.

IV.1.2 Infra-Estrutura de Chaves Públicas

O uso de uma Infra-Estrutura de Chaves Públicas (*Public-Key Infrastructure* - PKI) e assinatura digital permite solucionar o problema de interceptação, além de garantir a autenticação e o não-repúdio na rede. No PKI, existe uma terceira entidade, chamada autoridade certificadora (AC), capaz de garantir a quem pertence realmente uma chave pública, através da emissão, validação e revogação de certificados. Um certificado deve conter a identificação e a chave pública do nó, criptografados com a chave privada da AC. A assinatura digital será feita criptografando a mensagem ou o *h*ash da mensagem com a chave privada do emissor e enviando junto com ela um certificado emitido pela AC para aquele emissor. Assim, ao receber a mensagem, o destino decriptografa o certificado com a chave pública da AC, que deve ser conhecida por todos os nós da rede, obtendo a chave pública do emissor, que será a única chave capaz de decriptografar a mensagem. Assim, é possível garantir a autenticação e o não-repúdio, pois a chave privada deve ser mantida sempre em segredo pelos nós. É importante ressaltar que, apesar dessas características, tal mecanismo não garante a privacidade dos dados, o que poderia ser feito criptografando

todo o pacote com a chave pública do receptor ou ainda com uma chave secreta obtida por criptografia híbrida.

Uma outra entidade importante para o funcionamento da PKI é a Autoridade Registradora, que realiza o cadastro dos usuários que desejam ser certificados. Um usuário sem registro não pode assinar mensagens, pois a AC não será capaz de emitir certificados para ele.

IV.1.3 Funções *Hash*

Uma outra ferramenta importante que pode ser utilizada para garantir integridade dos dados são as funções *hash*. Uma função *hash* é definida como uma função H que mapeia uma seqüência de bits de tamanho arbitrário em uma de tamanho fixo. O conceito de função *hash* unidirecional foi introduzido em [26]. De maneira informal, uma função *hash* unidirecional deve ser simples de calcular, porém computacionalmente muito difícil de ser invertida. Para o uso em criptografia simétrica, ainda se exige que a função *hash* seja resistente a colisões, ou seja, é computacionalmente impossível encontrar duas seqüências distintas x e y tais que $H(x) = H(y)$. Os famosos algoritmos MD5 [27] e SHA-1 [28] foram projetados para possuírem essas propriedades. Além dessas propriedades básicas, as funções *hash* criptográficas geralmente possuem propriedades como a uniformidade dos valores de saída ao longo do conjunto imagem, a independência entre a entrada e a saída, a impossibilidade de inferência da saída ainda que partes da seqüência de entrada sejam conhecidas, etc. Devido a essas propriedades, essas funções também são conhecidas como funções de espalhamento.

IV.2 Criptografia Simétrica x Assimétrica: Vantagens e Desvantagens

A criptografia assimétrica trouxe inúmeros avanços, resolvendo questões como a autenticação e o não-repúdio através da Infra-Estrutura de Chave Pública (PKI). É importante destacar que os mecanismos que utilizam autenticação com chaves secretas e funções *hash* são capazes de ŞautenticarŤ os usuários que têm o segredo, a chave secreta. Portanto, se dois usuários compartilham uma chave simétrica, ou chave secreta, um pode autenticar o outro. Se um grupo de usuários possui uma chave secreta, também chamada de chave de grupo, qualquer componente do grupo pode ŞautenticarŤ qualquer outro componente do grupo como pertencente ao grupo, pois todos componentes do grupo possuem a chave secreta. No entanto, a autenticação é Şde pertencer ao grupoŤ ou autenticação de grupo, pois não é possível identificar qual usuário do grupo foi autenticado. Para uma autenticação individual é necessário um identificador ou Şsegredo únicoŤ o que é possível com a chave privada da criptografia assimétrica.

Assim, um sistema de comunicação seguro com n nós pode ser obtido com uma única chave simétrica compartilhada pelo grupo. Este sistema tem a desvantagem do gerenciamento da chave quando há uma inserção ou saída de nó ou ainda quando se deseja trocar a chave do grupo. Outra possibilidade é um sistema com criptografia simétrica por par de nós comunicantes, ou por enlace. Neste caso, cada nó deve possuir e armazenar a chave secreta correspondente a cada nó que deseja se comunicar. O total de chaves necessárias na rede é igual a$(n * (n-1))/2$ e cada nó armazena as $n-1$ chaves correspondentes à comunicação com os outros nós. É importante ressaltars que as chaves armazenada por cada nó são diferentes. Aqui, também, a tarefa de gerenciamentos de todas estas chaves é bem complexa. No caso da criptografia assimétrica cada nó deve possuir duas chaves, a simétrica e a assimétrica, e armazenar as chaves públicas dos outros nós com quem deseja se comunicar. A tarefa de gerenciamento é simplificada, pois não há necessidade de se manter segredo na divulgação das chaves públicas. No entanto, há necessidade de se certificar que a chave pública a ser usada é realmente de quem diz ser.

Cabe ressaltar que o tamanho da chave utilizada com a criptografia assimétrica tradicional é muito superior ao tamanho da chave da criptografia simétrica, o que pode ser um problema para nós com restrições de memória. Neste caso, é aconselhado o uso de criptografia elíptica, pois suas chaves são menores que a dos demais algoritmos assimétricos devido à alta complexidade da inversão da função elíptica.

Apesar de todas as vantagens do PKI, o seu uso em redes ad hoc não é simples. A primeira razão são as restrições de processamento e memória dos nós, o que torna muito complicado o cálculo para criptografar e decriptografar mensagens. Mesmo em redes com mais recursos, como as cabeadas, não se utiliza criptografia assimétrica para criptografar todas as informações, devido ao seu alto custo. Em redes de sensores sem fio, essa restrição é ainda maior, pois seus recursos são muito reduzidos. Medidas com sensores MICA mostraram que estes se tornam entre 100 e 1000 vezes mais lentos utilizando criptografia assimétrica [29]. Medidas utilizando *hardwares* específicos para os algoritmos RSA e DES mostraram um desempenho 1500 vezes mais lento do RSA [30]. Os recursos das redes ad hoc costumam ser menos restritos que os das redes de sensores, mas, ainda assim, o uso de criptografia assimétrica deve ser evitado ao máximo, tentando buscar formas de equilibrar suas vantagens com a facilidade da criptografia simétrica.

Deve-se ainda observar que a utilização de PKI exige uma terceira entidade para certificar a comunicação, o que é, geralmente, feito por um servidor. No entanto, redes ad hoc não possuem pontos centrais. Assim, um outro problema a ser solucionado é como distribuir as tarefas da AC pelos nós da rede e como fazer o registro das chaves públicas em uma rede que deve ser auto-configurável.

IV.3 Esquemas Criptográficos para Gerenciamento de Chaves

Alguns sistemas, como a Criptografia de Limiar, Criptografia por ID, Criptografia Comutativa, entre outros, já foram propostos como soluções para a distribuição de chaves

em redes ad hoc. A seguir, será descrita a proposta da Criptografia de Limiar, que substitui a PKI tradicional para redes ad hoc.

IV.3.1 Criptografia de Limiar

A criptografia de limiar (*threshold criptography*) é aplicada para solucionar o problema das Autoridades Certificadoras (AC). Através desse tipo de criptografia, um segredo D é dividido em n partes $(D_1, D_2, ..., D_i, ..., D_n)$, de maneira que o conhecimento de k ou mais D_i partes permite o cálculo de D, enquanto que o conhecimento de $k-1$ partes ou menos não permite determinar D. Um esquema como esse é chamado de esquema de limiar (k, n), e se aplica à distribuição da função de certificação de uma autoridade certificadora por um grupo de nós. Isto é eficiente em redes ad hoc, onde a existência de uma única autoridade certificadora introduz um ponto único de falha, enquanto o uso de réplicas da autoridade certificadora para melhorar a acessibilidade produz mais vulnerabilidades, pois a invasão de uma única réplica compromete a segurança de toda a rede.

Utilizando criptografia de limiar (k, n), onde $n = 2k - 1$, obtém-se um esquema de segurança robusto, pois mesmo que metade dos nós da rede fiquem comprometidos, ainda é possível reconstruir a chave k. O compromisso que se busca com a criptografia de limiar é entre segurança e conveniência, pois o ideal para a segurança é que todos os pedaços fossem necessários, embora para conveniência o ideal fosse utilizar o menor número de pedaços possível. Em outras palavras, no ambiente ad hoc se busca o equilíbrio entre a disponibilidade e tolerância à intrusão. Assim, um adversário precisa destruir $(n - k + 1)$ nós para indisponibilizar o serviço, ou ainda roubar o segredo de k nós para obter a chave secreta [31].

A criptografia de limiar é ideal para aplicações nas quais um grupo de indivíduos com interesses conflitantes mutuamente suspeitos devem cooperar. O esquema de criptografia de limiar proposto em [32] é baseado na interpolação polinomial de Lagrange com complexidade $O(nlog^2(n))$. Neste esquema, cada compartilhamento é calculado de um grupo de polinômios com grau k em um esquema (k, n). Usando os k valores em um sistema

de equações lineares é possível encontrar o segredo, e usando menos que k equações se obtém uma indeterminação. Um outro esquema baseado nos espaços euclidianos pode ser encontrado em [33], onde o número de espaços é dado por k e o segredo compartilhado é dado por um ponto no espaço. Cada compartilhamento é um plano contendo o segredo, e a interseção de k planos determina o ponto. Se $k = 3$, na ausência de alguns compartilhamentos, se obterá um plano ou uma linha, com infinitas possibilidades, o que torna o segredo indeterminável [34].

Uma vez que em redes ad hoc não se deseja que nenhum nó possua o segredo da autoridade certificadora, pois ele se tornaria um alvo de ataques, não é adequado que a chave possa ser recuperada, mesmo juntando k pedaços da mesma. O que se utiliza é a criptografia de limiar para fazer assinaturas. Nessa criptografia, cada nó de posse de um pedaço da chave privada da autoridade certificadora (AC) fará uma assinatura parcial da mensagem, de forma a que, quando todas as k assinaturas parciais forem obtidas, seja possível construir a assinatura completa. Na Figura IV.2 a chave S é dividida em n pedaços, que são distribuídos entre n nós. Um nó que deseje uma assinatura para a mensagem M deverá enviá-la aos n nós, e ainda que $n - k$ assinaturas se percam, ele será capaz de reconstituir a assinatura de M. Uma multi-assinatura de limiar através da combinação do RSA com a criptografia de limiar é proposta em [35].

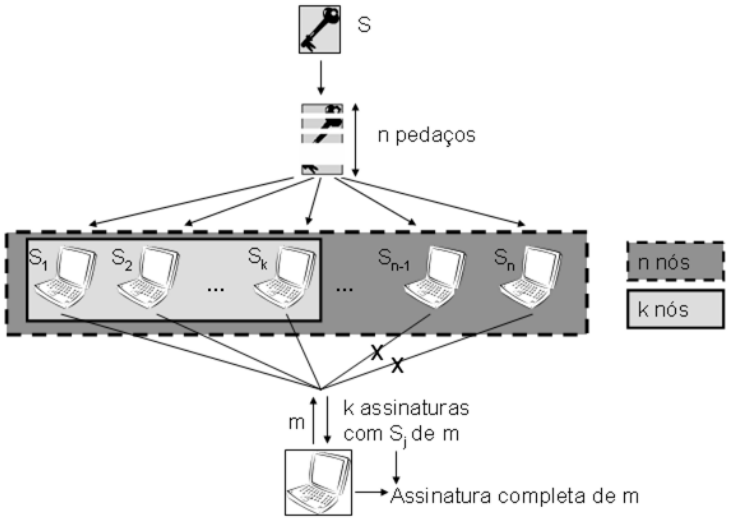

Figura IV.2: Esquema de assinatura com criptografia de limiar.

O maior problema desse tipo de criptografia é que os nós maliciosos podem enviar assinaturas parciais inválidas, o que geraria uma assinatura completa inválida. Assim, o nó que desejar que sua mensagem seja validada deve ser capaz de testar todas as possibilidades de combinações de chaves até que ele obtenha uma assinatura completa válida. Esquemas mais robustos para essa composição de chaves também foram propostos, como o *Digital Signature Standard* (DSS) de limiar e são baseados em redundâncias entre as chaves [36] [37].

Existem sistemas de atualização de chaves, ou pró-ativos [36], que têm como princípio impedir que os atacantes consigam obter k compartilhamentos da chave. Uma vez que um atacante leva um tempo até conseguir roubar um segredo compartilhado de um nó, se os segredos nunca mudarem, ele consegue, após certo longo tempo, roubar todos os k segredos. Se for aplicado um sistema de atualização de chaves, isso pode ser impedido. Nos sistemas de criptografia de limiar pró-ativos, os novos segredos são computados a partir dos antigos, por meio de colaboração dos servidores sem que o sistema de certificação deixe de funcionar. Os novos segredos compõem um compartilhamento $(n, t+1)$ da chave secreta, e não podem ser descobertos ainda que o invasor consiga obter todos os compartilhamentos antigos. O processo de obtenção dos novos segredos começa com todos os n nós gerando n compartilhamentos S_{ij} do compartilhamento S_i que ele possui do segredo S. Em seguida, por meio de um canal seguro, cada nó j receberá as chaves S_{1j} até S_{nj} dos seus vizinhos, podendo gerar $S'_j = S_j + \sum_{i=1}^{n} S_{ij}$. Esse esquema está representado na Figura IV.3.

Pode-se, então, dizer que a técnica da criptografia de limiar trouxe melhorias às redes ad hoc, permitindo dar integridade e confidencialidade aos dados, autenticação, não-repúdio e disponibilidade do serviço de certificação.

IV.3.2 Algumas Considerações

Vários sistemas para gerenciar chaves têm sido propostos recentemente, utilizando os esquemas citados na seção anterior. No entanto, esses sistemas utilizados isoladamente não são capazes de garantir todos os requisitos de segurança da rede. Primeiramente,

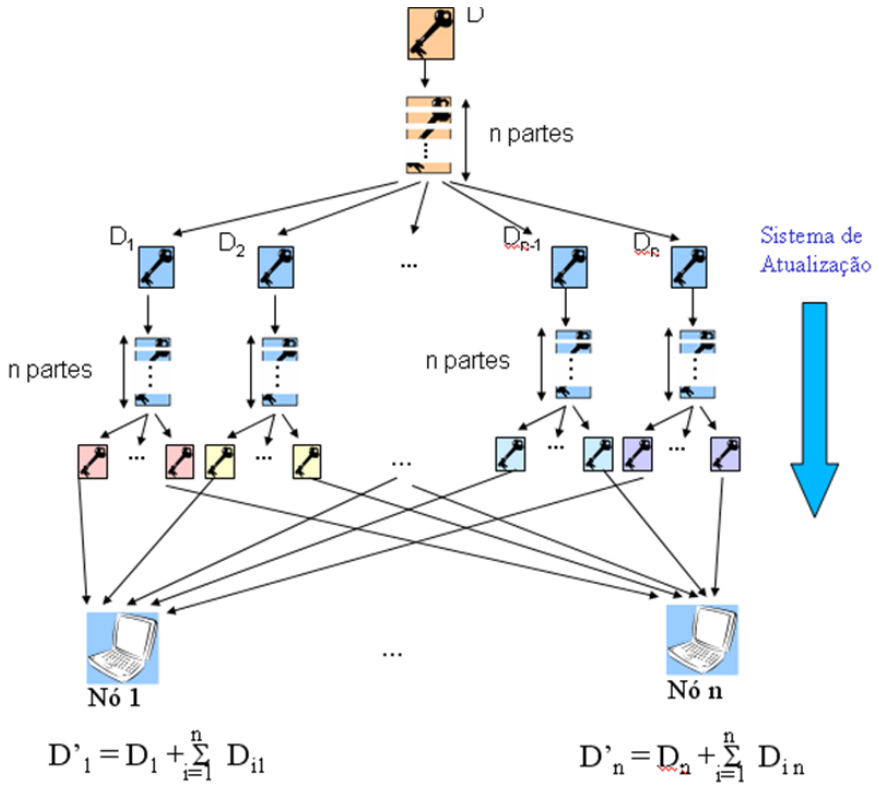

Figura IV.3: Esquema de atualização de chaves na criptografia de limiar.

deve-se levar em conta que todos os sistemas que assumem a existência de um segredo pré-estabelecido supõem a atuação prévia de um administrador, o que não é condizente com o cenário de redes ad hoc, embora em muitos casos seja necessário.

Muitos estudos foram feitos sobre os esforços para quebrar os sistemas propostos, e a escolha por cada um deles deve levar em consideração tanto o custo de mensagens inseridas na rede quanto o custo dos recursos utilizados, além do custo de quebra, de acordo com o cenário utilizado. É evidente que o nível de segurança de um cenário militar é totalmente diferente do de uma rede ad hoc domiciliar, e isso deve ser considerado na escolha do mecanismo de gerenciamento de chaves. Outro ponto que deve ser observado é a questão da sobrecarga de mensagens e processamento que é criado na rede. Por exemplo, a criptografia de limiar pode ser mais adequada a rede ad hoc, mas ela introduz

muito mais processamento e mensagens, e muitas vezes será mais adequado a utilização de réplicas de uma única autoridade certificadora.

Capítulo V

Protocolos de Roteamento Seguros: Descrição, Análise e Conclusões

Os protocolos de roteamento inicialmente propostos para redes ad hoc sem fio não continham nenhum mecanismo de segurança, pois consideram apenas cenários onde todos nós são confiáveis, premissa que compromete a sua robustez em ambientes hostis. Com a divulgação das vulnerabilidades das redes ad hoc, tornou-se evidente a fragilidade dos protocolos. Desde o início, percebeu-se que propostas de segurança convencionalmente usadas na Internet não se aplicavam ao roteamento. O IPSec (*IP Security*) [38], por exemplo, somente é válido para autenticação fim-a-fim e para prover segurança entre entidades que já possuem roteamento estabelecido entre si, não garantindo roteamento seguro.Foram, então, propostos novos protocolos que visam garantir a integridade, a autenticidade e o não-repúdio das mensagens de roteamento. Entre as principais propostas de protocolos de roteamento seguro estão o *Authenticated Routing for Ad hoc Networks* (ARAN) [39], o Ariadne [40], o *Secure Routing Protocol* (SRP) [41], entre diversos outros. Nesta seção, são abordados apenas o *Secure Ad hoc On-Demand Distance Vector* (SAODV) [42] e o *Secure Optimized Link State Routing protocol* (SOLSR) [43] por serem as versões seguras dos protocolos ad hoc mais utilizados atualmente.

V.1 SAODV

O *Secure AODV (SAODV)* [42] é uma extensão do protocolo AODV (*Ad Hoc On-Demand Distance Vector Routing Protocol*) [3], que garante segurança no processo de descoberta de rota. Zapata e Asokan propõem adicionar algumas extensões de mensagens chamadas de "Extensão de Assinatura" (*Signature Extension*), no intuito de acrescentar funcionalidades ao protocolo que permitam assegurar a integridade, a autenticação e o não-repúdio das informações de roteamento. De fato, o que se tenta evitar é o não-repúdio e a garantia de que o autor da mensagem, que deve estar íntegra, é quem realmente se diz ser. Assim, se evita que alguns campos, como o número de saltos e o IP de origem sejam modificados durante o processo de descoberta de rotas.

O SAODV possui dois mecanismos de proteção das mensagens de roteamento: assinatura digital e cadeias de *hash*. A assinatura digital garante autenticação fim-a-fim dos campos imutáveis da mensagem, que devem ser assinados pelo nó de origem antes do envio da mensagem. Exemplos destes campos imutáveis são os IPs de origem e de destino. No entanto, nas mensagens de pedido e resposta de rota do protocolo AODV, RREQ e RREP repectivamente, existe o campo número de saltos que deve ser modificado salto-a-salto pelos nós intermediários. Assim, não é possível fazer uma assinatura digital fim-a-fim para que seja possível a modificação deste campo pelos nós intermediários. Este campo também não pode ficar livre de qualquer proteção uma vez que um nó malicioso pode propositalmente reduzir o número de saltos no intuito de atrair rotas. O que se conclui é que o seu controle deve ser feito salto a salto e para realizar essa tarefa se utilizam as cadeias de *hash*.

V.1.1 Mensagens do SAODV

As mensagens do SAODV são extensões às mensagens do AODV. Nelas são inseridas assinaturas e outros dados que permitem maior segurança contra a falsificação de mensagens. Na Figura V.1 estão representados os campos dos pacotes de controle do AODV.

Nela estão destacados todos os campos mutáveis que não podem ser protegidos com a assinatura digital fim-a-fim. Estes campos são o número de saltos nas mensagens RREQ e RREP, protegido pela cadeia de *hash*, e os campos número de destinos inalcançáveis e a descrição destes destinos, composta do IP e número de seqüência das mensagens RERR que é tratado na sub-seção V.1.4.

Nas Figuras V.2 e V.3 estão representadas as extensões de assinatura que, junto com os campos das mensagens do AODV, formam o pacote do SAODV. Assim, um pacote de RREQ (*Route Request*) do SAODV é composto pelos campos da Figura V.1a e os campos da Figura V.2. A formação do RREP, RERR e RREP-ACK (*Route Reply*) são semelhantes à do RREQ, e as extensões de assinatura do RERR (*Route Error*) e do RREP-ACK estão na Figura V.3. Por essas representações é possível ver que os pacotes do SAODV possuem um tamanho superior ao do SAODV. A quantidade de bits excedentes está representada na Tabela V.1.

Tabela V.1: Carga extra em pacotes devido a utilização do SAODV.

Pacote	**Tamanho Extra Mínimo**
RREQ + Extensão	128 bits
RREP + Extensão	128 bits
RERR + Extensão	96 bits
RREP-ACK + Extensão	48 bits

Cabe ressaltar que o tamanho máximo das extensões depende do tipo de *hash* utilizado e também do tipo de assinatura e, desta forma, não é possível dizer o tamanho extra máximo para o pacote. Ambos os campos *hash* e assinatura devem ser preenchidos com zeros até que se complete um tamanho múltiplo de 32 bits para garantir o alinhamento do pacote.

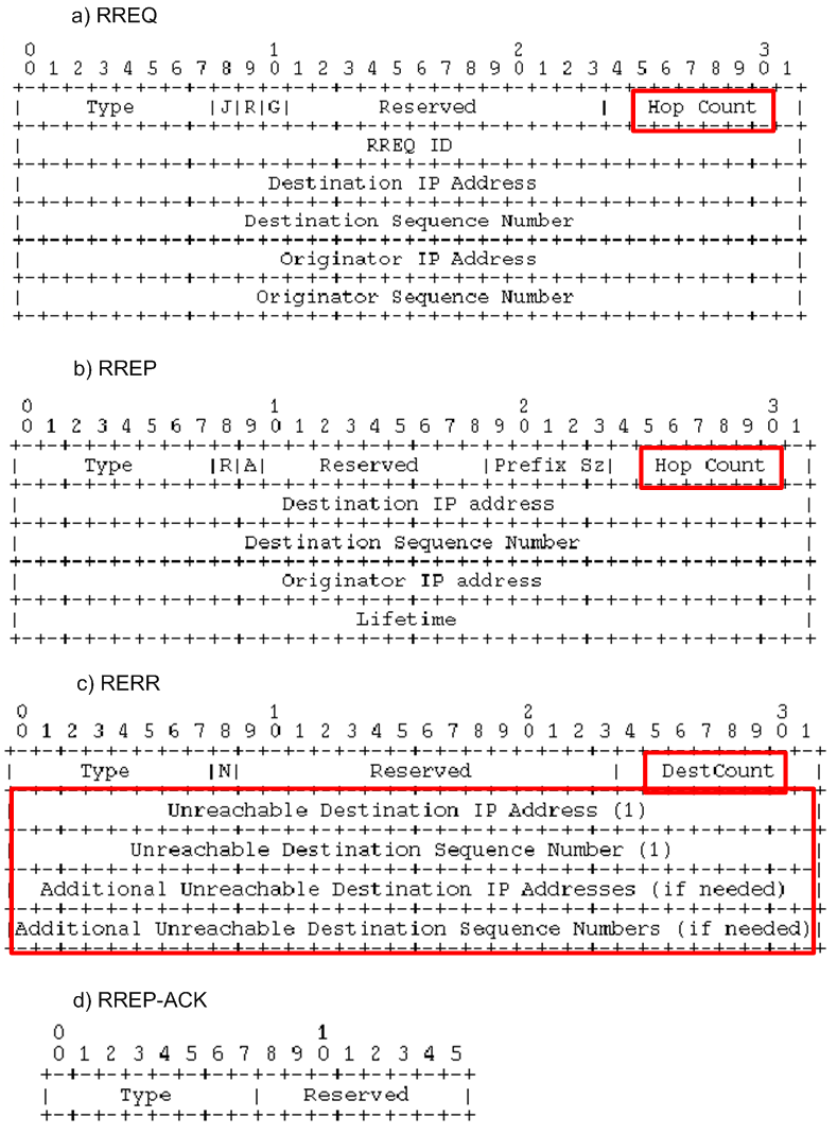

Figura V.1: Campos dos pacotes do AODV. Em destaque estão os campos mutáveis dos pacotes.

V.1.2 Cadeias de *hash* no SAODV

Uma cadeia de *hash* (*hash chain*) é definida como uma seqüência gerada a partir da aplicação sucessiva de uma função *hash* a uma semente, geralmente um número gerado aleatoriamente. Dado um dos elementos da cadeia de *hash*, pode-se garantir que os valores seguintes fazem parte da mesma cadeia, aplicando-se a função *hash* repetidamente sobre o elemento conhecido um número adequado de vezes. A unidirecionalidade da função

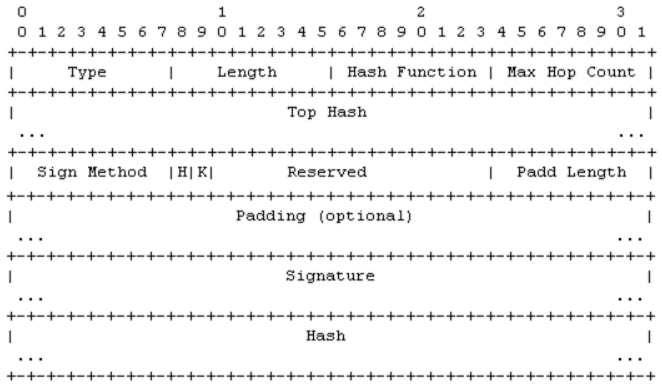

Figura V.2: Extensões de segurança para os pacotes RREQ e RREP do SAODV.

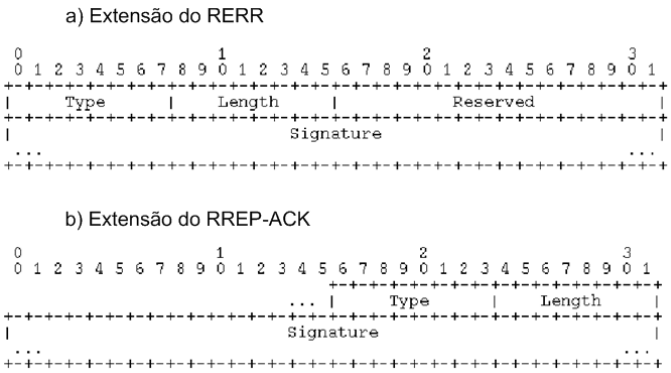

Figura V.3: Extensões de segurança para o RERR e o RREP-ACK do SAODV.

hash impede que se obtenham os elementos anteriores da cadeia. Assim, um terceiro nó intermediário no caminho da origem para o destino não consegue obter o valor *hash* dos nós anteriores e, portanto, não pode anunciar uma rota com um menor número de saltos.

Para criar uma cadeia de *hash* de n elementos, um nó deve gerar uma semente aleatória h_0 e calcular a lista de valores $h_0, h_1, h_2, h_3, ..., h_n$, onde $h_i = H(h_{i-1})$ para $0 < i \leq n$, como representado na Figura V.4. Dessa maneira, ao inicializar a cadeia de *hash*, os valores da lista são gerados da esquerda para direita. Em seguida, esses elementos podem ser utilizados para garantir a segurança da atualização das mensagens de roteamento, por exemplo. Nesse caso, para autenticar os campos atualizados da mensagem, o nó deve seguir da direita para a esquerda. Dado um elemento previamente autenticado da cadeia de *hash*, é possível verificar a pertinência dos elementos posteriores. Por exemplo, su-

pondo conhecido o valor autenticado h_i, pode-se autenticar o elemento h_{i-3} calculando-se $H(H(H(h_{i-3})))$ e verificando-se se o valor calculado é igual ao valor previamente autenticado h_i.

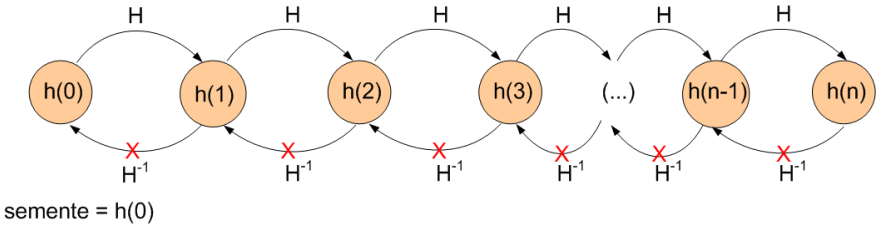

Figura V.4: Formação da cadeia de *hash*.

No SAODV, a cadeia de hash é aplicada para impedir que o número de saltos seja decrementado. Para tanto, ao enviar uma mensagem, o nó de origem deve inicializar o campo *Hash* com uma semente aleatória, o campo Número Máximo de Saltos com o valor desejado e o campo *Top Hash* com o resultado da aplicação da função *hash* sobre a semente um número de vezes igual a Número Máximo de Saltos . Quando um nó intermediário, no caminho da origem para o destino, receber a mensagem, pode autenticar o campo Número de Saltos, aplicando a mesma função *hash* um número de vezes igual a diferença entre Número Máximo de Saltos e Número de Saltos sobre o campo *Hash* e comparando o resultado com o campo *Top Hash*. Se os campos foram iguais, pode-se assegurar que o campo Número de Saltos não foi alterado e, então, o nó intermediário deverá incrementar o campo Número de Saltos e aplicar a função *hash* no campo *Hash* antes de reencaminhar a mensagem. Caso contrário, a mensagem deverá ser descartada. Dessa maneira, o mecanismo de cadeias de *hash* impede que um nó malicioso diminua o Número de Saltos da mensagem de roteamento, uma vez que não se podem obter os valores anteriores da seqüência, dada a propriedade unidirecional da função *hash*.

V.1.3 Extensão de assinaturas duplas

O SAODV ainda prevê extensões de assinaturas duplas (*Double Signature Extension*), que permitem que nós intermediários respondam imediatamente à origem quando eles já

possuírem uma rota atualizada para o destino, assim como é previsto no funcionamento original do AODV.

No SAODV, um número de seqüência só é aceito se estiver assinado pelo nó correspondente. Assim, existe uma dificuldade para o nó intermediário responder com uma rota conhecida, pois ele não teria como assinar pelo nó que está sendo procurado. Assim, foi criada a extensão de assinatura dupla, que permite que o nó que está sendo procurado autentique previamente o seu número de seqüência para que o nó intermediário possa enviar a mensagem de RREP. Neste esquema, sempre que um nó envia um RREQ ou um RREP, ele também inclui os *flags* de RREP, o tamanho do prefixo e a assinatura que pode ser utilizada pelo nó intermediário. Esses campos são utilizados no RREP gerado pelo nó intermediário que deverá assinar apenas a nova estampa de tempo, pois a validade da rota é diferente da validade original. Assim, o nó que receber esse pacote, com posse dos *flags* e do número de seqüência provenientes do RREQ ou do RREP originado pelo nó que está se buscando, pode checar que a informação de número de seqüência realmente vem do nó buscado, e pela assinatura da nova validade da rota, pode checar se a resposta veio realmente do nó intermediário indicado pelo campo IP de origem do pacote.

No caso da utilização de assinaturas duplas, o pacote de RREQ é formado pela união dos campos das Figuras V.1a e V.5a. A formação do RREP (*Route Reply*) é semelhante à do RREQ, utilizando a extensão de assinatura dupla representada na Figura V.5b. A quantidade de bits excedentes com relação ao AODV devido as extensões de assinatura dupla está representada na Tabela V.2.

Tabela V.2: Carga extra em pacotes devido a utilização do SAODV com assinatura dupla.

Pacote	**Tamanho Extra Mínimo**
RREQ + Extensão dupla	192 bits
RREP + Extensão dupla	192 bits

a) Extensão dupla de assinatura do RREQ

b) Extensão de assinatura dupla do RREP

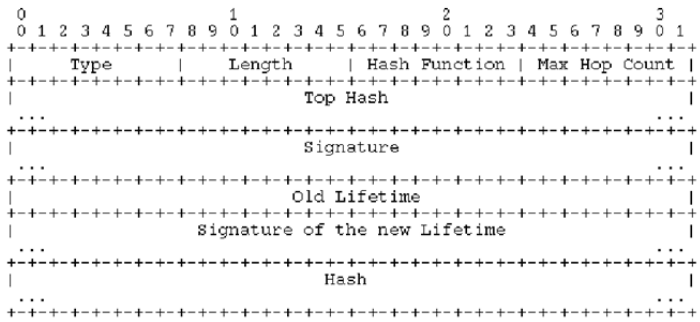

Figura V.5: Extensões de segurança do RREQ e do RREP para dupla assinatura do SAODV.

V.1.4 Mensagens de erro do SAODV

As mensagens de erro do SAODV (RERR) têm um tratamento especial com relação à questão das assinaturas. De fato, não é relevante quem foi o originador da mensagem de erro. Uma vez que o AODV se baseia no roteamento de vetor distância, os nós não conhecem toda a topologia, pois conhecem apenas o próximo nó para quem devem encaminhar a mensagem visando o destino. Caso o seu vizinho resolva não encaminhar mais pra um determinado destino, independente da rota ser válida ou não, ela deve ser descartada. Assim, não importa quem foi que originou a mensagem, mas sim, quais informações o nó vizinho levou em consideração como rotas que estão inválidas. Portanto, a autenticação das mensagens de erro deve ser feita nó-a-nó, e não fim-a-fim. Dessa forma, cada nó, ao receber uma mensagem de erro, RERR, deve conferir a assinatura, para garantir que a mensagem veio do seu vizinho, excluir as rotas de sua tabela, atualizar a lista das rotas indisponíveis, assinar a mensagem e encaminhar para o seu vizinho.

Outro ponto que deve ser observado é que os números de seqüência das mensagens de erro não devem ser utilizados para atualizar a tabela de rotas, pois estes números não estão assinados pelos nós correspondentes, podendo se tratar de uma informação falsa. Assim, caso este cuidado não fosse tomado, um nó malicioso pode realizar o Ataque Bizantino e aumentar o número de seqüência de um determinado nó indicado na mensagem de erro, de forma a fazer com que as próximas informações atualizadas desse nó sejam descartadas.

A princípio, poderia se imaginar que o tratamento correto para essas mensagens seria a assinatura fim-a-fim. No entanto, as mensagens de erro têm muitos campos que são mutáveis, como o campo DestCount, que contabiliza o número de nós inalcançáveis e os campos de IP e número de seqüência dos nós inalcançáveis, que podem ser vistos na Figura V.1c. Todos esses campos devem ser atualizados em cada salto da mensagem de erro, o que dificulta a autenticação da mensagem fim-a-fim.

V.2 SOLSR

O SOLSR (*Secure Optimized Link State Routing Protocol*) [43][44] é uma versão segura do protocolo de roteamento OLSR [1], cuja idéia é assinar, usando chaves simétricas, cada pacote de controle do OLSR a fim de garantir a autenticidade das mensagens.

No SOLSR, todo tráfego é assinado salto a salto, o que significa que é garantida a segurança até de campos que devem ser atualizados pelos nós intermediários, como o número de saltos e o campo TTL (*time-to-tive*). Além disso, somente é necessária uma assinatura por salto, apesar de existirem muitas mensagens de roteamento encapsuladas em um único pacote do OLSR. Por outro lado, a desvantagem é que a abordagem salto a salto não garante assinaturas fim-a-fim, já que um pacote recebido por um nó não terá sido assinado pelo nó de origem, mas apenas pelo nó anterior. Apesar disso, o protocolo determina que os nós só devem encaminhar pacotes oriundos de nós confiáveis. Conseqüentemente, os nós de uma dada rota são confiáveis, dois a dois. O processo de assinatura digital utiliza uma função *hash* com chave, de forma que um nó que não tenha

acesso à chave secreta não pode reproduzir a assinatura do nó emissor.

V.2.1 Mensagens do SOLSR

Foram definidos quatro tipos de mensagem para o SOLSR, sendo uma para transportar a assinatura e três para realizar a troca de estampas de tempo. Todas essas mensagens são transportadas no corpo de uma mensagem do OLSR. A estrutura da mensagem que leva a assinatura está representada na Figura V.6. As outras mensagens estão representadas na Figura V.7 [43].

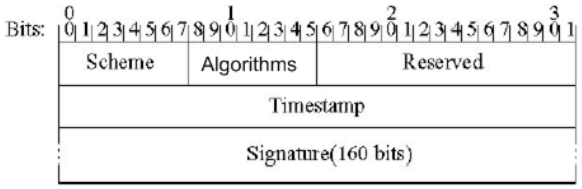

Figura V.6: Mensagem de assinatura do SOLSR.

V.2.2 Assinatura das mensagens

A assinatura exibida na Figura V.6 é anexada a todos os pacotes do SOLSR. Os campos *scheme* e *algorithms* servem para informar ao receptor qual o esquema de assinatura e qual tipo de algoritmo foi utilizado.

Para fazer a assinatura, é utilizada a função *hash* SHA-1 da mensagem com a chave simétrica, produzindo um total de 160 bits como resposta da função. O conteúdo da mensagem que serve para entrada do *hash* é composto pelo cabeçalho do OLSR, todas as mensagens do OLSR no pacote, com exceção da mensagem de assinatura, o cabeçalho e estampa de tempo da mensagem de assinatura. Para tornar a assinatura segura, utiliza-se a chave compartilhada para fazer o *hash*. Cada nó, ao receber a mensagem, aplica a função com as entradas disponíveis no pacote e a sua chave simétrica, e, em seguida, compara o resultado obtido com o que está no campo de assinatura da mensagem. Se os valores foram iguais, significa que aquela mensagem realmente veio do vizinho. Então, ele

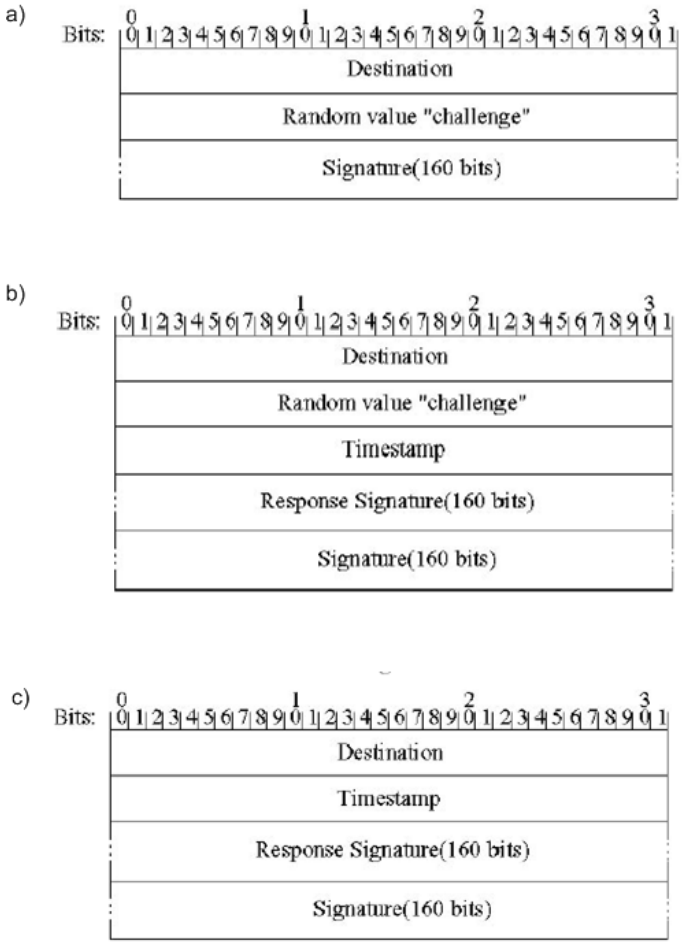

Figura V.7: Mensagens de estampa de tempo do SOLSR.

atualiza os campos mutáveis do pacote e recalcula o *hash* da mensagem, enviando-a em seguida aos seus vizinhos.

V.2.3 Estampas de Tempo

Um nó que não tenha a chave secreta não pode criar uma mensagem válida. No entanto, um nó malicioso pode receber uma mensagem válida e a repetir mais tarde. Os nós que receberem esta mensagem não têm como identificar que ela veio de um nó malicioso. Portanto, a fim de prevenir contra ataques da replicação, estampas de tempo são utilizadas no SOLSR. Essa solução não é baseada em sincronia de tempo e supõe a troca de estampas entre dois nós durante a conexão inicial.

No ataque da replicação, o atacante pode guardar tráfego antigo para reencaminhá-lo posteriormente, congestionando a rede, ou ainda causando *loops* de roteamento. Os números de seqüência podem prevenir tal ataque em mensagens que são inundadas na rede. No entanto, em mensagens como o HELLO, que deve ficar restrito ao alcance do seu emissor, números de seqüência não conseguem evitar o ataque, pois um nó malicioso poderia recolher todo o tráfego de um nó durante certo tempo e posteriormente repetí-lo em outro ponto da rede. Este ataque é eficaz porque, no novo ponto da rede, aquelas mensagens nunca teriam sido ouvidas e seriam consideradas como novas. Assim, a estampa de tempo poderia evitar que algum nó malicioso replicasse as mensagens de HELLO em outra área da rede, pois uma vez que o pacote está assinado, ele não teria como atualizar a estampa escrita no pacote.

Para evitar o ataque da replicação sem utilizar mecanismos de sincronização, o SOLSR utiliza suas mensagens de estampa de tempo. Essas mensagens são trocadas sem que sejam verificados os campos de assinatura, uma vez que elas são encaminhadas entre vizinhos que não registraram estampas de tempo em comum. Assim, para garantir a segurança, elas carregam uma assinatura interna própria.

A troca de estampas ocorre em três passos e tem como objetivo garantir que os dois nós saibam a diferença de tempo entre eles. O processo é iniciado sempre que um nó A recebe uma mensagem de um nó B e eles não possuem um registro de estampa. O nó A, então, gera uma mensagem, representada na Figura V.7a, contendo o IP de B e um *nonce*, que é um número aleatório escolhido por A para ser utilizado uma única vez. A mensagem é assinada, calculando-se o *hash* de toda a mensagem com a chave compartilhada. Ao receber essa mensagem, B checa a sua validade, recalculando o *hash*, e, em seguida, responde a A com uma mensagem desafio, representada na Figura V.7b, contendo o IP de A, a assinatura da mensagem enviada por A, um *nonce* gerado por B e a estampa de tempo de B. A mensagem é então assinada utilizando o *hash* com a chave compartilhada. Ao receber essa mensagem, A checa os valores das funções *hash* e calcula a diferença entre a sua estampa e B, armazenando esse valor. Por fim, A gera uma mensagem, representada na Figura V.7c, contendo o IP de B, a sua estampa de tempo e

o *hash* da mensagem enviada por B, assinando toda a mensagem. Ao receber essa última mensagem, B verifica sua validade, calcula a diferença de estampas de tempo, também armazenando esse valor, e assim se conclui o processo de registro de estampa de tempo entre dois nós.

É importante notar a necessidade de se incluir *nonces* nas mensagens, pois são eles que impedem o reenvio dessas mensagens por nós maliciosos. A segunda e a terceira mensagens do processo devem ser assinadas com os *nonces* dos dois nós que estão estabelecendo o registro, de forma que um nó malicioso não poderia simplesmente repetir as mensagens do nó A ou do nó B para realizar um processo semelhante falso, pois o outro lado provavelmente irá gerar um *nonce* diferente, de forma que a assinatura da mensagem não ficará correta. Outro ponto interessante é que a solução apresentada não exige sincronização de tempo. O nó precisa apenas guardar a diferença entre a sua estampa de tempo e a do nó com o qual está se comunicando. Para evitar erros devido a atrasos de transmissão, é permitido um intervalo S na diferença entre as estampas. Assim, se o nó enviou a mensagem com uma diferença de estampas calculada igual à T_N e T_0 é a diferença entre as estampas de tempo local e remota previamente armazenada, o nó considera a mensagem válida se T_N estiver no intervalo $]T_0 - S, T_0 + S[$. Para atualizar a diferença entre estampas, sempre que uma mensagem cuja assinatura é verificada como correta é recebida, o valor de T_0 é recalculado como $\frac{T_0+T_N}{2}$.

Para evitar ataques de negação de serviço, os nós possuem um contador de tempo que é disparado ao fim de cada processo de registro de estampa de tempo para o outro nó. Enquanto o contador não chegar a um determinado tempo, qualquer pedido de troca de estampas do outro nó será ignorado, evitando ataques onde o nó malicioso envia inúmeros pedidos de troca de estampa em um espaço de tempo muito curto, gerando excesso de processamento no nó atacado, além de impedí-lo de realizar outras funções.

V.3 Vulnerabilidades dos Protocolos SAODV e SOLSR

Os protocolos seguros normalmente especificam que tipo de segurança provêem. Assim, estes protocolos assumem algumas premissas e não são capazes de proteger a rede contra todos os tipos de ataque. Para isso, seria necessária a utilização de outros mecanismos que fossem capazes de detectar intrusões e puní-las. Assim, é necessário analisar a que tipos de ataque cada um dos protocolos seguros ainda está exposto, na ausência de outros mecanismos de segurança, e supondo um bom sistema de distribuição de chaves, que inclui uma infra-estrutura de chaves públicas adequada para o SAODV e um sistema de atualização de chave de grupo adequado ao SOLSR. Além disso, também supõe-se que a política de acesso esteja sendo feita de forma adequada, de maneira a garantir que, ainda que existam atacantes na rede, eles sejam minoria.

V.3.1 SAODV

Apesar de o SAODV resolver todos os problemas relativos à identidade falsa e mudança de conteúdo de pacotes, devido ao uso de assinaturas digitais, ele ainda é vulnerável a diversos outros ataques. Uma vez que ele não possui mecanismos que identifiquem e punam mau comportamento, ainda é possível que nós maliciosos realizem ataques de prioridade, como a Ganância e de descarte, tais como o Encaminhamento Seletivo, o Buraco Negro e o Buraco Cinza. Os ataques de atração de pacote, como o Ataque da Pressa e o Túnel de Minhoca também podem ser realizados, pois nenhum dos dois depende diretamente do conteúdo do pacote de roteamento. O único tipo de atração que foi impedida foi a modificação do número de saltos, que é o tipo de atração mais simples de se realizar. O Ataque da Pressa ganhou um facilitador a mais que é a criação de filas nas interfaces dos vizinhos, pois agora é possível gerar assinaturas digitais falsas, que por gerar excesso de processamento, cria filas nas interfaces dos vizinhos.

Um dos problemas do AODV que não foi solucionado no SAODV, apesar da solução ser simples é a verificação da bidirecionalidade do enlace, que permitiria evitar o ataque

da Inundação de HELLOs. O ataque da replicação, que foi solucionado no SOLSR, foi deixado em aberto no SAODV, o que permite que nós maliciosos capturem grande quantidade de tráfego legítimo e a reproduza em outras áreas da rede, criando enlaces que não existem de fato. Isso é possível pela reprodução de HELLOs antigos em outras áreas distantes, fazendo nós distantes acreditarem que aquele nó é um vizinho.

Por fim, é importante comentar que o SAODV não é imune a ambientes onde seja possível a realização de violações. Uma vez que a base de toda a sua segurança reside na segurança da autoridade certificadora e das chaves privadas dos nós, basta que qualquer nó seja violado para que o nó malicioso consiga ter mais que uma identidade e possa realizar o Ataque Bizantino, pelo menos com os pacotes do nó que violou.

Apesar de todos esses problemas, o SAODV pode ser considerado eficiente, uma vez que ele é capaz de impedir os ataques mais comuns e mais simples de serem realizados. De todo modo, os mecanismos para evitar os outros ataques podem ser muito custosos, tantos em termos de processamento quanto de mensagens de controle e, assim, usá-los em conjunto com o SAODV restringe a rede a ser formada apenas por dispositivos com alto potencial de processamento e de memória.

V.3.2 SOLSR

O SOLSR, por utilizar um esquema com chave de grupo, expõe toda a rede na existência de usuários maliciosos que façam parte do grupo. Outro ponto ainda mais grave é que esses usuários também podem repassar o segredo da rede para outros usuários maliciosos, que participarão do grupo como usuários legítimos. Assim, mesmo com um bom gerenciamento de chave de grupo, a rede que só utiliza esse tipo de autenticação está seriamente sujeita a diversos ataques advindos dos usuários internos, pois muitos dos problemas de rede vem pelos usuários internos, e não pelos externos. Assim, diferentemente do SAODV, o SOLSR, no caso de existirem atacantes internos, está exposto ao Ataque Bizantino, Direcionamento Falso, Sybil e Identidade Falsa. Todos esses ataques se devem a ausência de alguma forma de garantir a integridade da informação transmitida e a autenticidade do

nó originador da mensagem, já que chaves de grupo não permitem tal controle. Outros ataques exclusivos do SOLSR são o Estouro da Tabela de Roteamento e o Envenenamento de *Cache*, que também ocorrem pela ausência de uma forma de associar integridade do conteúdo com a origem dos dados.

As vulnerabilidades comuns aos dois protocolos são a Ganância, o Encaminhamento Seletivo, o Buraco Negro e o Buraco Cinza. Da mesma forma que o SAODV, o SOLSR carece de meios de detecção e punição de intrusão. Já o Ataque da Pressa e o Túnel de Minhoca têm uma maior dificuldade para serem realizados, embora não sejam impedidos, devido à troca de estampas de tempo. Caso a rota passe a ficar muito mais curta ou rápida, o intervalo limite de variação da estampa de tempo pode ser estourado e a assinatura passa a ser inválida. Assim, seria necessário um novo processo para troca de estampa, o que geraria um pequeno atraso na execução do ataque. De fato, a existência das estampas vai evitar o Ataque da Replicação, o qual o SAODV não consegue evitar.

O SOLSR também não é resistente à violação, pois caso um nó seja violado, o segredo da rede ficaria exposto e tantos nós maliciosos quanto fosse desejado poderiam ser inseridos na rede. Assim, o efeito da violação nesse protocolo é mais grave, pois, no SAODV, apenas uma identidade fica exposta com uma violação, enquant que no SOLSR, toda a rede fica exposta.

Desta análise do SOLSR pode-se concluir que toda a segurança desse protocolo está inserida na confiança que se tem nos nós que pertencem à rede. Assim, ele não é adequado a redes onde se desconfie dos usuários e exista a necessidade de medidas fortes de segurança. Por outro lado, ele é um protocolo simples, que não adiciona grande carga de mensagens ou processamento a rede, de forma que seria possível utilizar outros mecanismos, tal como um sistema de confiança em conjunto com ele. Além disso, seu sistema de estampas de tempo permite evitar ou dificultar todos os ataques que se valem da falta de sincronia na rede para funcionar.

Um resumo dos ataques que o SAODV e SOLSR são capazes de previnir está na Tabela V.3, para atacantes internos, e na Tabela V.4, para atacantes externos.

Tabela V.3: Relação de ataques solucionados ou não pelos protocolos SAODV e SOLSR, supondo atacantes internos.

Ataque	**SAODV**	**SOLSR**
Ataque Bizantino	Parcialmente	Não
Estouro da Tabela de Roteamento	Sim	Não
Envenenamento de cache	Sim	Não
Replicação de Pacotes	Não	Sim
Direcionamento Falso	Sim	Não
Inundação de Hello	Não	Sim
Ganância	Não	Não
Buraco Negro	Não	Não
Encaminhamento Seletivo	Não	Não
Ataque da Pressa	Parcialmente	Parcialmente
Túnel de Minhoca	Não	Não
Sybil	Sim	Não
Identidade Falsa	Sim	Não
Replicação de Identidade	Sim	Sim

V.4 Testes de Desempenho

V.4.1 Características das Implementações dos Protocolos SAODV e SOLSR

Para a realização do projeto, foram escolhidas implementações do SAODV e do SOLSR e suas respectivas versões sem segurança. O OLSR e o seu *plugin* [45] para segurança foram desenvolvidos pelo Projeto Hipercom da INRIA (*Institut National de Recherche en Informatique et en Automatique*). O A-SAODV (*Adaptative-SAODV*) [46] foi desenvolvido pela equipe do CEFRIEL (*ICT Center of Excellence For Research, Innovation,*

Tabela V.4: Relação de ataques solucionados ou não pelos protocolos SAODV e SOLSR, supondo atacantes externos.

Ataque	**SAODV**	**SOLSR**
Ataque Bizantino	Sim	Sim
Estouro da Tabela de Roteamento	Sim	Sim
Envenenamento de cache	Sim	Sim
Replicação de Pacotes	Não	Sim
Direcionamento Falso	Sim	Sim
Inundação de Hello	Sim	Sim
Ganância	Sim	Sim
Buraco Negro	Sim	Sim
Encaminhamento Seletivo	Sim	Sim
Ataque da Pressa	Sim	Sim
Túnel de Minhoca	Sim	Sim
Sybil	Sim	Sim
Identidade Falsa	Sim	Sim
Replicação de Identidade	Sim	Sim

Education and industrial Labs partnership) sobre o *kernel* do AODV-UU [47], desenvolvido pela *Uppsala University*.

A versão utilizada para os testes do OLSR é a 0.4.8 e ela deve ser compilada com o *plugin* de segurança para funcionar corretamente. Ela implementa tanto a chave de grupo quanto as estampas de tempo. Já o SAODV utilizou a versão 0.2 do A-SAODV, enquanto que o AODV-UU, a versão 0.8.1. Cabe observar que existem outras versões do AODV-UU mais recentes, mas optou-se por não utilizá-las, uma vez que o SAODV utiliza o *kernel* do AODV 0.8.1 e, desta forma, toda diferença de desempenho que for verificada pode ser creditada ao que foi introduzido na versão segura, eliminando qualquer possibilidade de alteração de desempenho por causa da alteração do *kernel*.

O A-SAODV não segue integralmente as especificações do *draft* [48]. Primeiramente, todos os nós precisam conhecer as chaves públicas de todos os outros nós antes da inicialização da rede. Além disso, foi implementado um comportamento adaptativo para as respostas dos nós intermediários. Simulações realizadas pela equipe do CEFRIEL mostraram que as respostas dos nós intermediários só são vantajosas para a rede caso elas não levem muito tempo para serem geradas. Assim, antes de responder aos RREQ, os nós intermediários pesam as suas filas e comparam esse valor com um limiar. Caso o peso da fila seja superior ao limiar, o nó encaminha o RREQ ao invés de gerar um RREP intermediário. Outro ponto que deve ser ressaltado é que essa versão do SAODV não tem suporte para *gateway*, apesar do AODV-UU possuir essa funcionalidade. Isso restringe consideravelmente a utilização desse protocolo, em especial dentro de ambientes como uma rede universitária, cujo objetivo é dar acesso a Internet. No entanto, futuras atualizações devem trazer essa funcionalidade.

Ambas as implementações de segurança são incompatíveis com suas versões sem segurança, uma vez que não aceitam nenhum tipo de mensagem que não possua a assinatura.

Maiores informações sobre como instalar e configurar esses protocolos estão disponíveis no Apêndice A.

V.4.2 Testes Realizados

Os testes realizados tiveram como objetivo analisar o impacto no desempenho causado pelos protocolos que provêem segurança. Assim, comparou-se os protocolos AODV com SAODV e OLSR com SOLSR. Uma segunda análise compara os dois protocolos seguros, avaliando impactos causados relacionados ao nível de segurança oferecido, ou seja, o custo imposto de processamento, carga de controle e memória relacionados aos ataques que o protocolo é capaz de evitar.

Como ambiente de teste, utilizou-se uma pequena rede composta por três computadores pessoais portáteis, com o sistema operacional Linux e distribuição Debian. Os portáteis foram instalados e configurados conforme descrito no Apêndice A. O objetivo

dos três portáteis era testar a qualidade do enlace ponto-a-ponto e a capacidade de roteamento dos protocolos. Assim, os computadores foram dispostos de forma a que o primeiro não estivesse no alcance do terceiro. Cabe observar que a determinação dos pontos onde esta característica era obtida dentro do laboratório do Grupo de Teleinformática e Automação (GTA), onde foram realizados os testes, não é função apenas da distância, mas principalmente dos obstáculos entre os nós. Na Figura V.8 está mostrado como foram dispostos os computadores. Entre os computadores A e B existe apenas o chão do mezanino de estrutura metálica e piso de madeira, e entre os computadores B e C existem duas paredes duplas de tijolos.

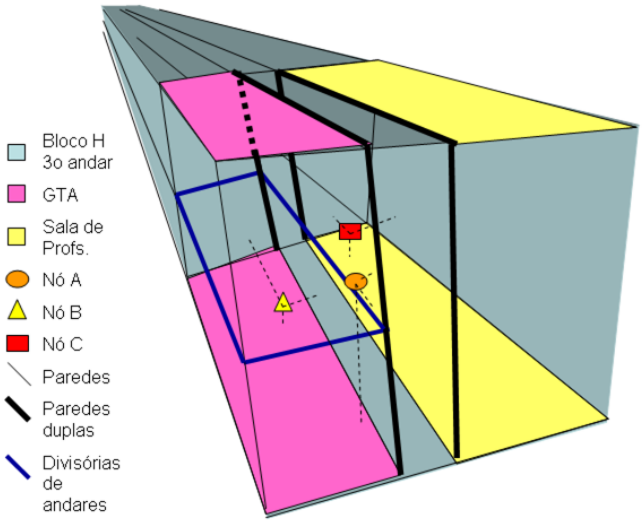

Figura V.8: Disposição dos portáteis no terceiro andar do Bloco H do CT.

As ferramentas utilizadas para as medidas na rede foram o analisador de protocolos de rede Ethereal [49], os programas de medida de desempenho Time [50] e KDE *System Guard* [51] e o programa Iperf [52].

Ethereal

O Ethereal é uma ferramenta utilizada para capturar e analisar todo o tráfego de uma interface. Ele foi utilizado em todos os nós para avaliar tanto o funcionamento do protocolo quanto o tráfego extra que é adicionado à rede. Para analisar os pacotes do SAODV, foi

necessária uma adaptação ao programa original, para que ele pudesse compreender o conteúdo dos pacotes. A nova versão do Ethereal, obtida com a compilação com o código para o SAODV, permite identificar os campos de segurança dos pacotes, enquanto que na versão original ele apenas identifica o tipo de pacote.

Iperf e os *Scripts* Desenvolvidos

O Iperf é um gerador de tráfego que mede a vazão, as perdas e a variação do atraso na entrega de rajadas de pacotes. O objetivo deste teste é avaliar o efeito causado no desempenho da rede com a carga adicionada à rede pelos protocolos de segurança, seja por atrasos devido ao processamento da criptografia ou por tráfego de mensagens de protocolo extras. Assim, foram desenvolvidos *scripts* para fazer medidas de tráfego com um e dois saltos, que estão no Apêndice B.

Para realizar os testes, os nós B e C da Figura V.8 funcionaram como servidores do Iperf para os testes de um e dois saltos, respectivamente, e o nó A funcionou como cliente em ambos os testes. Assim, os *scripts* desenvolvidos rodaram no computador A. Para cada teste, foram realizadas dez medidas em cada taxa de transmissão de dados do Iperf.

No *script*, a taxa de transmissão da camada física é ajustada em cada um dos nós e, em seguida, o Iperf inicia a transmissão de dados nas taxas de 1, 2, 5.5, até 54Mbps. Para cada taxa, são realizadas 10 medidas, para um e dois saltos.

Ferramenta KDE System Guard

O KDE *System Guard* é um programa que permite gerenciamento do funcionamento do sistema. Nos testes foi utilizado o Controlador de Processos dessa ferramenta, que possui diversos campos sobre os processos que estão sendo executados.

Para essa análise, utilizou-se apenas o nome do processo, a carga da CPU, o VmSize e o VmRSS, que são respectivamente memória virtual e memória física. Para maior acuidade, foram realizadas sete medidas e durante o teste evitou-se o uso de aplicativos, deixando

apenas os processos do sistema operacional rodando junto com o protocolo de roteamento.

Ferramenta Time

A ferramenta Time executa um programa e resume a quantidade de recursos do sistema utilizada. Após a execução do programa, ele apresenta informações sobre os recursos utilizados pelo programa testado. Dentre as medidas utilizadas para essa análise, estão o tempo total de teste, o tempo de uso da CPU pelo usuário e o tempo utilizado pelo sistema durante o teste pelo programa testado. A partir dessas informações é possível determinar o percentual de uso da CPU pelo programa, como a soma entre o tempo de uso do usuário e do sistema, divididos pelo tempo total de teste.

O uso dessa ferramenta foi necessário porque o uso de CPU dado pelo KDE *System Guard* não tem precisão suficiente, dado que o percentual de uso de CPU é muito baixo.

Para realizar o teste, deixou-se os protocolos funcionando por mais de quatorze minutos, de forma a permitir uma análise acurada do sistema. A utilização durante um tempo muito curto pode levar a um resultado incorreto.

V.4.3 Resultados dos testes

Os resultados obtidos visaram mostrar o desempenho da rede segura e o impacto causado pelas medidas de segurança. Primeiramente, são mostrados os resultados obtidos pela análise dos resultados de rede e, em seguida, os resultados obtidos em termos de desempenho de processamento e memória.

Vazão, Perdas e Variação do Atraso

Os testes utilizando o Iperf geraram resultados importantes sobre as caracterísicas da rede que está sendo utilizada. Nas Figuras V.9 e V.10 estão dispostos os resultados dos testes descritos na Seção V.4.2, mostrando a vazão na rede para um e dois saltos, nos

enlaces especificados na Figura V.8. Preferiu-se colocar o gráfico com pontos, ao invés da média, pois enlaces sem-fio possuem alta variabilidade e calcular a média muitas vezes representa perda de informações.

Neste primeiro teste, o OLSR e o SOLSR se saíram melhor, conseguindo uma vazão mais alta, atingindo picos de quase 14 Mbps. É interessante notar que, ao se observar o gráfico para dois saltos, a vazão cai quase pela metade. Isso se justifica pelas interferências do meio devido aos dois saltos, pois quando o primeiro nó transmite, o segundo não pode transmitir devido às interferências, e vice-versa, o que implica na queda de vazão do teste. De fato, como pode ser visto nas Figuras V.11 e V.12, a quantidade de perdas devido a colisões para dois saltos cresceu consideravelmente, quando comparado às perdas para apenas um salto, o que se justifica pela ausência do mecanismo RTS/CTS, que evitaria esse tipo de problema, embora reduzindo o desempenho devido à inserção de mais atrasos para cada transmissão. Nas Figuras V.13 e V.14, pode-se observar que a variação do atraso segue o mesmo comportamento da vazão e da perda, crescendo muito ao se aumentar o número de saltos.

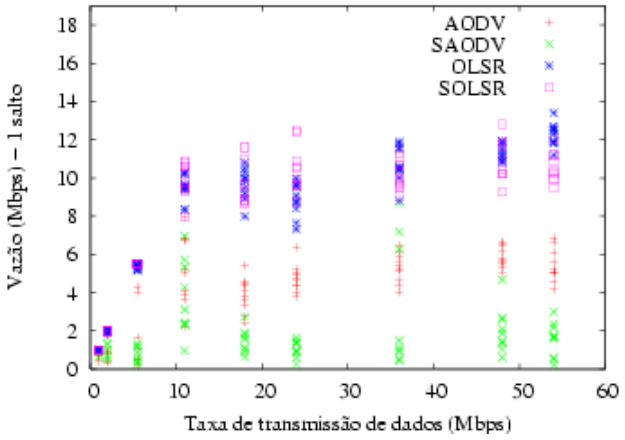

Figura V.9: Vazão na rede com um salto.

O resultado sobre qual são os melhores protocolos, com um teste apenas, é muito precipitado, pois o enlace sem-fio é muito instável. Existiram dias em que os nós B e C da Figura V.8 eram capazes de se comunicar, como visto na Figura V.10, e outros em que essa comunicação se tornava impossível. Na Figura V.15 é possível observar essa

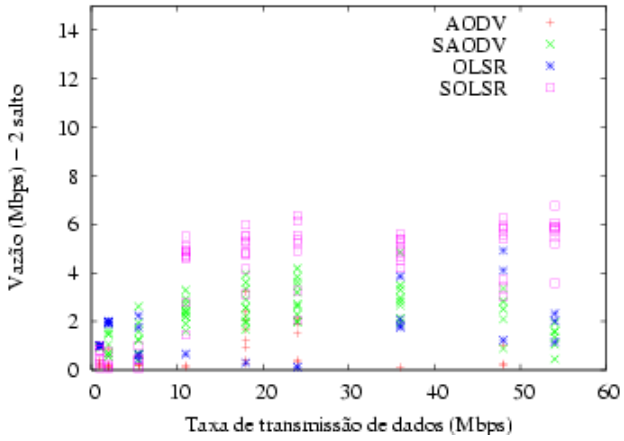

Figura V.10: Vazão na rede com dois saltos.

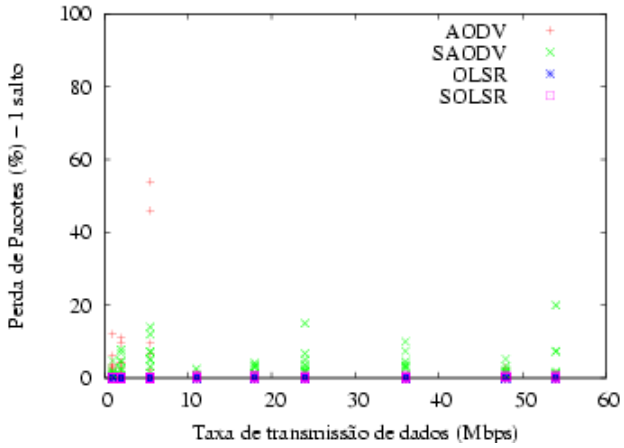

Figura V.11: Perdas na rede com um salto.

variação em cinco momentos diferentes para o protocolo SAODV. A partir desses dados, é possível dizer que apenas pela observação dos pacotes, não é possível afirmar qual o melhor protocolo no nível de rede. Nesses cinco testes também foram utilizadas dez repetições para cada medida, e as médias, com suas respectivas barras de erro estão representadas na Figura V.16.

Desta forma, se fez necessária uma avaliação mais fina sobre o funcionamento dos protocolos, para determinar o impacto da carga adicionada pela segurança. Para tanto, utilizou-se a ferramenta Ethereal, com a qual se obteve os dados apresentados na Tabela V.5. Nela estão representados o total de pacotes, o total de dados de controle do

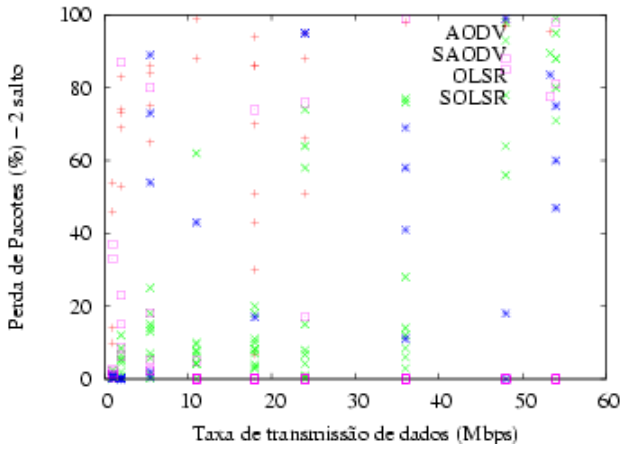

Figura V.12: Perdas na rede com dois saltos.

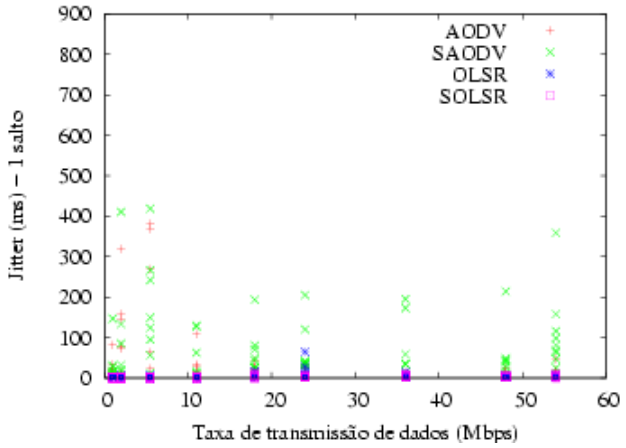

Figura V.13: Variação do atraso na rede com um salto.

protocolo de roteamento, o tempo total de captura e a taxa média de bits por segundo e de pacotes por segundo de cada protocolo. Cabe observar que os tempos de medida são diferentes e isso não interfere nos resultados, uma vez que o que está sendo comparado é a taxa média. A partir desses dados, é possível observar que o AODV e o SAODV obtiveram o pior desempenho nos casos sem e com segurança respectivamente. Isto aconteceu, apesar da pró-atividade do OLSR, porque o AODV também emite mensagens periódicas para a identificação dos vizinhos, o que representa uma carga considerável na rede. Uma vez que o intervalo padrão do AODV é de um HELLO por segundo, enquanto que o OLSR tem como padrão um HELLO para cada dois segundos, o AODV emite o dobro das mensagens de controle para um salto. Esse número, no entanto, pode ser ajustado

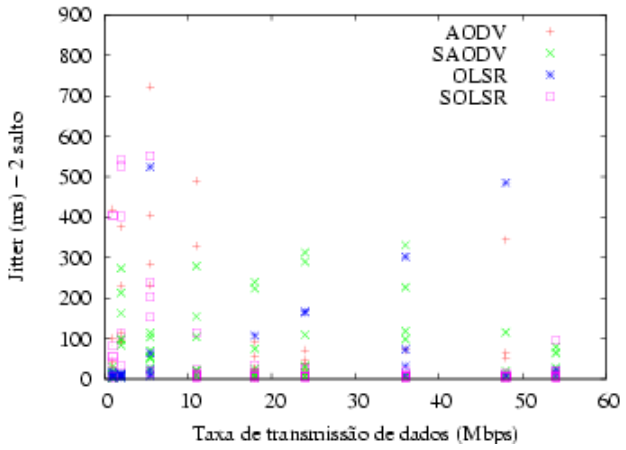

Figura V.14: Variação do atraso na rede com dois saltos.

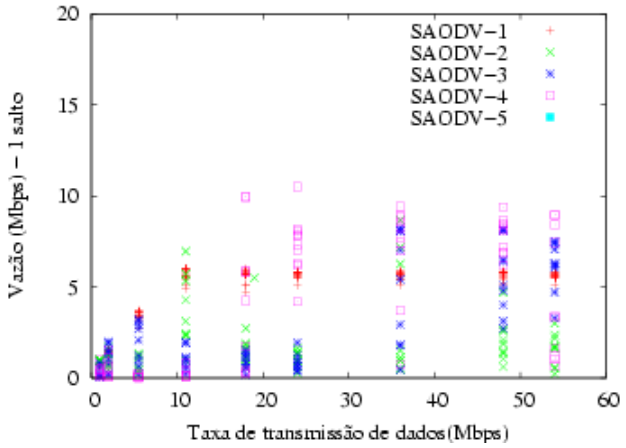

Figura V.15: Vazão na rede para o SAODV com um salto variando a taxa de dados, em cinco momentos diferentes.

por meio de configurações, o que interfere diretamente no funcionamento do protocolo, modificando sua robustez contra erros.

Uma análise mais detalhada foi realizada, mostrando o quanto pesa a segurança para o AODV em termos de carga de controle. Na Tabela V.6, pode-se notar que cada pacote, na versão segura, possui mais que o dobro do tamanho dos pacotes originais. Isto justifica o aumento de 197,31% na carga de controle na rede para o AODV ao se adicionar segurança. Cabe ressaltar que apesar de a utilização de segurança ter aumentado bastante a carga de controle da rede, ela não é grande o suficiente para representar um problema para a

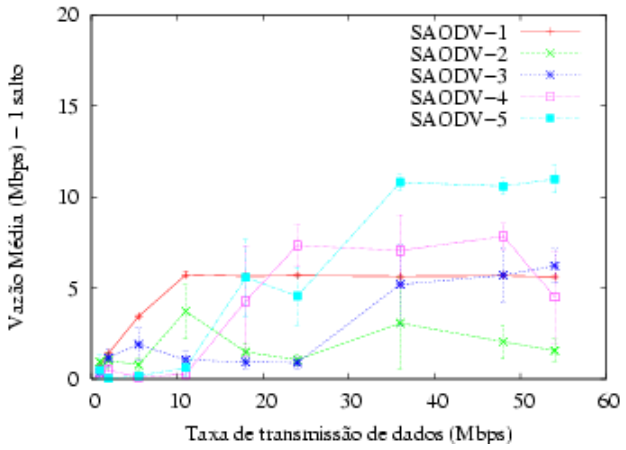

Figura V.16: Vazão média na rede para o SAODV com um salto variando a taxa de dados, em cinco dias diferentes.

Tabela V.5: Resumo dos resultados obtidos com o Ethereal para os protocolos.

Protocolo	**Total de pacotes**	**Total de dados de controle(B)**	**Tempo (s)**	**Taxa Média (bps)**	**Taxa Média (pct/s)**
AODV	145	9022	86.67	832.72	1.67
SAODV	609	109630	354.24	2475.84	1.71
OLSR	925	73718	859.08	686.48	1.07
SOLSR	196	23112	148.78	1242.8	1.31

rede. De fato, para redes que podem alcançar taxas de até 14Mbps de vazão, uma carga da ordem de $100B/s$ é insignificante.

Desempenho de Processamento e Memória

A análise com o KDE *System Guard* resultou nos dados mostrados na Tabela V.7. Para este teste, foram realizadas sete medidas de cada protocolo, e na tabela são apresentadas as respectivas médias. A análise de uso de CPU com a ferramenta *Time* gerou os resultados da Tabela V.8. Neste teste foram realizadas quatro medidas para cada proto-

Tabela V.6: Resultados obtidos com o Ethereal em 86,67s de tráfego do AODV e 354,24s do SAODV.

-	**HELLO**	**RREP**	**RREQ**	**RERR**
AODV - Tamanho (B)	62	62	66	54
AODV - Quantidade	115	7	18	5
SAODV - Tamanho (B)	178	262	254	130
SAODV - Quantidade	589	10	7	3

colo e a média está representada na tabela. Por esses resultados, é possível observar que o SAODV é muito mais custoso, em especial, em termos de memória. Para um *notebook*, este não é um nível elevado, mas para um roteador sem-fio, que dispõe de recursos bem mais escassos, com memória RAM variando entre 8 e 32MB e memória *Flash* entre 2 e 8 MB, tal consumo de memória pode representar um problema sério. Cabe observar que o OLSR não apresentou uma variação grande de uso de memória, com relação ao protocolo original, sendo possível afirmar que seu uso em roteadores não traria grandes prejuízos. Em termos de uso de CPU, todos os protocolos mostraram níveis desprezíveis de consumo de CPU, mostrando que neste aspecto, a utilização de segurança é custosa.

Tabela V.7: Resultados com a ferramenta KDE *System Guard*.

Protocolo	**Uso de CPU Médio (%)**	**Memória Virtual Média(kB)**	**Memória Física Média(kB)**
AODV	0.00	1562.86	568.00
SAODV	0.00	10591.00	1288.00
OLSR	0.00	1772.00	824.57
SOLSR	0.00	2362.29	1015.43

Tabela V.8: Resultados com a ferramenta *Time*.

Protocolo	**Tempo Total de Teste Médio**	**Tempo Médio Utilizado pelos Processos do Usuário**	**Tempo Médio Utilizado pelos Processos do Sistema**	**Uso de CPU Médio(%)**
AODV	2418.518s	0.162s	0.440s	0.029
SAODV	849.960s	0.429s	0.301s	0.081
OLSR	3713.576s	0.005s	0.007s	0.000
SOLSR	1651.253s	0.015s	0.010s	0.001

V.4.4 Conclusões

A partir da análise realizada, foi possível levantar os prós e os contras dos dois protocolos seguros. Primeiramente, pode-se observar que o SAODV apresenta uma segurança superior à do SOLSR contra os ataques mais comuns, embora o SAODV não seja capaz de proteger contra o ataque da replicação. De fato, para aplicações com a de uma rede universitária, seria interessante conhecer a identidade do usuário, pois é necessário um controle para evitar o uso malicioso da rede. A utilização de chaves de grupo, como propõe o SOLSR, só se aplica a ambientes de poucos usuários, aonde se pode depositar larga confiança em todos eles.

Com relação aos resultados de rede, o SOLSR se apresentou melhor tanto em termos de carga adicional, quanto em termos de uso de memória e CPU. Tal resultado era esperado, pois o SAODV possui uma grande complexidade, tanto devido às assinaturas digitais quanto ao uso de *hash*. De fato, alguns pacotes chegam a levar até duas assinaturas simultâneas, o que representa mais carga na rede, mais dados para serem armazenados e mais processamento a ser realizado, tanto para gerar quanto para conferir a assinatura. Apesar destas dificuldades, ainda assim ele seria mais adequado ao uso em aplicações como redes universitárias ou empresariais.

Um dos grandes problemas encontrados nesse estudo foi a ausência de métodos de distribuição de chaves para ambos os protocolos, o que exige a atuação do administrador em todos os nós sempre que algum nó quiser entrar ou sair da rede. Para resolver esse problema, seria necessário acoplar ao protocolo SAODV uma infra-estrutura de chaves-públicas e ao SOLSR um sistema de administração de chaves de grupo.

Apesar de o SAODV ter se mostrado melhor para os objetivos desejados, apesar da ausência da infra-estrutura de chaves públicas, a sua distribuição atual possui um problema grave. Até o momento, não foi desenvolvido o suporte para *gateway* no SAODV, pois o nó que tem essa função tem a responsabilidade extra de ter que assinar todos os pacotes que vem de fora como se fossem seus. Esta questão ainda não foi solucionada por apresentar um novo problema, gerado pela necessidade de algum mecanismo de reconhecimento se um determinado nó que se anuncia *gateway* o é de fato, ou se se trata apenas de um nó malicioso querendo realizar ataques. Este problema também pode ser solucionado pelo uso das autoridades certificadoras. Devido a essa restrição, que atinge diretamente o principal objetivo da rede universitária sem-fio, que é o acesso à Internet, optou-se pelo uso do SOLSR, a princípio, como solução para rede segura ad hoc.

O que se observa de todos esses resultados é que a segurança em redes ad hoc é um campo que ainda está em desenvolvimento, e que ainda precisa aprimorar muitos de seus instrumentos, tanto para dar maior proteção contra ataques, como para diminuir o consumo de recursos dos nós e da rede. Assim, muitos estudos ainda estão sendo realizados dentro desta área, o que deverá gerar novas soluções interessantes dentro dos próximos anos.

Referências Bibliográficas

[1] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum e L. Viennot, "Optimized link state routing protocol for ad hoc networks", *5th IEEE Multi Topic Conference (INMIC 2001)*, pp. 62–68, dezembro de 2001.

[2] C. Huitema, *Routing in the Internet*. Prentice Hall, 2ª ed., 1999.

[3] C. E. Perkins, E. M.Belding-Royer e R. S. Das, *Ad Hoc On-Demand Distance Vector Routing*. Request for Comments: 3561, julho de 2003.

[4] T. Clausen e P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*. RFC 3626, outubro de 2003.

[5] European Telecommunications Standards Institute, *ETSI Home Page*. http://www.etsi.org/, acessado em novembro de 2006.

[6] C. Murthy e B. Mano, *Ad Hoc wireless networks: architetures and protocols*. Prentice Halll Professional Technical Reference, 2004.

[7] Wikipedia - The Free Encyclopedia, *Eaves*. http://en.wikipedia.org/wiki/Eaves, acessado em julho de 2006.

[8] A. Wood e J. Stankovic, "Denial of service in sensor networks", *Computer*, vol. 35, no. 10, pp. 54–62, outubro de 2002.

[9] L. Lamport, R. Shostak e M. Pease, "The byzantine generals problem", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, julho de 1982.

[10] Y.-C. Hu, A. Perrig e D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols", *Second ACM Workshop on Wireless Security (WiSe 03)*, pp. 30–40, setembro de 2003.

[11] Y.-C. Hu, A. Perrig e D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks", *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 3, pp. 1976–1986, abril de 2003.

[12] J. R. Douceur, "The sybil attack", *First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 251–260, março de 2002.

[13] J. Newsome, E. Shi, D. Song e A. Perrig, "The sybil attack in sensor networks: Analysis & defenses", *3rd IEEE/ACM Information Processing in Sensor Networks 2004 - IPSN 04*, pp. 259–268, abril de 2004.

[14] H. Chan, A. Perrig e D. Song, "Random key predistribution schemes for sensor networks", *IEEE Symposium on Security and Privacy*, pp. 197–213, maio de 2003.

[15] S. Yi, P. Naldurg e R. Kravets, "Security-aware ad hoc routing for wireless networks", *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, Long Beach, CA, pp. 299–302, outubro de 2001.

[16] R. Anderson e M. Kuhn, "Tamper resistance - a cautionary note", *Second USENIX Workshop en Electronic Commerce*, pp. 1–11, novembro de 1996.

[17] L. Buttyan e J. P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks", *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, pp. 579–592, outubro de 2003.

[18] Z. Ye, S. V. Krishnamurthy e S. K. Tripathi, "A framework for reliable routing in mobile ad hoc networks", *INFOCOM 2003. IEEE Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 270–280, março de 2003.

[19] D. Ganesan, R. Govindan, S. Shenker e D. Estrin, "Highly resilient, energy-efficient multipath routing in wireless sensor networks", *ACM Mobile Computing and Communications Review*, vol. 4, no. 5, pp. 11–25, outubro de 2001.

[20] C. Karlof e D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures", *IEEE International Workshop on Sensor Network Protocols and Applications 2003*, pp. 113–127, maio de 2003.

[21] National Bureau of Standards, *Data Encryption Standard*. FIPS-Pub.46, janeiro de 1977.

[22] J. Daemen e V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[23] B. Kaliski e J. Staddon, *PKCS #1: RSA Cryptography Specifications Version 2.0*. RFC 2437, outubro de 1998.

[24] E. Rescorla, *Diffie-Hellman Key Agreement Method*. RFC 2631, junho de 1999.

[25] I. 1363, *IEEE Standard Specifications for Public-Key Cryptography*. IEEE Std 1363-2000, agosto de 2000.

[26] W. Diffie e M. E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, novembro de 1976.

[27] R. L. Rivest, *The MD5 Message-Digest Algorithm*. RFC 1321, abril de 1992.

[28] National Institute of Standards, *Secure hash standard*. FIPS 180-2, agosto de 2000.

[29] S. S. Kulkarnia, M. G. Goudab e A. Arora, "Secret instantiation in ad-hoc networks", *Computer Communications*, vol. 2, no. 29, pp. 200–215, julho de 2006.

[30] R. K. Nichols e P. C. Lekkas, *Wireless Security Models, Threats,and Solutions*. McGraw-Hill, 2002.

[31] J. Kong, P. Zerfos, H. Luo, S. Lu e L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks", *Ninth International Conference on Network Protocols (ICNP'01)*, pp. 251–260, novembro de 2001.

[32] A. Shamir, "How to share a secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, novembro de 1979.

[33] G. R. Blakley, "Safeguarding cryptographic keys", *National Computer Conference (AFIPS)*, vol. 48, pp. 313–317, junho de 1979.

[34] C. Gahlin, "Secure ad hoc networking", Tese de Mestrado, University of Umeå, março de 2004.

[35] Y. Frankel e Y. Desmedt, "Parallel reliable threshold multisignature", Relatório Técnico, University of Wisconsin, 1992.

[36] L. Zhou e Z. J. Haas, "Securing ad hoc networks", *IEEE Network*, vol. 13, no. 6, pp. 24–30, novembro de 1999.

[37] R. Gennaro, S. Jarecki, H. Krawczyk e T. Rabin, "Robust threshold DSS signatures", *Advances in Cryptology - Eurocrypt '96*, pp. 354–371, maio de 1996.

[38] R. Atkinson, *Security architecture for the internet protocol.* RFC 1825, agosto de 1995.

[39] K. Sanzgiri, B. Dahill, B. N. Levine e E. M. Belding-Royer, "A secure routing protocol for ad hoc networks", *10th IEEE International Conference on Network Protocols*, pp. 78–87, novembro de 2002.

[40] Y.-C. Hu, A. Perrig e D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", *Wireless Networks*, vol. 11, no. 1–2, pp. 21–38, janeiro de 2005.

[41] P. Papadimitratos e Z. Haas, "Secure routing for mobile ad hoc networks", *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, pp. 27–31, janeiro de 2002.

[42] M. G. Zapata, "Secure ad hoc on-demand distance vector (SAODV) routing", *ACM Mobile Computing and Communications Review*, vol. 6, no. 3, pp. 106–107, julho de 2002.

[43] A. Hafslund, A. Tønnesen, R. B. Rotvik, J. Andersson e Øivind Kure, "Secure extension to the olsr protocol", *OLSR Interop and Workshop*, San Diego, California, pp. 1–4, agosto de 2004.

[44] A. Tønnesen, "Implementing and extending the optimized link state routing protocol", Tese de Mestrado, University of Oslo, agosto de 2004.

[45] Andreas Tønnesen, *OLSR.ORG*. http://www.olsr.org, acessado em novembro de 2006.

[46] ICT Center of Excellence For Research, Innovation, Education and industrial Labs partnership (CEFRIEL), *A-SAODV Homepage*. http://saodv.cefriel.it/, acessado em novembro de 2006.

[47] Erik Nordström, *Ad-hoc On-demand Distance Vector Routing - For real world and simulation*. http://core.it.uu.se/core/index.php/AODV-UU, acessado em novembro de 2006.

[48] M. G. Zapata, *Secure Ad hoc On-Demand Distance Vector (SAODV) Routing*. Internet Draft 6, setembro de 2006.

[49] Ethereal, Inc., *Ethereal - The world's most popular network protocol analyzer*. http://www.ethereal.com/, acessado em novembro de 2006.

[50] David MacKenzie and Dirk Eddelbuettel, *Manpages*. http://man.cx/time(1), acessado em novembro de 2006.

[51] Chris Schlaeger, *The K Desktop Environment*. http://www.kde.gr.jp/help/doc/kdebase/doc/ksysguard/HTML/index.html, acessado em novembro de 2006.

[52] Ajay Tirumala and Feng Qin and Jon Dugan and Jim Ferguson and Kevin Gibbs, *NLANR - Distributed Applications Support Team - Iperf - Testing the limits of your network*. http://dast.nlanr.net/Projects/Iperf, acessado em novembro de 2006.

Apêndice A

Instalação e Configuração dos Protocolos AODV, OLSR, SAODV e SOLSR

A.1 Instalação e Configuração

Os códigos dos quatro protocolos testados, AODV, SAODV, OLSR e SOLSR foram obtidos na Internet e instalados em computadores portáteis, doravante chamados de portáteis, para a realização dos testes de desempenho. Os portáteis usam o sistema operacional Linux com *kernel* 2.6.8-2-386 e distribuição Debian. Portanto, as instruções para instalação e dificuldades encontradas são relativas a essa distribuição do sistema operacional. A seguir, estão dispostos os passos necessários para a instalação e configuração dos protocolos.

A.1.1 AODV

O AODV-UU 0.8.1 tem como requisitos a utilização de Linux com *kernel* 2.4.x ou 2.6.x e suporte a *Netfilter*, o que é um requisito atendido pela maioria das distribuições.

Além disso, o compilador deve ser o gcc-3.3.

Para compilar o protocolo é necessário que os arquivos de cabeçalho do *kernel* estejam instalados, pois, caso contrário, os módulos do *kernel* podem apresentar erros de compilação.

Para realizar a instalação, após baixar o código e descompactá-lo, deve-se seguir os comandos típicos de instalação do Linux. Assim, com direitos de super-usuário, os seguintes comandos devem ser executados:

> make CC=gcc-3.3

> make install

É necessário incluir o *tag* CC para determinar o compilador, caso contrário, o código não compila corretamente. Com isso, todo o código é compilado e o protocolo AODV é instalado. Para utilizar o protocolo, deve-se executar o comando:

> aodvd

Ao executar esse comando, o módulo do kaodv.ko é carregado automaticamente no sistema operacional. Erros podem ocorrer caso o AODV não tenha sido instalado corretamente, ou ainda se o carregador de módulos do sistema não estiver devidamente configurado.

A.1.2 SAODV

Os requisitos de instalação do SAODV são semelhantes ao do AODV. . Assim, ele também opera apenas para os *kernels* do Linux versões 2.4.x e 2.6.x com *Netfilter*. Além disso, também é necessário instalar as bibliotecas Libgcrypt 1.2 e Libgpg-error 1.0, com seus respectivos cabeçalhos. Outro requisito é o uso do compilador gcc na sua versão 3.3, pois, caso contrário, a compilação não é completada. Para compilar o SAODV, utiliza-se os mesmos comandos do AODV, executados com permissão de super-usuário. Assim:

> apt-get install libgcrypt11-dev

> make CC=gcc-3.3

> make install

Dependendo da versão do *kernel* pode ocorrer um erro de compilação com o código obtido pela Internet. Esse erro é de simples correção, bastando remover o símbolo '&' do terceiro argumento da linha 202 do arquivo kaodv.c. Após essa modificação, o código compila normalmente.

Outro erro encontrado durante a instalação do protocolo no ambiente de testes foi relativo a algumas configurações do Linux. Para solucionar este problema, foi necessário trocar a linha MODULE_ARCH_VERMAGIC do arquivo /usr/src/kernel-headers-2.6.8-2-386/include/linux/vermagic.h por "386 ". Após essa modificação foi possível compilar e instalar o A-SAODV corretamente. Após esses passos, o SAODV é instalado. No entanto, é necessário ainda que as chaves púbicas e privadas de todos os nós sejam criadas e que cada nó possua armazenadas as chaves públicas em todos os outros nós. No sítio [46] também é possível obter o código de um gerador de chaves RSA, chamado saodv-genkey, que fornece dois arquivos como saída no formato necessário para o SAODV. Para utilizá-lo, basta obter o código, descompactá-lo e executar o seguinte comando:

>saodvd-genkey

Esse comando gera os arquivos private.key.tmp e public.keys.tmp. A chave privada deve ser armazenada em /etc/saodvd/private.key, e a chave pública é guardada em todos os nós num arquivo contendo todas as chaves públicas da rede, que fica em /etc/saodvd/public.keys. Para realizar a primeira operação basta executar os comandos:

> cp ./private.key.tmp /etc/saodvd/private.key

> rm ./private.key.tmp

A chave pública deve ser transferida para todos os nós, através de algum comando como o sftp ou ainda através de alguma mídia, e com algum editor, ela deve ser inserida no arquivo public.keys. O arquivo final deve conter uma linha por nó da rede, tal como:

IP rsa modulo_em_código_hexadecimal expoente_em_código_hexadecimal

Por exemplo:

192.168.0.23 rsa 00DCC889464E6CFCA122DFF850513A2FE3821C6E62C98B1660A33 C85D7474013419DDBEB31EA29FD9981568E35C7DC2CE15D11388C6E8E6966ACBB93 A8B0AF1C7B 010001

O tamanho de chave máximo aceito nessa versão do SAODV é de 512 bits.

Para inicializar o SAODV deve-se executar o comando:

> saodvd

A.1.3 OLSR e o *plugin* SOLSR

A instalação do OLSR é mais simples que a do SAODV. Para realizá-la, deve-se executar os comandos:

>make OS=linux

> make install

Para instalar o *plugin*, que é, de fato, uma biblioteca dinâmica, deve-se acessar o diretório /lib/secure do código do OLSR executar os comandos a seguir. Antes de compilar o código do *plugin*, é necessário instalar a biblioteca openssl. Assim:

> apt-get install libssl-dev

> make

> make install

Após esses passos, é necessário editar o arquivo de configuração do OLSR, indicando o *plugin* e o arquivo com a chave de grupo. Para armazenar a chave de grupo, cria-seo diretório /root/.olsr e dentro dele o arquivo olsrd_secure_key, no qual se insere a chave de 128 bits, o que corresponde a 16 caracteres. No arquivo de configuração é inserida a

chamada para o *plugin*, com uma variável indicando o caminho onde a chave está armazenada. Além disso, são inseridas informações sobre a rede para o correto funcionamento do OLSR. O arquivo de configuração, que fica armazenado em /etc/olsr.conf, é do tipo:

Debug level(0-9)

If set to 0 the daemon runs in the background

DebugLevel 0

IP version to use (4 or 6)

IpVersion 4

Clear the screen each time the internal state changes

ClearScreen yes

Hna4

{

Internet gateway:

0.0.0.0 0.0.0.0

more entries can be added:

192.168.1.0 255.255.255.0

}

AllowNoInt yes

IpcConnect

{

Determines how many simultaneously IPC connections that will be allowed Setting this to 0 disables IPC

MaxConnections 30

By default only 127.0.0.1 is allowed to connect. Here allowed hosts can be added

Host 127.0.0.1

You can also specify entire net-ranges that are allowed to connect. Multiple entries are allowed

Net 192.168.0.0 255.255.255.0

}

Whether to use hysteresis or not Hysteresis adds more robustness to the link sensing but delays neighbor registration. Used by default. 'yes' or 'no'

UseHysteresis no

Hysteresis parameters Do not alter these unless you know what you are doing! Set to auto by default. Allowed values are floating point values in the interval 0,1 THR_LOW must always be lower than THR_HIGH.

HystScaling 0.05

HystThrHigh 0.30

HystThrLow 0.001

LinkQualityLevel 2

LinkQualityWinSize 10

Pollrate 0.05

#Plugin de segurança Ű Deve ser comentado para executar o OLSR sem segurança

LoadPlugin "olsrd_secure.so.0.4"

{

PlParam "Keyfile/root/.olsr/olsrd_secure_key"

}

#A interface deve ser configurada de acordo com a interface sem-fio utilizada(eth1,wlan0,...)

Interface "eth1"

{

Hello interval in seconds(float)

HelloInterval 2.0

HELLO validity time

HelloValidityTime 20.0

}

Para executar o OLSR sem segurança basta comentar as linhas correspondentes ao *plugin* no arquivo de configuração. Para executar ambas as versões, basta executar o comando:

>olsrd

Apêndice B

Scripts utilizados nos testes com o Iperf

O desempenho de cada um dos protocolos é avaliado, através do programa Iperf, para as taxas de transmissão da camada física iguais a 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48 e 54 Mbps e iguais taxas de envio de mensagens do Iperf. Para cada taxa são realizadas 10 medidas tanto no cenário de um salto como no de dois saltos. Para obter estas medidas, desenvolveu-se um *script* para ser executado no cliente do Iperf para cada protocolo como o disposto a seguir. Nele, primeiramente, são criados os diretórios que armazenam os resultados para um e dois saltos. Em seguida, para cada uma das taxas, se configura a taxa da camada física com um valor igual nos três nós de teste e se emite os dados.

```
#!/bin/sh

mkdir solsr

mkdir solsr/results_2saltos

mkdir solsr/results_1salto

for rate in 1 2 5.5 6 9 11 12 18 24 36 48 54; do

iwconfig eth1 rate $rateM

ssh 192.168.0.13 iwconfig wlan0 rate $rateM
```

```
ssh 192.168.0.24 iwconfig eth1 rate $rateM
touch solsr/results_2saltos/saida.$rate
for round in 1 2 3 4 5 6 7 8 9 10; do
echo "1o Comecando taxa"$rate "rodada"$round "olsr"
iperf -c 192.168.0.13 -b $rateM | grep
sleep 1
done
done
for rate in 1 2 5.5 6 9 11 12 18 24 36 48 54; do
iwconfig eth1 rate $rateM
ssh 192.168.0.24 iwconfig eth1 rate $rateM
touch solsr/results_1salto/saida.$rate
for round in 1 2 3 4 5 6 7 8 9 10; do
echo "2o Comecando taxa"$rate "rodada"$round "solsr"
iperf -c 192.168.0.24 -b $rateM | grep
sleep 1
done
done
```

Nos nós servidores, é executado o comando abaixo, no qual o -s significa modo servidor e o -u, tráfego UDP:

```
>iperf -s -u
```

Para realizar o ssh sem digitar a senha no *script* do cliente, o que representaria uma

interação com o usuário, executa-se os seguintes comandos no nó cliente antes do início dos testes:

ssh-keygen -t rsa

scp /.ssh/id_rsa.pub root@192.168.0.22:/tmp

scp /.ssh/id_rsa.pub root@192.168.0.23:/tmp

Primeiramente se criou o par de chaves, e, em seguida, com o scp, transferiu-se a chave pública do cliente para os dois servidores do Iperf. Após isso, nos servidores, foram executados comandos para transcrever a chave pública do cliente para o arquivo de chaves autorizadas para ssh dos servidores:

cat /tmp/id_*.pub » /root/.ssh/authorized_keys

chmod 600 /root/.ssh/authorized_keys

A mudança de autorização com o comando chmod é necessária para restringir o acesso ao arquivo de chaves públicas.