Clytia Higa Tamashiro

Uma Análise de Protocolos de Roteamento Anônimo para Redes Sem Fio Ad Hoc Móveis

Florianópolis – SC

2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Clytia Higa Tamashiro

UMA ANÁLISE DE PROTOCOLOS DE ROTEAMENTO ANÔNIMO PARA REDES SEM FIO AD HOC MÓVEIS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. João Bosco Mangueira Sobral

Florianópolis, setembro de 2007

UMA ANÁLISE DE PROTOCOLOS DE ROTEAMENTO ANÔNIMO PARA REDES SEM FIO AD HOC MÓVEIS

Clytia Higa Tamashiro

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação na Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Rogério Cid Bastos
Coordenador do PPGCC

Banca Examinadora

Prof. Dr. João Bosco Mangueira Sobral
Orientador

Prof. Dr. Joni da Silva Fraga

Prof. Dr. Mário Antônio Ribeiro Dantas

Prof. Dr. Ricardo Felipe Custódio

Dedico a meus queridos pais.

Agradecimentos

Primeiramente, agradeço a Deus por guiar meus passos, abençoar minha vida e me fortalecer nos momentos de tristeza.

Agradeço aos meus queridos pais, Lena e Maurício, por estarem sempre presentes e me amarem com tanto carinho e dedicação.

Agradeço ao meu irmão, Márcio, à minha cunhada, Ludi, e ao meu sobrinho, Luís Felipe, por me mostrarem a importância da união e da perseverança.

Agradeço aos meus familiares por todas as orações e palavras de apoio e de incentivo.

Agradeço ao meu orientador, professor Bosco, pelo apoio e confiança na realização desta dissertação.

Agradeço ao Programa de Pós-Graduação em Ciência da Computação e à Universidade Federal de Santa Catarina por me permitirem desenvolver este trabalho.

Agradeço à CAPES pelo apoio financeiro durante o desenvolvimento da pesquisa.

Agradeço aos membros da banca pela atenção despendida na correção e avaliação deste trabalho.

Agradeço a todos os meus amigos pela amizade verdadeira e, principalmente, pelas conversas e mensagens de incentivo.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para que eu vencesse esta importante etapa de minha vida.

"Tudo posso naquele que me torna forte."

(Filipenses 4,13)

Resumo

Redes sem fio ad hoc móveis são mais vulneráveis a ataques passivos, como monitoramento e análise de tráfego, do que redes estruturadas, principalmente, devido a características como ausência de infra-estrutura fixa e meio sem fio compartilhado. A fim de garantir a privacidade dos nós e tentar impedir a revelação de informações relevantes através de pacotes de roteamento, protocolos de roteamento anônimo foram propostos. Porém, a maioria das pesquisas tem como enfoque a análise dos desempenhos, analisa as propostas com base em diferentes propriedades e definições de anonimato, e não as especifica formalmente. Assim, esta dissertação tem como objetivo indicar, com base em uma única definição, o protocolo de roteamento anônimo proposto para redes sem fio ad hoc móveis mais seguro quanto a anonimato e verificá-lo formalmente. Nove protocolos, ANODR, SDAR, ASR, ANDSR, MASK, ASRP, AnonDSR, CARP e ODAR, são analisados detalhadamente e comparados em relação à vulnerabilidade a ataques de análise de tráfego e garantia de propriedades como anonimato da identidade, do *venue* e da rota, e privacidade da localização e do padrão de movimento. Os resultados da análise indicam que nenhum protocolo provê todas as propriedades de anonimato desejadas e que o protocolo ANODR é o mais seguro. Este é especificado em LOTOS e sua verificação formal demonstra que funciona corretamente e que não garante anonimato da identidade do destino e do *venue* da origem em relação a todos os nós, porém modificações simples podem melhorar sua segurança.

Palavras-chave: Redes Ad Hoc. Roteamento. Anonimato.

Abstract

Mobile ad hoc wireless networks are more susceptible to passive attacks, such as eavesdropping and traffic analysis, than wired networks, mainly, due to characteristics as absence of fixed infrastructure and shared wireless medium. In order to guarantee the privacy of the nodes and to thwart the disclosing of relevant information through the routing packets, anonymous routing protocols have been proposed. However, most of the current researches on those protocols has focused the analysis of their performance and analyzed them based on different anonymity aspects and definitions; moreover, there has not been concern about formal specification. Thus, this work aims to indicate, considering the same definition, the more secure anonymous routing protocol and to verify it formally. Nine protocols, ANODR, SDAR, ASR, ANDSR, MASK, ASRP, AnonDSR, CARP and ODAR, are analyzed and compared in relation to vulnerability to traffic analysis attacks and guarantee of properties such as anonymity of identity, venue and route, and privacy of location and motion pattern. The results of our analysis indicate that no protocol provides all the desired anonymity properties and the protocol ANODR is the more secure. It is specified in LOTOS and its formal verification demonstrates that it functions correctly and it does not guarantee destination identity and source venue anonymity in relation to all the nodes, however simple modifications can improve its security.

Keywords: Ad Hoc Networks. Routing. Anonymity.

Sumário

Lista de Ilustrações

Lista de Quadros

Lista de Acrônimos

ACT ONE – *Abstract Data Type Formalism*
ADO – *Anonymous Communication Data Onion*
ALOHA – *Areal Locations of Hazardous Atmospheres*
ANDSR – *Anonymous Dynamic Source Routing*
ANODR – *ANonymous On Demand Routing*
AnonDSR – *Anonymous Dynamic Source Routing Protocol*
AODV – *Ad hoc On-Demand Distance Vector*
ASR – *Anonymous Secure Routing*
ASRP – *Anonymous Secure Routing Protocol*
BRAN – *Broad Band Radio Access Networks*
CADP – *CAESAR/ALDEBARAN Development Package*
CARP – *Certificate-free Anonymous Routing Protocol*
CCS – *Calculus of Communicating System*
CSMA – *Carrier Sense Multiple Access*
CSMA/CA – *Carrier Sense Multiple Access with Collision Avoidance*
CSP – *Communicating Sequential Processes*
DCF – *Distributed Coordination Function*
DSDV – *Destination Sequenced Distance-Vector Routing Protocol*
DSR – *Dynamic Source Routing*
DSSS – *Direct-Sequence Spread Spectrum*
FHSS – *Frequency Hopping Spread Spectrum*
GPS – *Global Positioning System*
HIPERACCESS – *HIgh PErformance Radio Access*
HIPERLAN – *HIgh PErformance Radio Local Area Network*
HIPERMAN – *HIgh PErformance Radio Metropolitan Area Network*
HR-DSSS – *High Rate Direct Sequence Spread Spectrum*
HMAC – *Hashed Message Authentication Code*
HTLCK – *High Trust Level Community Key*

IBE – *Identity Based Encryption*

GloMo – *Global Mobile Information System*

ISM – *Industrial, Scientific, Medical*

LOTOS – *Language of Temporal Ordering Specification*

LPR – *Low-cost Packet Radio*

MAC – *Medium Access Control*

MANET – *Mobile Ad hoc NETworks*

MTLCK – *Medium Trust Level Community Key*

nonce – *number used once*

ODAR – *On-Demand Anonymous Routing*

OFDM – *Orthogonal Frequency Division Multiplexing*

OLSR – *Optimized Link State Routing Protocol*

PCF – *Point Coordination Function*

PDO – *Protected Path Discovery Onion*

PRO – *Path Reverse Onion*

PRNet – *Packet Radio Network*

RFC – *Request For Comments*

RREP – *Route Reply*

RREQ – *Route Request*

RERR – *Route Error*

SDAR – *Secure Distributed Anonymous Routing*

SIG – *Special Interest Group*

SRP – *Secure Routing Protocol*

SURAN – *Survivable Radio Networks*

TBRPF – *Topology Dissemination Based on Reverse-Path Forwarding*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

Wi-Fi – *Wireless-Fidelity*

WLAN – *Wireless Local Area Network*

WPAN – *Wireless Personal Area Network*

WWAN – *Wireless Wide Area Network*

1 *Introdução*

Redes sem fio ad hoc móveis são redes constituídas por dispositivos sem fio móveis, chamados nós, que se comunicam diretamente ou através uns dos outros, sem o gerenciamento centralizado de qualquer infra-estrutura. Outras características relevantes são: topologia dinâmica, roteamento distribuído, recursos computacionais e segurança física limitados.

Devido a essas características e, principalmente, ao meio sem fio compartilhado, tais redes são vulneráveis a ataques de análise de tráfego, que podem comprometer a privacidade dos nós e revelar informações importantes e comprometedoras sobre a rede e seus usuários. Com o objetivo de impedir ou, pelo menos, dificultar a obtenção e inferência de informações através da captura e da análise de pacotes de roteamento, pesquisadores propuseram diversos protocolos de roteamento anônimo.

KONG & HONG (2003) apresentam o primeiro protocolo de roteamento anônimo desenvolvido para redes sem fio ad hoc móveis, ANODR - *ANonymous On Demand Routing*. É constituído por três fases: descoberta de rota, manutenção e encaminhamento dos dados. Utiliza mecanismos como: difusão, pseudônimos dinâmicos, criptografia salto a salto da carga útil dos dados (nós consecutivos compartilham uma chave; quando um nó recebe um pacote, decifra os dados com a chave compartilhada com o emissor, e os cifra novamente com a chave compartilhada com o próximo nó na rota), técnicas *mixing* (processamento dos pacotes em ordem aleatória e envio de pacotes falsos), pacotes de requisição e de resposta com tamanho único e fixo (intermediários acrescentam *paddings* para esconder o tamanho real) e *trap-door* (mecanismo através do qual um nó sabe que o pacote é destinado a ele, mas outros nós não; uma informação é cifrada de modo que somente o destino é capaz de decifrá-la corretamente) de chaves simétricas.

EL-KHATIB *et al.* (2003) e BOUKERCHE *et al.* (2004a, 2004b, 2004c) propõem

o protocolo SDAR - *Secure Distributed Anonymous Routing*. Provê um sistema de gerenciamento de confiança entre os nós e é constituído por três fases: descoberta de rota, rota reversa e transferência dos dados. Utiliza mecanismos como: difusão, *Onion Routing* (mecanismo através do qual há a geração de uma estrutura formada por várias camadas de criptografia; cada camada corresponde a um nó da rota), nós finais adicionam *paddings* de tamanho aleatório aos pacotes de requisição e de resposta e *trapdoor* de chaves assimétricas.

ZHU *et al.* (2004) descreve o protocolo ASR - *Anonymous Secure Routing*. É constituído por quatro fases: requisição, resposta, transmissão e manutenção. Utiliza mecanismos como: difusão, criptografia salto a salto da carga útil dos dados, técnicas *mixing*, pacotes de resposta com tamanho único e fixo e *trapdoor* de chaves simétricas.

ZHANG *et al.* (2004, 2005) propõe o protocolo MASK. Possui fases de pré-configuração e de autenticação de vizinhos, nas quais pseudônimos, chaves secretas e identificadores de enlace são estabelecidos, de requisição, de resposta e de encaminhamento dos dados. Utiliza mecanismos como: difusão, identificadores de enlace dinâmicos, criptografia salto a salto da carga útil dos dados, pacotes de requisição e de resposta com tamanho único e fixo, pseudônimos e técnicas *mixing*.

ARAUJO (2005) apresenta o protocolo ANDSR - *Anonymous Dynamic Source Routing*. É uma extensão do protocolo DSR - *Dynamic Source Routing* (JOHNSON; HU; MALTZ, 2007) e do trabalho de JIANG *et al.* (2001, 2004). O algoritmo de descoberta de nós misturadores é acrescentado ao processo de descoberta de rota do DSR. É constituído por seis fases: requisição e resposta do misturador, requisição e resposta da rota, transferência e manutenção. Utiliza mecanismos como: difusão e misturadores. Apresenta especificação e verificação formal em LOTOS - *Language of Temporal Ordering Specification*.

BANERJEE *et al.* (2006) propõe o protocolo CARP - *Certificate-free Anonymous Routing Protocol*. É constituído por quatro fases: descoberta e resposta de rota, transferência e manutenção. Utiliza IBE - *Identity Based Encryption* (a identidade de um nó é utilizada como sua chave pública e a chave privada correspondente é gerada por um servidor de chaves), difusão, pseudônimos e *trapdoor* de chaves simétricas.

CHENG & AGRAWAL (2005) apresentam o protocolo ASRP - *Anonymous*

Secure Routing Protocol. É constituído por três fases: requisição, resposta e transmissão. Utiliza mecanismos como: difusão, criptografia salto a salto da carga útil dos dados, requisição e resposta com tamanho único e fixo, pseudônimos e *trapdoor* de chaves assimétricas.

SONG *et al.* (2005) descreve o protocolo AnonDSR - *Anonymous Dynamic Source Routing Protocol*. Provê um mecanismo de estabelecimento de parâmetros de segurança entre origem e destino e é constituído por três fases: requisição, resposta e transferência. Utiliza mecanismos como: difusão, técnicas *mixing*, *Onion Routing*, pseudônimos, *paddings* de tamanho aleatório e *trapdoor* de chaves simétricas.

SY *et al.* (2006) propõe o protocolo ODAR - *On-Demand Anonymous Routing*. A geração de chaves secretas entre origens e destinos é realizada através do algoritmo *Diffie-Hellman* e com o auxílio de um servidor de chaves. É constituído por quatro fases: solicitação do valor público *Diffie-Hellman* do destino, requisição, resposta e transferência. Utiliza mecanismos como: *Bloom Filter* (mecanismo no qual há k funções hash e um vetor com m posições inicialmente com o valor 0; essas funções são aplicadas à informação a ser inserida no vetor, para descobrir as posições que devem ser modificadas para 1), difusão, técnicas *mixing*, requisição e resposta com tamanho único e fixo, e *trapdoor* de HMAC - *Hashed Message Authentication Code*.

Porém, tais propostas são descritas de forma textual, têm como enfoque o desempenho e baseiam-se em diferentes definições e propriedades de anonimato.

1.1 Objetivos

O objetivo principal desta dissertação é indicar, com base em uma única definição, o protocolo de roteamento anônimo proposto para redes sem fio ad hoc móveis mais seguro quanto a anonimato e verificá-lo formalmente. Os objetivos específicos são: definir anonimato no contexto de protocolos de roteamento, comparar e especificar formalmente um dos protocolos.

1.2 Metodologia

Para a realização desta dissertação, adotou-se a seguinte metodologia:

- Pesquisar definições e propriedades de anonimato existentes;
- Definir as propriedades a serem analisadas e os tipos de adversários;
- Enumerar as questões pertinentes a cada propriedade;
- Compreender e analisar detalhadamente os protocolos quanto a essas propriedades;
- Especificar e verificar formalmente utilizando a técnica de descrição formal LOTOS (IS8807, 1988) e o pacote de ferramentas CADP (VASY-INRIA, 2007).

1.3 Contribuições e Limitações

As principais contribuições são: descrição detalhada dos protocolos, comparação com base em uma única definição, especificação formal e publicação de um artigo, "*Towards an Architecture for Self-Organization in Wireless Mesh Networks*", na 33^{rd} *Latin-American Conference on Informatics*. Nesse artigo, é proposta uma arquitetura para auto-organização em redes em malha sem fio, na qual uma das camadas baseia-se no estudo sobre anonimato realizado nesta dissertação. As limitações são: a análise não é quantitativa, ataques ativos de análise de tráfego não são considerados, a segurança dos protocolos não é analisada e a especificação formal não contempla todas as propriedades de anonimato.

1.4 Trabalhos Relacionados

Além dos nove protocolos mencionados acima, foram pesquisados mais três: GPSR - *Greedy Perimeter Stateless Routing* (KARP; KUNG, 2000), AO2P - *Ad Hoc On-Demand Position-Based Private Routing* (WU; BHARGAVA, 2005) e AODPR - *Anonymous On-Demand Position-based Routing* (RAHMAN et al., 2006). Não são discutidos por terem classificação distinta; são protocolos que usam informações geográficas.

Nos trabalhos de HONG, KONG & GERLA (2006), KONG (2004) e KONG *et al.* (2005), sugere-se que, além da identidade, é necessário proteger informações como *venue*, topologia e padrão de movimento, e que técnicas *mixing* e roteamento

sob demanda e sem identidades são as melhores abordagens contra ataques de anonimato.

LIU *et al.* (2006) simula os protocolos ANODR, SDAR, ASR, MASK e AnonDSR e os avalia quanto ao desempenho. Os resultados indicam que o impacto no desempenho depende do tamanho do pacote de controle e do grau de mobilidade; há dificuldade em se garantir segurança e desempenho; os protocolos baseados em criptografia simétrica e assimétrica são melhores para nós com baixa e alta capacidade computacional, respectivamente.

BHARGAVAN *et al.* (2002), utilizando o provador de teorema HOL e o checador de modelo SPIN, verifica se o protocolo AODV - *Ad Hoc On-Demand Distance Vector* é livre de *loops*. RENESSE & AGHVAMI (2004), utilizando o checador de modelo SPIN, verificam formalmente o protocolo WARP - *Wireless Adaptive Routing Protocol*. MARSHALL (2003), utilizando lógica BAN e CPAL-ES - *Cryptografic Protocol Analysis Language Evaluation System*, verifica formalmente o protocolo SRP - *Secure Routing Protocol*.

WRIGHT *et al.* (2005) revisa os principais trabalhos relacionados à formalização de anonimato. SCHNEIDER & SIDIROPOULOS (1996) verificam formalmente o esquema *DC-Net* utilizando CSP - *Communicating Sequential Processes*. HALPERN & NEILL (2003) definem formalmente anonimato em sistemas multiagentes de forma probabilística utilizando lógica modal. SHMATIKOV (2004) verifica formalmente o esquema *Crowds* com o checador de modelo probabilístico PRISM.

1.5 Estrutura da Dissertação

O capítulo 2 apresenta uma visão geral de redes sem fio ad hoc móveis: definição, histórico, características, aplicações, arquitetura e segurança. O capítulo 3 aborda aspectos de anonimato: definições, sistemas anônimos para redes fixas e protocolos de roteamento anônimo para redes sem fio ad hoc móveis. O capítulo 4 compara os protocolos de roteamento anônimo propostos para redes sem fio ad hoc móveis: vulnerabilidade a ataques de análise de tráfego e segurança em relação a anonimato são os principais tópicos abordados. O capítulo 5 apresenta o processo de verificação formal do protocolo escolhido a partir da análise: técnica e ferramenta utilizadas, modelagem e resultados. O capítulo 6 apresenta conclusões e trabalhos futuros.

2 *Redes Sem Fio Ad Hoc Móveis*

Neste capítulo, são apresentados os principais conceitos referentes a redes sem fio ad hoc móveis: definição, histórico, características, aplicações, arquitetura e questões de segurança.

2.1 Definição

Uma rede sem fio ad hoc (termo em latim que significa "para isto") móvel ou MANET - *Mobile Ad hoc NETworks* é uma rede formada por dispositivos de comunicação móveis (nós) que se comunicam através de enlaces sem fio, na ausência de infra-estrutura fixa e de controle centralizado (*backbone* ou estação-base). A topologia não é predeterminada e a responsabilidade por organizar e controlar a rede é distribuída entre os nós. Estes são responsáveis por descobrir, dinamicamente, com quais podem se comunicar diretamente e por encaminhar pacotes, cujos destinos não estão no raio de alcance de suas origens (HAAS et al., 1999; RAMANATHAN; REDI, 2002).

A Figura 2.1 ilustra uma rede sem fio ad hoc móvel de enlaces simétricos (se as transmissões do nó X são recebidas por Y, então as transmissões de Y também são recebidas por X), formada por quatro nós. Conforme indicado pelas linhas pontilhadas, os pares de nós A e B, A e C, B e C, C e D podem se comunicar diretamente, enquanto A e D, B e D não podem. Quando o nó A quer se comunicar com D, seus pacotes devem ser roteados através de C.

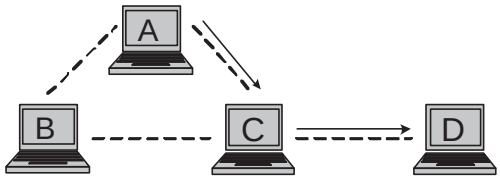

Figura 2.1: Rede Sem Fio Ad Hoc Móvel

2.2 Histórico

As pesquisas em redes sem fio ad hoc móveis iniciaram-se em 1972 com o projeto PRNet - *Packet Radio Network* (KAHN et al., 1978; JUBIN; TORNOW, 1987), desenvolvido pela DARPA - *Defense Advanced Research Projects Agency*. O objetivo deste projeto era desenvolver um sistema de comunicação sem fio, baseado na tecnologia de comutação de pacotes, para aplicação militar. PRNet é uma rede sem fio móvel, distribuída, de múltiplos saltos e com capacidade de operação em uma grande área. O controle de acesso ao meio é uma combinação de ALOHA - *Areal Locations of Hazardous Atmospheres* e CSMA - *Carrier Sense Multiple Access*. A tecnologia de transmissão é o esquema DSSS - *Direct-Sequence Spread Spectrum* e o algoritmo de roteamento é do tipo vetor de distância. Este projeto mostrou, ainda que com limitações, a viabilidade e as vantagens de redes sem fio ad hoc móveis.

Em 1983, a DARPA estendeu o projeto PRNet através do projeto SURAN - *Survivable Radio Networks* (BEYER, 1990), com o objetivo de desenvolver uma rede segura, tolerante a falhas, com escalabilidade e suporte a dispositivos menores, de baixo custo e com capacidades restritas de energia e de processamento. Este projeto resultou em protocolos de gerenciamento de rede avançados, em uma topologia de rede hierárquica baseada em *clusters* dinâmicos e na tecnologia LPR - *Low-cost Packet Radio* (DSSS controlado digitalmente por um microprocessador) (FIFER; BRUNO, 1987).

No início da década de 90, o termo *packet radio network* foi substituído por ad hoc pelo grupo de trabalho IEEE 802.11 WLAN (IEEE 802.11 WG, 2006) e essas redes foram incluídas na padronização do protocolo IEEE 802.11. Além disso, com os notebooks e outros equipamentos, surgiu o interesse em aplicações não militares. O ETSI - *European Telecommunications Standards Institute* iniciou a padronização do protocolo HIPERLAN/1 - *HIgh PErformance Radio Local Area Network*, concluído em 1996. Em 1997, o ETSI criou o projeto BRAN - *Broad Band Radio Access*

Networks (ETSI, 2006), a fim de criar especificações para HIPERLAN/2 (primeira versão em 2000), HIPERACCESS - *HIgh PErformance Radio Access* e HIPERMAN - *HIgh PErformance Radio Metropolitan Area Network*.

Em 1994, a DARPA iniciou o programa GloMo - *Global Mobile Information Systems* (LEINER; RUTH; SASTRY, 1996), cujo objetivo era oferecer conectividade multimídia do tipo Ethernet em qualquer hora e lugar entre dispositivos sem fio móveis. Neste mesmo ano, a Ericsson propôs o desenvolvimento do Bluetooth - tecnologia para WPAN - *Wireless Personal Area Network*. A primeira versão foi concluída em 1999 pelo Bluetooth SIG - *Special Interest Group* (BLUETOOTH SIG, 2006), um conjunto de empresas de computação e de telecomunicação, fundado em 1998. Em 2002, o grupo de trabalho IEEE 802.15 (IEEE 802.15 WG, 2006) publicou a especificação para redes pessoais IEEE 802.15 (Bluetooth), que teve como base o documento do SIG.

Em 1997, o IETF - *Internet Engineering Task Force* formou o grupo de trabalho MANET - *Mobile Ad hoc NETworks* (MANET WG, 2007), com o objetivo de padronizar protocolos de roteamento. RFCs (*Request for Comments*), com *status* experimental, e *IETF Internet-Drafts* foram publicados: *Ad hoc On-Demand Distance Vector (AODV) Routing* (PERKINS; BELDING-ROYER; DAS, 2003), *Optimized Link State Routing Protocol (OLSR)* (CLAUSEN; JACQUET, 2003), *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)* (OGIER; TEMPLIN; LEWIS, 2004), *Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4* (JOHNSON; HU; MALTZ, 2007), *Dynamic MANET On-demand (DYMO) Routing* (CHAKERES; PERKINS, 2007) e *The Optimized Link State Routing Protocol version 2* (CLAUSEN; DEARLOVE; JACQUET, 2007). O grupo de trabalho IEEE 802.11 lançou a primeira versão do protocolo 802.11, que funcionava na faixa ISM (*Industrial, Scientific, and Medical*), com velocidade de 1 Mbps ou 2 Mbps.

Com o objetivo de criar padrões mais rápidos, foram criados dois grupos - 802.11a e 802.11b. Em 1999, os grupos publicaram, respectivamente, os protocolos IEEE 802.11a e IEEE 802.11b. IEEE 802.11a opera na banda de freqüência 5 GHz, com taxa de dados de 54 Mbps. IEEE 802.11b (também chamado de *Wireless-Fidelity* ou Wi-Fi) opera na banda 2,4 Ghz ISM, com taxa de dados de 11 Mbps. Em 2003, o grupo 802.11g estendeu o padrão 802.11b, criando o IEEE 802.11g com velocidade de 54 Mbps. Outros grupos foram criados com objetivos como: definir

extensões para prover qualidade de serviço, melhorar protocolos e resolver problemas de interoperabilidade.

Projetos em redes sem fio ad hoc móveis com finalidade militar continuam sendo realizados. No meio acadêmico, pesquisas são conduzidas, a fim de propor soluções para questões como roteamento, segurança, qualidade de serviço, conservação de energia, entre outras. Com o surgimento dos padrões e evolução dos dispositivos sem fio, o interesse comercial aumentou e produtos como, por exemplo, MeshNetworks (MESHNETWORKS, 2006) e SPANWorks (SPANWORKS INC., 2006) foram lançados (CHLAMTAC; CONTI; LIU, 2003).

O Quadro 2.1 resume os principais acontecimentos na evolução das redes sem fio ad hoc móveis.

2.3 Características

Segundo CORSON & MACKER (1999), CHLAMTAC et al. (2003) e MURTHY & MANOJ (2004), as principais características de redes sem fio ad hoc móveis são:

- Autonomia e ausência de infra-estrutura: redes sem fio ad hoc móveis são auto-organizadas, isto é, não dependem de qualquer infra-estrutura pré-estabelecida nem administração centralizada para seu gerenciamento e manutenção. O gerenciamento da rede é feito de modo distribuído. Assim, não há um ponto único de falha, porém a detecção de falhas e o gerenciamento da rede tornam-se mais difíceis;

- Topologia dinâmica: como os nós se movem de forma arbitrária, a topologia muda freqüentemente. A mobilidade é uma característica necessária em algumas aplicações, mas pode resultar em mudanças de rotas, partições da rede e perda de pacotes;

- Roteamento distribuído através de múltiplos saltos (nós): todo nó funciona como roteador e encaminha pacotes para destinos que não estão no raio de alcance de suas origens;

- Instalação rápida e com baixo custo: o tempo e o custo são menores do que os requeridos por redes estruturadas, visto que não há necessidade de cabos e

Quadro 2.1: Principais Acontecimentos na Evolução das Redes Sem Fio Ad Hoc Móveis

Data	**Acontecimento**
1972	*Projeto Packet Radio Network* (PRNet)
1983	*Projeto Survivable Radio Networks* (SURAN)
Início da década de 90	Adoção do termo ad hoc Inclusão de redes sem fio ad hoc móveis na padronização do IEEE 802.11
1994	Programa *Global Mobile Information System* (GloMo)
1996	Padronização do protocolo HiperLAN1
1997	Projeto *Broadband Radio Access Networks* (BRAN)* Grupo de trabalho MANET* Primeira versão do padrão IEEE 802.11*
1998	Bluetooth SIG*
1999	Primeira versão do Bluetooth* Protocolo IEEE 802.11a* Protocolo IEEE 802.11b*
2000	Primeira versão do protocolo HiperLAN2*
2002	Primeira versão do protocolo IEEE 802.15 (Bluetooth)*
2003	RFC 3561: *Ad hoc On-Demand Distance Vector* (AODV) *Routing** RFC 3626: *Optimized Link State Routing Protocol* (OLSR)* Protocolo IEEE 802.11g*
2004	RFC 3684: *Topology Dissemination Based on Reverse-Path Forwarding* (TBRPF)*
2007	*Internet-Draft: Dynamic MANET On-demand* (DYMO) *Routing** *Internet-Draft: The Optimized Link State Routing Protocol version 2** RFC 4728: *The Dynamic Source Routing Protocol* (DSR) *for Mobile Ad Hoc Networks for IPv4**

NOTA: * indica trabalho em andamento.

manutenção;

- Implantação incremental: redes sem fio geograficamente distribuídas (WWAN - *Wireless Wide Area Network*) baseadas em redes sem fio ad hoc móveis podem ser implantadas de forma incremental. Não é necessário que toda a rede esteja instalada para que as partes já prontas possam funcionar;

- Largura de banda limitada, enlaces com capacidade variável: a vazão é menor do que a taxa de transmissão máxima do canal, devido a acesso múltiplo, interferência, enfraquecimento do sinal, entre outros problemas próprios do meio sem fio. Os enlaces entre os nós possuem baixa capacidade e podem ser assimétricos (quando a capacidade de transmissão e de recepção dos nós são diferentes, um nó X pode estar no raio de transmissão de Y, mas Y não estar no de X);

- Operação com restrição de energia: as operações realizadas por cada nó são limitadas pela quantidade de energia disponível em suas baterias;

- Segurança física limitada: redes sem fio ad hoc móveis são mais vulneráveis a ataques do que as redes estruturadas. Há maior possibilidade de ataques de escuta, de negação de serviço, de injeção de pacotes falsos, de modificação dos pacotes e de comprometimento dos nós.

2.4 Aplicações

Inicialmente, as redes sem fio ad hoc móveis foram desenvolvidas tendo como enfoque aplicações militares. Com o avanço das pesquisas, atraiu-se o interesse de outros setores e essas redes também passaram a ser utilizadas em aplicações civis e comerciais (CHLAMTAC; CONTI; LIU, 2003).

Devido a características como custo e flexibilidade, podem ser utilizadas em diversos cenários: guerras, operações de emergência e resgate, redes de sensores, conferências, entre outros. O Quadro 2.2 categoriza os diferentes cenários de aplicação atuais e futuros.

Quadro 2.2: Aplicações de Redes Sem Fio Ad Hoc Móveis

Aplicações	**Descrição/Serviços**
Redes Táticas	Comunicação em operações militares Batalhas automatizadas
Redes de Sensores	Monitoramento de residências Medição de parâmetros como umidade, temperatura, radiação nuclear Monitoramento de dados, como por exemplo atividades sísmicas
Emergência	Operações de busca e resgate Substituição da infra-estrutura fixa em caso de terremotos, furacões
Ambientes Comerciais	Comércio eletrônico: serviços como pagamento de contas em qualquer lugar Negócios: acesso dinâmico a arquivos armazenados em uma localização central, escritório móvel Serviços em veículos: transmissão de notícias, condição das estradas, tempo e música; formação de rede entre veículos próximos
Redes Caseiras e Corporativas	Redes sem fio locais (WLAN - *Wireless Local Area Network*) Redes sem fio pessoais (WPAN - *Wireless Personal Area Network*)
Aplicações Educacionais	Configuração de salas de aula virtuais ou salas de conferência Criação de uma rede para comunicação rápida em conferências, encontros e palestras
Redes Mesh	Zonas residenciais: conexão à Internet Auto-estradas: comunicação para os automóveis Zonas comerciais: alternativa à rede de celulares Campus universitário: rede de baixo custo
Entretenimento	Acesso à Internet em ambientes abertos Jogos entre múltiplos jogadores
Localização de Serviços	Serviços de informação: localização de serviços como postos de gasolina

Fonte: CHLAMTAC et al. (2003) e MURTHY & MANOJ (2004).

2.5 Arquitetura de um Nó

Em uma rede sem fio ad hoc móvel, a arquitetura básica de um nó é constituída por cinco camadas: física, enlace de dados, rede, transporte e aplicação. As subseções descrevem, resumidamente, as quatro primeiras.

2.5.1 Camada Física e Camada de Enlace de Dados

O padrão IEEE 802.11 especifica as camadas física e de enlace de dados de WLANs. As técnicas de transmissão suportadas são: infravermelho, FHSS - *Frequency Hopping Spread Spectrum*, DSSS - *Direct-Sequence Spread Spectrum*, OFDM - *Orthogonal Frequency Division Multiplexing* e HR-DSSS - *High Rate Direct Sequence Spread Spectrum*. Essas técnicas diferem quanto à: modulação, codificação, faixa de freqüência, largura de banda e velocidade alcançada. A sub-camada MAC - *Medium Access Control* oferece dois métodos de controle de acesso ao meio: PCF - *Point Coordination Function* e DCF - *Distributed Coordination Function*. No método PCF, o controle é centralizado (feito pela estação-base), portanto não pode ser utilizado no modo ad hoc. No método DCF, o controle é distribuído e o protocolo utilizado é o CSMA/CA - *Carrier Sense Multiple Access with Collision Avoidance*, com mecanismo RTS/CTS (envio de quadros *Request To Send* e *Clear To Send*).

Pesquisas sobre o protocolo MAC do IEEE 802.11 demonstraram problemas de desempenho no modo ad hoc. Outros protocolos MAC foram propostos e podem ser classificados em: baseados em contenção, baseados em contenção com reserva de recursos e baseados em contenção com mecanismos de escalonamento (MURTHY; MANOJ, 2004). Nos protocolos baseados em contenção, os nós disputam o canal simultaneamente, ou seja, não há reserva de recursos. Nos protocolos baseados em contenção com reserva de recursos, há mecanismos para reserva do canal. Nos protocolos baseados em contenção com mecanismos de escalonamento, critérios como prioridade de fluxo, igualdade dos nós e energia restante são utilizados para determinar qual nó deve ter acesso ao canal.

2.5.2 Camada de Rede

Protocolos usados em redes tradicionais não podem ser utilizados diretamente em redes sem fio ad hoc móveis, devido à topologia dinâmica, ausência de infra-estrutura estabelecida para administração centralizada, enlaces sem fio com limitação da largura de banda e nós com recursos restritos. Os principais desafios de projeto são a mobilidade, a limitação de recursos, o canal propenso a erros e os problemas "terminal escondido" e "terminal exposto" (MURTHY; MANOJ, 2004).

Vários protocolos de roteamento para redes sem fio ad hoc móveis foram propostos, mas ainda não foi estabelecido um padrão. Pode-se destacar os protocolos DSR - *Dynamic Source Routing* (JOHNSON; MALTZ, 1996), AODV - *Ad hoc On-Demand Distance Vector* (PERKINS; ROYER, 1999), OLSR - *Optimized Link State Routing Protocol* e TBRPF - *Topology Dissemination Based on Reverse-Path Forwarding*. Os dois primeiros são estudados e citados freqüentemente nas pesquisas acadêmicas. O grupo MANET criou RFCs para AODV, DSR, OLSR e TBRPF, além de *Internet Drafts* para outros protocolos.

CORSON & MACKER (1999) indicam as propriedades que os protocolos de roteamento para essas redes devem possuir: operação distribuída, ausência de *loop*, operação sob demanda, operação pró-ativa (nos casos em que latência é inaceitável), segurança, operação com períodos inativos e suporte a enlaces uni-direcionais (assimétricos). Outras propriedades interessantes são: adaptação a freqüentes mudanças de topologia, conservação de energia, múltiplas rotas e suporte a qualidade de serviço (MURTHY; MANOJ, 2004).

Protocolos de roteamento para redes sem fio ad hoc móveis podem ser classificados de acordo com critérios como: mecanismo de atualização das informações (ROYER; TOH, 1999), uso de informações temporais (MURTHY; MANOJ, 2004), organização da topologia (FEENEY, 1999), uso de recursos específicos (MURTHY; MANOJ, 2004) e tipo de transmissão (KUOSMANEN, 2002). Esses critérios não são os únicos e não são mutuamente exclusivos.

De acordo com o mecanismo de atualização das informações de roteamento, os protocolos podem ser classificados em três categorias:

- Protocolos orientados à tabela ou pró-ativos: todo nó mantém tabelas com informações de roteamento para cada um dos nós presentes na rede. Essas

informações são enviadas e atualizadas periodicamente.

- Protocolos sob demanda ou reativos: as rotas são estabelecidas e mantidas somente quando necessárias. Quando um nó deseja se comunicar com outro para o qual não possui rota válida, inicia o processo de descoberta. Enquanto a rota é necessária, é executado o processo de manutenção.

- Protocolos híbridos: combinam as abordagens anteriores, a fim de obter suas vantagens. Para cada nó, é definida uma área, chamada zona de roteamento, que contém os vizinhos pertencentes a uma área geográfica específica ou distantes, no máximo, n saltos deste nó. Para rotear pacotes dentro de sua zona, executa um algoritmo pró-ativo e para fora, um sob demanda.

De acordo com o uso de informações temporais para roteamento, os protocolos podem ser classificados em duas categorias:

- Protocolos que usam histórico: as decisões de roteamento são tomadas com base no estado (passado ou do momento do roteamento) dos enlaces.

- Protocolos que usam predição: as decisões de roteamento são tomadas com base na suposição do estado futuro dos enlaces, considerando informações como bateria e localização, por exemplo.

De acordo com a organização da topologia, os protocolos podem ser classificados em duas categorias:

- Protocolos com topologia *flat* ou uniformes: todo nó possui o mesmo papel durante o roteamento.

- Protocolos com topologia hierárquica ou não uniformes: os nós são organizados hierarquicamente e a responsabilidade pelo roteamento restringe-se a alguns nós. Há duas categorias: uma em que cada nó possui alguns vizinhos roteadores e outra em que a rede é particionada em *clusters*, onde há um nó (*cluster-head*), responsável pelo roteamento.

De acordo com os recursos específicos utilizados, os protocolos podem ser classificados em três categorias:

- Protocolos com controle de energia: as decisões de roteamento são tomadas considerando a redução do consumo de energia.
- Protocolos que usam informações geográficas: as posições dos nós são obtidas através, por exemplo, de um GPS - *Global Positioning System*.
- Protocolos com inundação eficiente: métodos são aplicados para que os pacotes sejam difundidos eficientemente.

De acordo com o tipo de transmissão, os protocolos podem ser classificados em três categorias:

- Protocolos *unicast*: um nó envia pacotes para um único destino.
- Protocolos *multicast*: um nó envia pacotes para um subconjunto de destinos. Baseado na topologia, podem ser divididos em *mesh-based* e *tree-based*. Protocolos *mesh-based* constroem uma malha entre os nós e, portanto, pode haver vários caminhos entre cada par origem-destino. Protocolos *tree-based* constroem árvores para conectar os nós e há um único caminho entre cada par origem-destino. Podem ser subdivididos em *source-tree-based* e *shared-source-based*. Os protocolos *source-tree-based* mantêm uma árvore para cada nó origem e seus destinos. Os protocolos *shared-source-based* mantêm uma única árvore para todo o grupo.
- Protocolos *geocast*: um nó envia pacotes para um grupo de nós localizados em uma área geográfica específica.

As Figuras 2.2 e 2.3 resumem a classificação.

2.5.3 Camada de Transporte

O protocolo TCP - *Transmission Control Protocol* não pode ser diretamente aplicado em redes sem fio ad hoc móveis, devido à má interpretação de perda de pacotes, quebra de enlaces freqüentes, enlaces assimétricos, entre outros fatores (MURTHY; MANOJ, 2004). Quando o protocolo UDP - *User Datagram Protocol* é utilizado, esses problemas não acontecem ou acontecem com menos intensidade (CHLAMTAC; CONTI; LIU, 2003). Por isso, foram propostos protocolos específicos para redes sem fio ad hoc móveis e mecanismos para otimização do protocolo TCP.

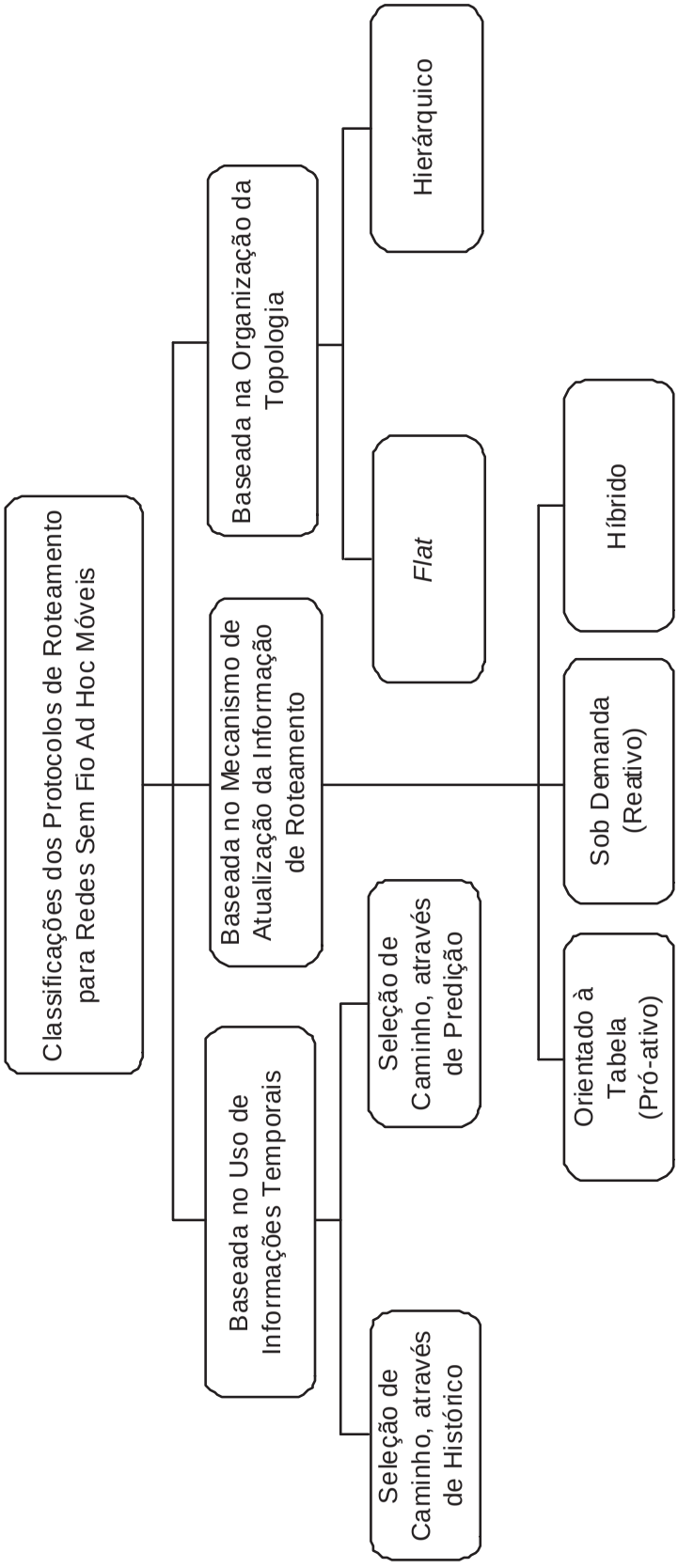

Figura 2.2: Classificações dos Protocolos de Roteamento

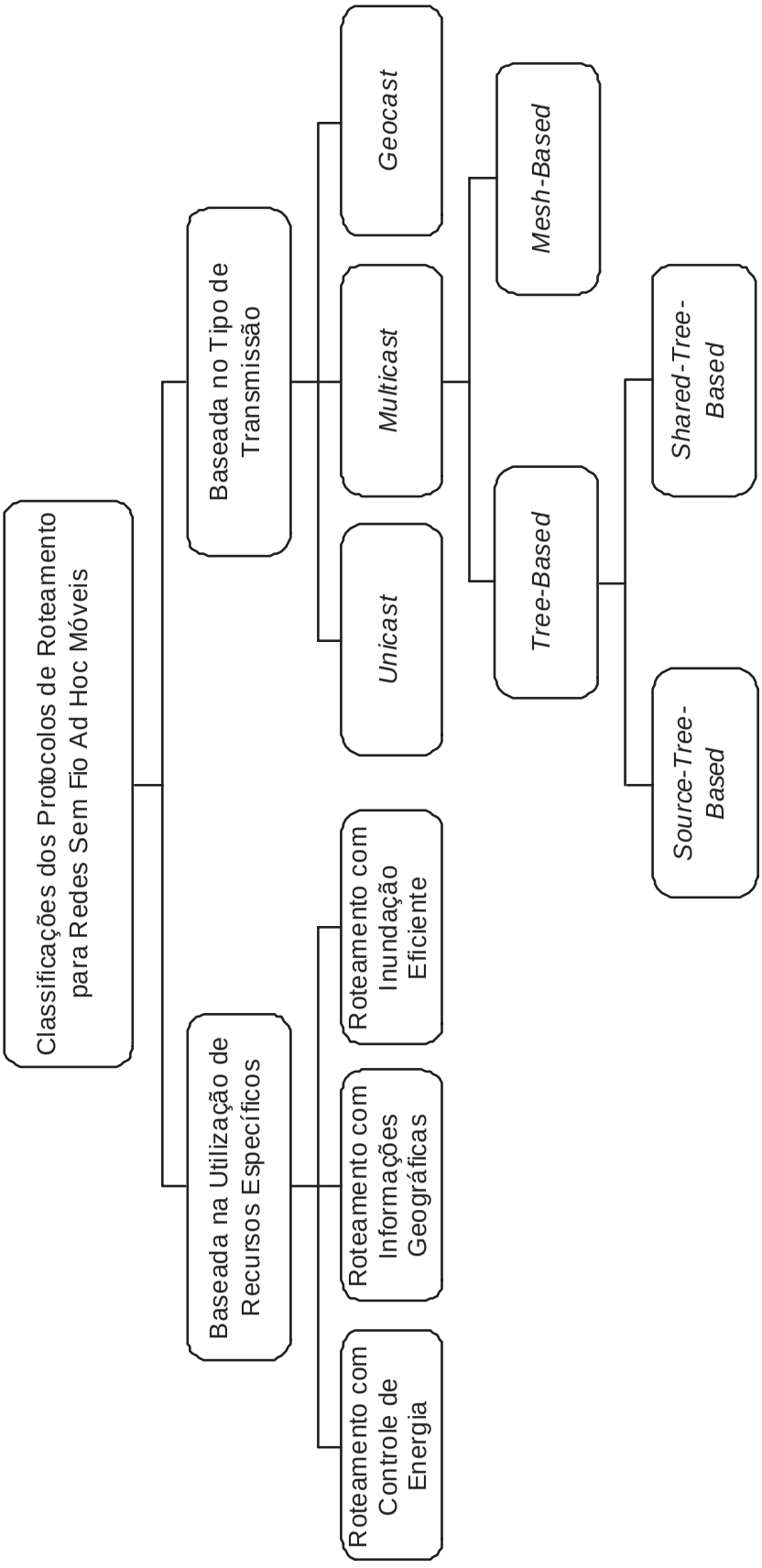

Figura 2.3: Classificações dos Protocolos de Roteamento

2.6 Aspectos de Segurança

Segundo MISHRA & NADKARNI (2003), segurança em redes sem fio ad hoc móveis consiste em: disponibilidade, autenticidade, confidencialidade, integridade, não-repúdio, privacidade, anonimato, robustez, entre outros requisitos. As principais pesquisas nessa área podem ser divididas em três vertentes: mecanismos seguros para estabelecimento de chaves criptográficas (BUTTYAN; HUBAUX, 2003; MISHRA; NADKARNI, 2003; MICHIARDI; MOLVA, 2004), detecção de intrusão (MISHRA; NADKARNI, 2003; MISHRA; NADKARNI; PATCHA, 2004) e segurança dos protocolos de roteamento (MISHRA; NADKARNI, 2003; HU; PERRIG, 2004; MICHIARDI; MOLVA, 2004; YANG et al., 2004).

Redes sem fio ad hoc móveis são, inerentemente, vulneráveis a ataques, devido a características como topologia dinâmica, vulnerabilidade dos nós e dos enlaces, ausência de gerenciamento e monitoramento centralizado e limitação de recursos (MISHRA; NADKARNI, 2003). As soluções tradicionais não podem ser aplicadas diretamente, porque, normalmente, dependem de uma infra-estrutura fixa (YANG et al., 2004).

Os ataques podem ser classificados em ativos ou passivos, externos ou internos (MISHRA; NADKARNI, 2003; MURTHY; MANOJ, 2004). Em um ataque ativo, o adversário interfere na rede e tenta mudar o comportamento normal dos protocolos. Exemplos: *wormhole*, manipulação do tráfego (fabricação, modificação, destruição), seqüestro de sessão, personificação e negação de serviço. Em um ataque passivo, o adversário somente observa a rede, a fim de obter informações importantes (explícitas ou implícitas). Exemplos: monitoramento das mensagens e análise de tráfego. Ataques externos são realizados por nós externos à rede, que têm acesso ao meio de transmissão, e internos são realizados por nós pertencentes à rede, que têm acesso ao meio e às configurações (nós comprometidos ou atacantes autenticados).

Entre os ataques ativos contra a camada de rede (MISHRA; NADKARNI, 2003; HU; PERRIG, 2004; MICHIARDI; MOLVA, 2004; MURTHY; MANOJ, 2004), destacam-se:

- Ataque bizantino: um nó intermediário ou um conjunto de nós intermediários comprometidos aliam-se para realizar ataques, como criação de *loops* e envio de mensagens de atualização falsas.

- Ataques contra o protocolo de roteamento: estouro da tabela de roteamento por meio da criação de rotas para nós inexistentes, inserção de informações falsas nas tabelas de roteamento por meio de pacotes de atualização ou de roteamento falsos, repetição de pacotes, *rushing* (é realizado contra protocolos reativos; um adversário dissemina requisições de rota rapidamente, a fim de impedir que requisições legítimas sejam tratadas, uma vez que um nó, ao recebê-las, pensará que se trata de duplicação).

- *Blackhole*: o adversário induz os nós a acreditarem que possui o melhor caminho para determinado destino por meio de informações de roteamento falsas (no caso de protocolos reativos, métricas durante a descoberta de rotas e, no caso de pró-ativos, as mensagens de atualização). Assim, pode interceptar e descartar pacotes.

- Consumo de recursos: o adversário tenta consumir os recursos (computacional, energia) dos nós presentes na rede.

- *Wormhole*: dois nós maliciosos estabelecem um túnel entre si, através do qual, podem receber pacotes em uma localização da rede e retransmiti-los em outra. Assim, podem controlar algumas rotas.

Através de análise de tráfego, um adversário pode inferir informações relevantes como os nós origem e destino, localização, topologia, freqüência da comunicação e padrões de movimento. Entre os ataques de análise de tráfego (RAYMOND, 2000; SONG; KORBA, 2002), destacam-se:

- Ataque de análise de tempo: se os pacotes são processados e encaminhados na mesma ordem em que são recebidos, um atacante pode inferir quais pertencem à mesma rota ou quais transmissões referem-se a um pacote sendo encaminhado.

- Ataque de conteúdo do pacote: um atacante pode identificar e seguir um pacote, quando parte ou todo seu conteúdo (campos do cabeçalho ou carga útil) permanece inalterado durante sua transmissão.

- Ataque de volume do pacote: um atacante pode identificar e seguir um pacote durante sua transmissão, quando pacotes distintos possuem tamanhos diferen-

tes e seus volumes não são alterados ou são modificados de forma padrão pelos nós intermediários.

- Ataque de reconhecimento de fluxo: se os pacotes são vulneráveis a análise de tempo ou se possuem informações em comum, um atacante pode identificar quais pertencem à mesma rota.

2.7 Considerações

Este capítulo apresentou uma visão geral sobre redes sem fio ad hoc móveis, destacando-se definição, características e ataques passivos de análise de tráfego. Essas redes possuem inúmeras aplicações e benefícios, porém suas características inerentes também tornam a segurança uma questão desafiadora. Entre os ataques, análise de tráfego é um dos mais problemáticos, pois é difícil detectá-lo e preveni-lo. Os aspectos de segurança relacionados às camadas de enlace, de transporte e de aplicação não são discutidos, por estarem fora do escopo do trabalho. O próximo capítulo discorre sobre anonimato e privacidade, com ênfase na definição e nos protocolos de roteamento anônimo propostos para redes sem fio ad hoc móveis.

3 *Anonimato*

Neste capítulo, são introduzidos conceitos relevantes sobre anonimato: definições, sistemas para prover anonimato em redes estruturadas e protocolos de roteamento anônimo para redes sem fio ad hoc móveis.

3.1 Conceitos e Definições

O padrão ISO/IEC 15408 (COMMON CRITERIA, 1999), que descreve requisitos de segurança para a avaliação de propriedades de segurança de produtos e sistemas de tecnologia, indica quatro requisitos de privacidade:

- Anonimato: assegura que um usuário pode usar um recurso ou serviço sem revelar sua identidade.
- Pseudo-anonimato: assegura que um usuário pode usar um recurso ou serviço sem revelar sua identidade, mas ainda pode ser associado a esse uso.
- Não-correlação: assegura que um usuário pode utilizar vários recursos ou serviços sem que outros usuários possam correlacionar esses usos.
- Não-observação: assegura que um usuário pode usar um recurso ou serviço sem que outros usuários possam ser capazes de observar que o recurso ou serviço está em uso.

PFITZMANN & HANSEN (2006) propõem uma terminologia para a área de anonimato e justificam por que sua definição é melhor do que a apresentada pelo padrão ISO/IEC 15408:

- Anonimato: uma entidade (ex.: pessoa, computador, processo), ao realizar uma ação, não pode ser identificada entre todas as possíveis entidades que

poderiam tê-la realizado (conjunto de anonimato). Esta definição é mais geral, pois não se restringe à proteção da identidade dos usuários, mas a qualquer entidade.

- Não-correlação: a probabilidade de relacionar dois ou mais itens de interesse (ex.: entidades, mensagens, ações, relações) permanece igual antes e depois de serem observados por um adversário. Esta definição é mais geral, pois não enfoca os usuários, mas qualquer item de interesse. Além disso, é uma definição relativa (afirma que o conhecimento do adversário sobre a relação entre os itens não muda), enquanto a da ISO 15408 é absoluta (afirma que não há conhecimento sobre a relação entre o usuário e suas ações).

- Não-detecção: um item de interesse é indistinguível de outros, ou seja, não é detectado. Ao contrário de anonimato e não-correlação, onde a relação entre item de interesse e elemento é protegida, não-detecção se preocupa com a proteção do item de interesse. Em versões anteriores, PFITZMANN & HANSEN utilizaram esta definição para definir não-observação.

- Não-observação: um item de interesse é indistinguível de outros por todas as entidades não relacionadas aos possíveis itens, e a relação entre um item de interesse e uma entidade permanece anônima a outra(s) entidade(s) relacionada(s) ao mesmo item.

- Pseudo-anonimato: pseudônimos são usados como identificadores das entidades. Tipos de pseudônimos: públicos - relação entre identidades reais e pseudônimos é conhecida antes de iniciar a comunicação; inicialmente não públicos - relação entre identidades reais e pseudônimos não é conhecida por todos; não associados inicialmente - relação entre identidades reais e pseudônimos não é estabelecida antes de iniciar a comunicação.

Considerando o envio e o recebimento de mensagens como itens de interesse, anonimato pode ser definido em termos de não-correlação entre um item de interesse e uma entidade qualquer (PFITZMANN; HANSEN, 2006):

- Anonimato da origem: uma mensagem não pode ser relacionada a qualquer origem, e a uma origem não é relacionada qualquer mensagem.

- Anonimato do destino: uma mensagem não pode ser relacionada a qualquer destino, e a um destino não é relacionada qualquer mensagem.

- Anonimato da relação: não é possível descobrir as partes envolvidas na comunicação, ou seja, origem e destino não podem ser relacionados.

KONG (2004), KONG et al. (2005) e HONG et al.(2006), considerando roteamento em redes sem fio ad hoc móveis, acrescentam novos requisitos às definições anteriores:

- Anonimato da origem (ou destino): compreende dois aspectos - anonimato da identidade e do *venue*; *venue* é o local onde se encontra um nó, sob o ponto de vista de um adversário (normalmente, compreende uma área), após detectar uma transmissão. Um adversário não pode correlacionar a identidade da origem (ou destino) ou seu *venue* a um evento de conexão fim a fim (transmissões referentes a um pacote enviado pela origem para o destino).

- Anonimato da relação entre origem e destino: um adversário não pode identificar a correlação entre origem e destino, ou seja, não pode seguir um fluxo em direção à origem e/ou ao destino.

- Privacidade da localização ou privacidade do *venue*: um adversário não pode identificar os membros da rede e seus padrões de comunicação em determinada localização.

- Privacidade da topologia da rede: compreende os itens anteriores, uma vez que a identificação de uma rota revela parcialmente a topologia e se não há anonimato da identidade e privacidade da localização, os nós podem ser identificados e localizados.

- Privacidade do padrão de movimento: um adversário não pode identificar a localização anterior e a atual de um conjunto de nós, assim como qualquer indicação da direção de movimentação.

KONG (2004) também provê uma definição formal de anonimato, utilizando o conceito de entropia. Um sistema provê determinado grau de anonimato, $H(A) \rightsquigarrow H(B)$, se $(H(B) - H(A)) < c$, onde $H(B)$ é um limite de entropia e c é uma constante

definida pelo sistema, dentro dos parâmetros de segurança. Se $c = 0$, o anonimato é perfeito. Assim:

- Anonimato da identidade da origem: a rede provê esse tipo de anonimato, se $H(ID_S|X) \rightsquigarrow H(I)$, onde ID_S é a identidade da origem; I é o conjunto, cujo tamanho é N, de identidades conhecidas; X são as transmissões interceptadas pelo atacante; $H(ID_S) = H(I) = log_2 N$ é a entropia de incerteza que mede o conhecimento do adversário antes de qualquer transmissão; $H(ID_S|X)$ é o conhecimento do adversário sobre a identidade da origem após as transmissões.

- Anonimato da identidade do destino: é definido de forma semelhante ao item anterior, como $H(ID_D|X) \rightsquigarrow H(I)$, onde ID_D é a identidade do destino.

- Anonimato do *venue* da origem: a rede provê esse tipo de anonimato, se $H(V_S|X) \rightsquigarrow H(V)$, onde V_S é o *venue* da origem; V é o conjunto, cujo tamanho é N, de *venues* identificados; X são as transmissões interceptadas pelo atacante; $H(V_S) = H(V) = log_2 N$ é a entropia de incerteza que mede o conhecimento do adversário antes de qualquer transmissão; $H(V_S|X)$ é o conhecimento do adversário sobre o *venue* da origem após as transmissões.

- Anonimato do *venue* do destino: é definido de forma semelhante ao item anterior, como $H(V_D|X) \rightsquigarrow H(V)$, onde V_D é o *venue* do destino.

- Anonimato da relação entre as identidades da origem e do destino: a rede provê esse tipo de anonimato, se $H((ID_S, ID_D)|X) \rightsquigarrow H(I, I)$, onde ID_S e ID_D são, respectivamente, as identidades da origem e do destino; X são as transmissões interceptadas pelo atacante; $H(ID_S, ID_D) = H(I, I) = 2log_2 N$ é a entropia de incerteza antes de qualquer transmissão; $H((ID_S, ID_D)|X)$ é o conhecimento do adversário sobre a relação entre as identidades após as transmissões.

- Anonimato da relação entre os *venues* da origem e do destino: é definido de forma semelhante ao item anterior, como $H((V_S, V_D)|X) \rightsquigarrow H(V, V)$. Quaisquer dois eventos de transmissão de um mesmo evento de conexão fim a fim não podem ser correlacionados. Evento de transmissão é um pacote que pode ser interceptado e ter seu formato, conteúdo, tempo e *venue* de interceptação registrados.

- Privacidade da localização ou privacidade do *venue*: emissor e destinatário podem ser quaisquer nós em um *vertex* (local onde se encontra um adversário), isto é, $H(ID_S) \rightsquigarrow H(I)$ e $H(ID_D) \rightsquigarrow H(I)$. Quaisquer dois eventos de transmissão ocorridos no mesmo *vertex* não podem ser correlacionados. Se o sistema assegura os dois itens, há privacidade forte da localização, mas se o segundo não é assegurado, há privacidade fraca da localização.

- Privacidade do padrão de movimento: emissor e destinatário de qualquer evento de transmissão podem ser quaisquer nós, isto é, $H(ID_S) \rightsquigarrow H(I)$ e $H(ID_D) \rightsquigarrow H(I)$. Quaisquer dois eventos de transmissão, de um único nó ou de um conjunto de nós, não podem ser correlacionados globalmente. Se o sistema assegura os dois itens, há privacidade forte do padrão de movimento, mas se o segundo não é assegurado, há privacidade fraca do padrão de movimento.

Considerando roteamento em redes sem fio ad hoc móveis, ZHU et al. (2004) define os seguintes conceitos:

- Privacidade da identidade: as identidades reais dos nós origem e destino são conhecidas somente por eles, e os nós origem e destino não sabem as identidades reais dos nós intermediários que pertencem à rota.

- Privacidade fraca da localização: a localização dos nós origem e destino é conhecida somente por eles.

- Privacidade forte da localização: além do requisito anterior, nós intermediários não sabem sua distância (número de saltos) em relação aos nós origem e destino.

- Anonimato da rota: adversários pertencentes ou não à rota não podem rastrear o fluxo de pacotes de volta ao nó origem ou destino; adversários fora da rota não possuem qualquer informação sobre partes da rota; é difícil para nós adversários inferirem padrões de transmissão e de movimento dos nós origem e destino.

Considerando as definições anteriores e roteamento em redes sem fio ad hoc móveis, anonimato é a propriedade de não revelar, explícita ou implicitamente, as identidades dos nós origem, destino e intermediários, suas localizações e padrões de tráfego e de movimento, além de não ser possível identificar a rota.

3.2 Anonimato e Redes Estruturadas

Esta seção apresenta uma breve descrição de esquemas para anonimato em redes estruturadas. Alguns protocolos anônimos para redes sem fio ad hoc móveis são baseados em algumas destas propostas.

3.2.1 *Mix-Net*

CHAUM (1981) apresenta a técnica *Mix-Net* (rede de misturadores), cujo objetivo é esconder o remetente e os dados da comunicação em um sistema de e-mail.

O sistema consiste em computadores, chamados misturadores, responsáveis por processar os e-mails antes que sejam entregues ao destino e esconder a correlação entre a mensagem que chega e a que sai. Ao invés de enviar uma mensagem diretamente ao destino, o usuário envia para um misturador, que a envia para o destino ou para uma cascata de misturadores. No caso de um sistema com um único misturador, o usuário cifra os dados (M) e um número aleatório (R_0) com a chave pública do destino (PK_D), anexa o endereço do destino (A_D) e um número aleatório (R_1), cifra com a chave pública do misturador (PK_M) e envia para o misturador [E_{PK_M}(R_1, E_{PK_D}(R_0, M), A_D)]. O misturador decifra a mensagem recebida, obtém [E_{PK_D}(R_0, M), A_D] e encaminha para o destino. Quando o sistema consiste em uma cascata de misturadores, o usuário cifra a mensagem várias vezes; cada vez, usando a chave de um misturador. A mensagem, após ser processada pelo misturador, não pode ser correlacionada à mensagem recebida, devido às transformações criptográficas e ao uso de técnicas *mixing*, como retardo aleatório, encaminhamento reordenado das mensagens e envio de mensagens falsas, caso necessário.

3.2.2 *DC-Net*

CHAUM (1988) apresenta o esquema *DC-Net* (Jantar dos Criptólogos), cujo objetivo é manter a confidencialidade de quem envia mensagens.

Três criptólogos jantam em um restaurante. No fim do jantar, o garçom informa que a conta foi paga por um dos três ou pela *U.S. National Security Agency*. Desejando saber se a conta foi paga por um deles ou pela agência, desenvolvem um protocolo. Cada criptólogo joga uma moeda, de modo que somente ele e o criptólogo

à sua direita vejam o resultado. Então, cada um diz, em voz alta, se as duas moedas que viram são iguais ou não; se pagou o jantar (quer enviar mensagem), responde incorretamente, ou seja, se as moedas são iguais, diz que são diferentes e vice-versa. Se o número de respostas diferentes é ímpar, significa que um deles pagou a conta, mas não é possível identificar quem foi. A confidencialidade do emissor é mantida, desde que os participantes com quem compartilha algum segredo não estejam em conluio.

3.2.3 *Onion Routing*

REED et al. (1996, 1998) propõe o esquema *Onion Routing*, cujo objetivo é prover conexões anônimas, em tempo real, resistentes a espionagem e análise de tráfego.

Uma aplicação não se conecta diretamente à máquina destino, mas a um conjunto de máquinas, chamadas roteadores *onion*. O primeiro roteador é responsável por definir uma rota até o destino. É criada uma estrutura com várias camadas, chamada *onion*, onde cada camada é cifrada, corresponde a um roteador e possui o endereço do próximo roteador e material *key seed* para geração de chaves a serem usadas para encaminhamento de dados. Cada roteador, ao receber a estrutura, remove uma camada e envia para o próximo, até que chegue ao destino como texto em claro. Os pacotes enviados não trazem informações como origem e destino, assim o roteador sabe apenas a identidade do roteador de quem recebeu o pacote e do próximo.

A diferença em relação a Mix-Net é que os roteadores não usam técnicas *mixing*: os pacotes são recebidos e enviados, em seguida, para o próximo salto.

3.2.4 *Crowds*

REITER & RUBIN (1998, 1999) apresentam o sistema *Crowds*, cujo objetivo é prover anonimato dos usuários e de suas ações em sua navegação pela Internet. *Crowds* previne servidores Web e terceiros de aprenderem informações como endereço IP, domínio, plataforma computacional e páginas visitadas.

O sistema consiste em grupos de usuários, chamados *Crowd*, onde cada usuário executa um processo, chamado *jondo* (*John Do*), responsável por encaminhar re-

quisições Web de outros membros do grupo. Quando um usuário realiza uma requisição, esta não é enviada diretamente para o servidor Web, mas para outro membro do *Crowd*, que então, encaminha para o servidor ou para outro *jondo*. O objetivo é tornar anônima a identidade do usuário que solicitou a requisição, utilizando vários *jondos*.

3.3 Anonimato e Redes Sem Fio Ad Hoc Móveis

Esta seção descreve as características, os objetivos e o funcionamento de protocolos de roteamento anônimo propostos para redes sem fio ad hoc móveis.

Para descrever o funcionamento dos protocolos, as seguintes notações e terminologia são utilizadas:

- ID_X: identidade do nó X.
- A_X: endereço do nó X.
- *src*: *tag* pública que identifica o nó origem.
- *dest*: *tag* pública que identifica o nó destino.
- K_X: chave secreta do nó X (pode ser temporária) ou chave compartilhada por dois nós.
- PK_X: chave pública do nó X.
- SK_X: chave privada do nó X.
- TPK_X: chave pública temporária do nó X.
- TSK_X: chave privada temporária do nó X.
- $E_{K_X}(M)$: mensagem M cifrada com a chave secreta K_X (algoritmo criptográfico simétrico).
- $E_{PK_X}(M)$: mensagem M cifrada com a chave pública PK_X (algoritmo criptográfico assimétrico).
- $Sign_{SK_X}(M)$: assinatura digital da mensagem M com a chave privada do nó X.

- $H(M)$: *hash* da mensagem M.
- $H_K(M)$: HMAC - *Hashed Message Authentication Code* da mensagem M.
- Nó origem: nó que inicia o processo de requisição de rota.
- Nó destino: nó com o qual o nó origem pretende estabelecer uma rota.
- Nó intermediário: nó que participa dos processos de roteamento e de encaminhamento de dados. É responsável por processar os pacotes recebidos e encaminhá-los ou descartá-los, ou seja, funciona como roteador.
- Nó emissor ou transmissor: nó que envia um pacote. Pode ser inicial, quando origina o pacote, ou local, quando o encaminha.
- Nó destinatário ou receptor: nó para o qual um pacote deve ser enviado. Pode ser final, quando o pacote destina-se a ele, ou local, quando é o próximo nó no caminho entre origem e destino.

3.3.1 *ANonymous On Demand Routing* - ANODR

KONG & HONG (2003) propõem o primeiro protocolo de roteamento anônimo para redes sem fio ad hoc móveis, ANODR - *ANonymous On Demand Routing*. Os pesquisadores definem anonimato em termos de não-correlação e objetivam garantir duas propriedades - anonimato da rota e privacidade da localização contra adversários externos e internos. Antes de obter o modelo final, que é detalhado a seguir, os pesquisadores apresentaram três esquemas e analisaram suas limitações: ANODR-PO (*Public Key Protected Onion*), ANODR-BO (*Symetric Key based on Boomerang Onion*) e ANODR-TBO (*Trapdoor Boomerang Onion*). A principal diferença entre os esquemas é o tipo de estrutura *onion*.

ANODR é constituído por três fases: descoberta de rota, encaminhamento de dados e manutenção de rota. Quando um nó precisa descobrir uma rota, encaminha, por difusão, um pacote de requisição RREQ (*Route Request*), cujo formato é

$$[RREQ, seq, TPK_i, tr_{dest}, TBO_i],$$

onde *seq* é um número de seqüência único global, usado para identificar a requisição; TPK_i é uma chave pública temporária gerada pelo emissor i; tr_{dest} é um *trapdoor*,

que pode ser visto apenas pelo destino, igual a $[E_{K_{SD}}(dest, K_c), E_{K_c}(dest)]$; K_{SD} é uma chave compartilhada previamente pelos nós origem e destino; *dest* é uma *tag* pública que identifica o destino; K_c é um *nonce* aleatório (valor usado uma única vez); TBO_i é um *onion*. No caso da origem, TBO_i é igual a $[E_{K_S}(src)]$, onde K_S é uma chave simétrica temporária gerada pela origem; *src* é uma *tag* pública que identifica a origem.

Quando um nó intermediário recebe RREQ, verifica o número de seqüência *seq*, para descobrir se é a primeira vez que o recebe, e tenta abrir o *trapdoor*, para saber se é o destino. Se o pacote não foi recebido anteriormente e como não é o destino, armazena *seq*, TPK_i, tr_{dest} e TBO_i, gera uma chave pública temporária TPK_{i+1} e substitui no lugar correspondente. Além disso, acrescenta um *nonce* N_{i+1} à estrutura onion e cifra com uma chave simétrica aleatória K_{i+1}; por exemplo, supondo que o primeiro nó intermediário é A, o resultado é $[E_{K_A}(N_A, E_{K_S}(src))]$. A chave e o *nonce* constituem um *trapdoor*, conhecido apenas pelo intermediário. Então, encaminha RREQ.

O nó destino, ao receber RREQ, encaminha um pacote de resposta RREP (*Route Reply*), cujo formato é

$$[RREP, E_{TPK_i}(K_{i+1}), E_{K_{i+1}}(pr_{dest}, TBO_i)],$$

onde TPK_i é a chave pública temporária do próximo nó na rota reversa, recebida no pacote RREQ; K_{i+1} é uma chave escolhida pelo emissor para ser compartilhada com o próximo nó, é utilizada por eles para a geração de uma seqüência de pseudônimos de rota, que são usados na fase de encaminhamento dos dados; pr_{dest} é a prova de que o destino recebeu e verificou tr_{dest}, é igual a K'_c; TBO_i é o *onion* recebido no pacote RREQ. Para cada requisição distinta, pode-se encaminhar uma resposta.

Quando um nó recebe RREP, tenta decifrar $E_{TPK_i}(K_{i+1})$, $E_{K_{i+1}}(pr_{dest}, TBO_i)$ e TBO_i, para verificar se faz parte da rota. Se faz, verifica se $K'_c(dest)$ é igual a $K_c(dest)$ da requisição, gera a chave a ser compartilhada com o próximo nó K_i, armazena K_{i+1} e K_i. Com esta, cifra pr_{dest} e o *onion* sem a primeira camada; com a chave pública temporária do próximo nó na rota reversa, recebida no pacote RREQ, cifra K_i; substitui os resultados nos locais correspondentes e encaminha, por difusão, o pacote RREP modificado.

O nó origem, ao receber RREP, pode encaminhar os pacotes de dados, cujo

formato é

$$[DATA, n_i, E_{K_i}(data)],$$

onde n_i é um pseudônimo de rota, correspondente à chave secreta K_i, compartilhada com o próximo salto; *data* são os dados a serem transmitidos; podem ser protegidos por protocolos fim a fim. Quando um nó recebe DATA, verifica o pseudônimo para saber se é o destinatário pretendido. Se é, decifra *data*, procura a chave e o pseudônimo compartilhados com o próximo salto, cifra os dados com essa chave e substitui n_i. O processo é repetido até que se chegue ao destino. Em relação à manutenção da rota, há um tempo de expiração para as rotas e assume-se que os nós são capazes de detectar quebras de enlaces, quando o número de retransmissões excede um determinado limiar. Quando um nó detecta uma quebra, verifica em sua tabela a chave K_i associada a K_{i+1} do nó inativo (ou que se moveu para fora de seu raio de alcance) e encaminha um pacote RERR (*Route Error*), cujo formato é $[RERR, K_i]$. Este pacote é modificado e encaminhado por cada nó até ser recebido pela origem.

Para evitar ataques de análise de tráfego, como análise de tempo e inundação, sugere-se a utilização de técnicas *mixing*, como armazenamento e reordenação dos pacotes de resposta e de dados em um *buffer* e inserção de pacotes falsos, caso necessário. Para esconder o tamanho atual dos pacotes de requisição e de resposta, cada nó acrescenta um *padding* aleatório ao *onion*. Para evitar que um adversário siga um pacote, a carga útil dos dados é cifrada salto a salto com a chave secreta compartilhada pelos nós consecutivos. Como limitação, os autores citam: se um nó X está comprometido, adversários podem associar dois pseudônimos para cada rota através de X. Se F nós são comprometidos e consecutivos, podem associar um segmento de rota de F+1 nós. Se não são consecutivos, então podem formar segmentos de rotas, mas não conseguem associá-los.

Os esquemas são analisados através de simulação e comparados aos protocolos AODV e DSR. Os aspectos avaliados são: *untraceability* (incapacidade de seguir um fluxo) em termos de tolerância a intrusão, desempenho e impacto do uso de técnicas *mixing*. Em relação a *untraceability*, os resultados indicam que, no caso do DSR, *untraceability* aumenta quando o tamanho da rota aumenta e que, no caso do ANODR, o aumento de saltos não influi. Em relação ao desempenho, ANODR-TBO apresenta resultados inferiores apenas em relação ao AODV. Em relação ao impacto

das técnicas *mixing*, os resultados indicam que a entrega de pacotes verdadeiros não é afetada e que essas técnicas podem ser utilizadas em redes sem fio ad hoc móveis, se valores apropriados para o tamanho da janela e do *buffer* são selecionados.

3.3.2 *Secure Distributed Anonymous Routing* - SDAR

EL-KHATIB et al. (2003) e BOUKERCHE et al. (2004a, 2004b, 2004c) propõem o protocolo SDAR - *Secure Distributed Anonymous Routing*, cujo objetivo é permitir que nós intermediários confiáveis participem do processo de descoberta de rotas, sem comprometer o anonimato dos nós origem e destino. Provê um mecanismo de gerenciamento de confiança entre os nós e é constituído por três fases: descoberta de rota, rota reversa e transferência de dados.

O mecanismo de gerenciamento de confiança é responsável por estabelecer e atualizar os níveis de confiança entre os nós. Um nó (chamado central) e todos os seus vizinhos formam a sua comunidade. Cada nó central define dois valores, δ_1 e δ_2, e estabelece três níveis de confiança, baixo (entre 0 e δ_1), médio (entre δ_1 e δ_2) e alto (entre δ_2 e 1). Todo nó possui uma lista de vizinhos e, periodicamente, envia uma mensagem *hello* com sua chave pública. Para cada vizinho detectado, o nó central envia a chave da comunidade, cifrada com a chave pública desse vizinho, HTLCK (*High Trust Level Community Key*) ou MTLCK (*Medium Trust Level Community Key*), de acordo com a confiança estabelecida. Os vizinhos com nível alto de confiança possuem as duas chaves, enquanto os de baixo não possuem chaves. Normalmente, quando o vizinho é detectado pela primeira vez, o nível é configurado como médio e de acordo com a interação, é atualizado. Um nó é considerado malicioso quando modifica ou não encaminha os pacotes, e um nó é confiável quando se comporta de acordo com a especificação do protocolo.

Quando um nó precisa descobrir uma rota, encaminha um pacote de descoberta, constituído por cinco partes. Todo o pacote, com exceção de sua primeira parte, é cifrado com a chave da comunidade, correspondente ao nível de confiança desejado. A primeira parte possui o formato

$$[RREQ, TRUST\text{-}REQ, TPK_S],$$

onde *TRUST-REQ* especifica o nível de confiança que os intermediários devem possuir; TPK_S é uma chave pública temporária gerada pela origem e usada pelos

intermediários para cifrar informações de roteamento adicionadas ao pacote, serve também como identificador do pacote. A segunda parte possui o formato

$$[E_{PK_D}(ID_D, K_{SD}, PL_S)],$$

onde PK_D e ID_D são, respectivamente, a chave pública e a identidade do destino; K_{SD} é uma chave simétrica gerada pela origem e usada para cifrar a quarta parte e evitar ataques de repetição; PL_S é o tamanho da terceira parte, P_S, que é um *padding*. A quarta parte possui o formato

$$[E_{K_{SD}}(ID_S, PK_S, TPK_S, TSK_S, SN_{Session\_ID_S}), Sign_{SK_S}(M_S)],$$

onde ID_S e PK_S são, respectivamente, a identidade e a chave pública da origem; TPK_S e TSK_S são chaves pública e privada temporárias geradas pela origem e usadas pelo destino para decifrar e verificar as informações de roteamento adicionadas pelos intermediários; $SN_{Session\_ID_S}$ é um número aleatório gerado pela origem, que identifica a sessão; $Sign_{SK_S}(M_S)$ é o *hash* do pacote, assinado com a chave privada da origem; M_S é igual a $[H(TYPE,\ TRUST\text{-}REQ, TPK_S, TSK_S, ID_D, K_{SD}, ID_S, PK_S, SN_{Session\_ID_S}, PL_S, P_S)]$. A quinta parte é constituída pelas informações adicionadas pelos intermediários.

Quando um nó intermediário recebe RREQ, verifica TPK_S, para saber se é a primeira vez que o recebe. Se é, verifica se possui a chave de comunidade correspondente, decifra o restante do pacote e verifica se é o destino (tenta decifrar $E_{PK_D}(ID_D, K_{SD}, PL_S)$). Como não é, adiciona uma entrada, cujo formato é

$$[E_{TPK_S}(ID_i, K_i, SN_{Session\_ID_i}, Sign_{SK_i}(M_i))],$$

onde ID_i é a identidade do intermediário i; K_i é uma chave simétrica gerada por i; $SN_{Session\_ID_i}$ é um número aleatório para identificar a sessão; $Sign_{SK_i}(M_i)$ é o *hash* do pacote, assinado com a chave privada de i; M_i é igual a $[H(M_{i-1},\ ID_i, K_i,\ SN_{Session\_ID_i})]$. Armazena TPK_S e cifra as quatro últimas partes do pacote com a chave de comunidade correspondente ao nível de confiança estabelecido e re-encaminha o pacote.

O nó destino, ao receber RREQ, decifra a segunda parte e descobre o início da quarta. Com K_{SD}, decifra a quarta parte, obtém TSK_S, decifra as informações anexadas pelos intermediários e obtém as chaves geradas por eles. Então, reenvia a

requisição recebida e gera um pacote de resposta com várias camadas, cifradas com as chaves da origem e dos intermediários. A camada cifrada com a chave da origem contém todas as chaves e identificadores de sessão de cada nó, um *padding* e seu tamanho. As demais camadas contêm identificadores de sessão e funções *hash*. Esse pacote possui o formato

$$
\begin{aligned}
[&RREP,\ E_{K_i}(E_{K_{i-1}}(E_{K_{i-2}}\ldots(E_{K_2}(E_{K_1}(E_{K_{SD}}(\\
&SN_{Session\_ID_1}, K_1, \ldots, SN_{Session\_ID_i}, K_i, SN_{Session\_ID_D}, PL_D, P_D),\\
&SN_{Session\_ID_S}, SN_{Session\_ID_{S-1}}, H(P), H_{K_{SD}}(N_S)),\\
&SN_{Session\_ID_1}, SN_{Session\_ID_{S-1}}, H(P), H_{K_1}(N_1)),\\
&SN_{Session\_ID_2}, SN_{Session\_ID_S}, H(M_S), H_{K_2}(N_2)), \ldots,\\
&SN_{Session\_ID_{i-2}}, SN_{Session\_ID_{i-4}}, H(M_{i-4}), H_{K_{i-2}}(N_{i-2})),\\
&SN_{Session\_ID_{i-1}}, SN_{Session\_ID_{i-3}}, H(M_{i-3}), H_{K_{i-1}}(N_{i-1})),\\
&SN_{Session\_ID_i}, SN_{Session\_ID_{i-2}}, H(M_{i-2}), H_{K_i}(N_i)],
\end{aligned}
$$

onde $SN_{Session\_ID_{S-1}}$ é um número aleatório gerado pelo destino, com o mesmo número de bits de qualquer $SN_{Session\_ID_i}$; P é um *padding* com mesmo tamanho de qualquer M_i; M_i é igual a $[E_{K_{i-1}}(M_{i-2})$, $SN_{Session\_ID_i}$, $SN_{Session\_ID_{i-2}}$, $H(M_{i-2})$, $H_{K_i}(N_i)]$; N_i é igual a $[E_{K_i}(M_{i-1})$, $SN_{Session\_ID_i}$, $SN_{Session\_ID_{i-2}}$, $H(M_{i-2})]$; H_{K_i} (N_i) é o HMAC de N_i; PL_D é o tamanho do *padding* P_D. Cada intermediário que recebe o pacote usa $SN_{Session\_ID_i}$ para descobrir a chave correspondente, remove uma camada e encaminha o pacote para o próximo nó na rota reversa.

Na fase de transferência de dados, a origem cifra os dados com as chaves dos intermediários e cada nó, ao longo da rota, remove uma camada e encaminha o pacote de dados, até chegar ao destino (este encaminha também), cujo formato é

$$[DATA, SN_{Session\_ID_i}, E_{K_i}(SN_{Session\_ID_{i+1}}, E_{K_{i+1}}\ldots(E_{K_{SD}}(data)))],$$

onde $SN_{Session\_ID_i}$ e K_i são, respectivamente, o identificador de sessão e a chave do próximo salto; *data* são os dados a serem transmitidos.

De acordo com os autores, SDAR é seguro contra ataques passivos e ativos, exceto negação de serviço, mantém anonimato dos nós origem e destino, é capaz de identificar nós maliciosos e evitá-los, é capaz de estabelecer rotas confiáveis e é resistente a seqüestro de rotas. Entre as limitações, citam-se a ausência de controle do tamanho da rota e o alto poder computacional exigido. Os resultados de simulação

indicam que SDAR é uma boa solução para prover anonimato, com custo razoável, quando comparado ao DSR.

3.3.3 *Anonymous Secure Routing* - ASR

ZHU et al. (2004) propõe o protocolo ASR - *Anonymous Secure Routing*, cujo objetivo é prover anonimato da identidade, privacidade forte da localização, anonimato da rota e segurança do protocolo contra ataques ativos e passivos. ASR é constituído por quatro fases: requisição e resposta de rota, transmissão de dados e manutenção de rota.

Quando um nó precisa descobrir uma rota, encaminha, por difusão, um pacote de requisição, cujo formato é

$$[RREQ, seq, E_{K_{SD}}(ID_D, K_S, U_0), E_{K_S}(seq, END), TPK_i, U_i],$$

onde *seq* é o número de seqüência da sessão atual; K_{SD} é a chave secreta compartilhada previamente pelos nós origem e destino; ID_D é a identidade do destino; K_S é uma chave simétrica temporária gerada pela origem; U_0 é um número aleatório escolhido pela origem; *END* é um sinal utilizado pelo destino para confirmar o recebimento da requisição; TPK_i é uma chave pública temporária gerada pelo emissor i; U_i é um número aleatório calculado por i ($U_i = (U_{i-1} \oplus S_i) >> p_x$, $p_s = (H_{max}+1)p_x$; S_i é um valor computado por i, com tamanho p_x; p_s é o tamanho de U_0; H_{max} é o número máximo desejado de nós entre a origem e o destino).

Quando um nó intermediário recebe RREQ, verifica se já o recebeu anteriormente (por meio de *seq*). Se não o recebeu, tenta decifrar $E_{K_{SD}}(ID_D, K_S, U_0)$, para verificar se é o destino. Como não é, armazena *seq*, $E_{K_S}(seq, END)$ e TPK_i, gera uma chave pública temporária TPK_{i+1} e um número aleatório U_{i+1}, substitui TPK_i e U_i por TPK_{i+1} e U_{i+1} e envia o pacote modificado por difusão.

O nó destino, ao receber RREQ, decifra $E_{K_{SD}}(ID_D, K_S, U_0)$, compara U_0 e U_i, a fim de obter o tamanho da rota. Para todas as rotas com tamanho menor ou igual a H_{max}, é criado um pacote de resposta, cujo formato é

$$[RREP, E_{TPK_i}(K_{i+1}), E_{K_{i+1}}(seq, K'_s)],$$

onde TPK_i é a chave pública temporária do próximo nó na rota reversa, recebida

no pacote RREQ; K_{i+1} é um número aleatório escolhido pelo emissor, que é usado, durante a transmissão dos dados, como chave secreta por ele e pelo próximo nó; *seq* é o número de seqüência da RREQ; K'_s é a prova de que o destino decifrou o terceiro elemento do pacote *RREQ*.

Quando um nó recebe RREP, tenta decifrar $E_{TPK_i}(K_{i+1})$ e $E_{K_{i+1}}(seq, K'_s)$, para descobrir se faz parte da rota. Verifica se $E_{K'_S}(seq, END)$ é igual a $E_{K_S}(seq, END)$, para confirmar que o pacote foi enviado pelo destino. Se sim, escolhe um número aleatório K_i, armazena K_i e K_{i+1}, calcula $E_{TPK_{i-1}}(K_i)$ e $E_{K_i}(seq, K'_s)$ e substitui os dois últimos elementos do pacote, que é então enviado por difusão.

Durante a transmissão de dados, é enviado um pacote, cujo formato é

$$[DATA, TAG, E_{K_i}(data)],$$

onde TAG é igual a [N, $H_{K_i}(N)$]; N é um número escolhido pelo emissor e, é aumentado a cada pacote recebido ou enviado por esta rota; K_i é a chave secreta compartilhada pelo emissor e pelo destinatário, que foi estabelecida na fase de resposta; *data* são os dados a serem transmitidos. Quando um nó recebe $DATA$, verifica TAG, para saber se é o destinatário pretendido e se foi enviado pelo nó correto. Calcula o novo valor de TAG, decifra os dados e os cifra com a chave compartilhada com o próximo nó e encaminha o pacote modificado por difusão. Quando um nó detecta quebra de enlace (número de retransmissões excede um limiar), envia um pacote [$RERR$, TAG], onde TAG é a informação compartilhada com o nó anterior.

O protocolo é analisado intuitivamente e comparado aos protocolos ANODR e SDAR. De acordo com os pesquisadores, ASR provê privacidade da identidade, privacidade forte da localização, em relação a nós pertencentes ou não à rota, e anonimato da rota, além de prevenir negação de serviço, *wormhole* e ataques à manutenção de rota. Ataques de análise de tempo são prevenidos com a utilização de técnicas *mixing* (envio reordenado dos pacotes de dados e envio de pacotes falsos). SDAR não provê privacidade das identidades dos nós intermediários presentes na rota nem privacidade forte da localização em relação a nós fora e a pertencentes à rota. ANODR não provê privacidade da identidade do destino e privacidade forte da localização, em relação a nós pertencentes à rota.

3.3.4 MASK

ZHANG et al. (2004, 2005) propõe o protocolo MASK, cujo objetivo é prover anonimato dos nós origem e destino e da sua relação, *untraceability* e *unlocatability* (adversários não podem seguir um pacote até origem ou destino), autenticação segura e anônima entre vizinhos, custo criptográfico baixo, roteamento eficiente e resistência a um grande número de ataques. É constituído por quatro fases: configuração do sistema, autenticação dos vizinhos, descoberta de rota e encaminhamento de dados.

Durante a fase de configuração, uma autoridade confiável determina dois grupos de ordem q, $\mathbb{G}_1$ e $\mathbb{G}_2$; uma função de mapeamento bilinear com curva elíptica, f: $\mathbb{G}_1$ x $\mathbb{G}_1 \rightarrow \mathbb{G}_2$; uma chave master do sistema, $g \in \mathbb{Z}_q^*$; duas funções hash, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ (mapeamento arbitrário de *strings* para pontos em $\mathbb{G}_1$) e $H_2 : \{0,1\}^* \rightarrow \{0,1\}^\beta$ (mapeamento arbitrário de *strings* para *strings* de tamanho fixo β). Todos os nós são informados sobre estes parâmetros, com exceção da chave master. A autoridade provê cada nó com pares secretos de pseudônimos $\mathcal{PS}_i$ e pontos $\mathcal{S}_i = gH_1(\mathcal{PS}_i) = \{S_{i,j}\} = \{gH_1(PS_{i,j}) \in \mathbb{G}_1\}(1 \leq j \leq |\mathcal{PS}_i|)$.

O protocolo de autenticação anônima de vizinhos é utilizado para estabelecer chaves secretas e identificadores de enlace. Quando um nó A se move, escolhe um pseudônimo ainda não utilizado (por exemplo, $PS_{A,4}$) do seu conjunto $\mathcal{PS}_A$ e o envia por difusão local com um *nonce* aleatório n_1. Quando um nó B recebe o pacote e concorda com o *handshake*, usa seu pseudônimo ativo ($PS_{B,5}$) para calcular a chave master de sessão $K_{BA} = f(S_{B,5}, H_1(PS_{A,4}))$, onde $S_{B,5} = gH_1(PS_{B,5})$ é o ponto secreto correspondente a seu pseudônimo $PS_{B,5}$. O nó B envia, por difusão, um pacote de resposta, que contém $PS_{B,5}$, um *nonce* aleatório n_2 e um autenticador $V_B = H_2(K_{BA}||PS_{A,4}||PS_{B,5}||n_1||n_2||0)$. Ao receber a resposta de B, A pode calcular a chave master $K_{A,B} = f(H_1(PS_{B,5}), S_{A,4})$, onde $S_{A,4} = gH_1(PS_{A,4})$ é o ponto secreto correspondente a seu pseudônimo $PS_{A,4}$. A equação $K_{AB} = K_{BA} = f(H_1(PS_{B,5}), H_1(PS_{A,4}))^g \in \mathbb{G}_2$ deve ser satisfeita. O nó A pode autenticar B verificando V_B. A fim de B autenticar A, este retorna seu autenticador $V_A = H_2(K_{AB}||PS_{A,4}||PS_{B,5}||n_1||n_2||1)$. Após a autenticação, ambos podem calcular pares de chaves de sessão e identificadores de enlace (*SKey*, *LinkID*): $K_{AB}^\gamma = H_2(K_{AB}||PS_{A,4}||PS_{B,5}||n1||n2||2 * \gamma)$ e $L_{AB}^\gamma = H_2(K_{AB}||PS_{A,4}||PS_{B,5}||n1||n2||2 * \gamma + 1)$, onde ($K_{AB}^\gamma$, L_{AB}^γ) é o par de número γ, $(1 \leq \gamma \leq \Gamma)$, Γ é um parâmetro de

projeto. Após autenticar seus vizinhos, o nó A cria uma tabela de vizinhos, onde cada entrada contém o pseudônimo do vizinho, os pares (*SKey*,*LinkID*) e o índice γ do par atual. Assim, quando o nó B enviar um pacote identificado por L_{AB}^{γ}, A sabe que o pacote destina-se a ele. Quando os pares acabam, os nós devem aumentar *nonces* e gerar novos pares.

A descoberta de rota é constituída por duas fases: requisição e resposta de rota. Durante a requisição, é difundido um pacote de requisição, cujo formato é

$$[RREQ, seq, ID_D, destSeq, PS_i],$$

onde *seq* identifica o pacote RREQ; ID_D é o identificador do destino; *destSeq* é o número de seqüência da última rota conhecida pela origem; PS_i é o pseudônimo ativo do nó emissor i.

Nós intermediários que não satisfazem RREQ (não possuem rota válida para o destino), ao recebê-lo, armazenam *seq*, ID_D, *destSeq* e PS_i, substituem o pseudônimo pelo seu e reenviam o pacote. Todos os nós possuem uma tabela de encaminhamento de rotas, cujas entradas possuem o formato [ID_D, *destSeq*, *pre-link-list*, *next-link-list*], onde *pre-link-list* contém os pseudônimos dos nós dos quais podem vir pacotes destinados ao nó, cujo identificador é ID_D; *next-link-list* contém os pseudônimos dos nós para os quais os pacotes destinados a ID_D devem ser encaminhados. Todos os nós, inclusive o destino, reenviam RREQ.

O nó destino, ao receber RREQ, envia, por *unicast*, um pacote de resposta, cujo formato é

$$[LinkID, E_{SKey}(RREP, ID_D, destSeq)],$$

onde *LinkID* é o próximo identificador de enlace que o emissor compartilha com o próximo nó na rota reversa, isto é, com o nó do qual recebeu a requisição; $SKey$ é a chave correspondente a *LinkID*; *destSeq* é o número de seqüência que identifica a rota descoberta. Intermediários que possuem rotas válidas (*destSeq* maior ou igual ao contido no RREQ) também podem enviar respostas.

Quando um nó recebe RREP, verifica se *LinkID* pertence à sua tabela de vizinhos. Se sim, decifra $E_{SKey}(RREP, ID_D, destSeq)$ e verifica se *destSeq* é menor do que o que está na sua tabela de encaminhamento de rotas. Se não é, transmite um novo RREP (substitui *LinkID* pelo próximo identificador de enlace compartilhado

com o próximo nó na rota reversa; cifra [*RREP*, ID_D, *destSeq*] com a chave *SKey* correspondente ao novo *LinkID*). Também atualiza sua tabela de encaminhamento (se não possui entrada para ID_D, cria uma; se *destSeq* armazenado é menor do que o contido em RREP, atualiza; se possui *destSeq* igual, atualiza tabela de encaminhamento inserindo $LinkID$ do anterior e do próximo em *pre-link-list* e *next-link-list*, respectivamente).

Durante o encaminhamento de dados, são enviados pacotes, cujo formato é

$$[DATA, \textit{next-LinkID}, \textit{MASK payload}],$$

onde *next-LinkID* é o identificador de enlace compartilhado pelo emissor e o próximo salto; *payload* contém os dados, pode ser cifrado para proteger integridade e/ou com $SKey$ compartilhado pelos vizinhos. Nós, ao longo da rota (verificam se *next-LinkID* lhe pertence), substituem *next-LinkID* por um valor escolhido aleatoriamente em *next-link-list*. Também acrescentam *paddings* aleatórios à carga útil. O identificador de enlace pode ser mudado dinamicamente (após determinado período ou a cada pacote encaminhado) pelos nós que o compartilham.

A análise de MASK é feita intuitivamente e através de simulação. Segundo os pesquisadores, MASK provê anonimato incondicional dos nós emissor e destinatário e de sua relação, anonimato incondicional da origem e dos intermediários, anonimato condicional do destino e *unlocatability*. É resistente a ataques de conteúdo de pacote, de reconhecimento de fluxo, de repetição e de análise de tempo (quando a carga é baixa, pode-se usar técnicas como envio de pacotes falsos e retardo aleatório para envio dos pacotes recebidos). Quando os nós origem e destino estão cercados por nós comprometidos, não há como impedir análise de tempo; para amenizar esse problema, a solução é a mobilidade. Os resultados da simulação indicam que o protocolo possui desempenho similar ao protocolo AODV.

3.3.5 *Anonymous Dynamic Source Routing* - ANDSR

ARAUJO (2005) propõe o protocolo ANDSR - *Anonymous Dynamic Source Routing*. É uma extensão do protocolo DSR (JOHNSON; HU; MALTZ, 2007) e do trabalho de JIANG et al. (2001, 2004): o algoritmo de descoberta de nós misturadores (*Closest Mix*) é acrescentado ao processo de descoberta de rota do DSR.

Quando um nó deseja comunicar-se anonimamente com outro, mas não possui uma rota válida e não conhece nós misturadores, envia, por difusão, um pacote MREQ (*Mix Request*), cujo formato é [*MREQ*, A_S, A_B, *seq*], onde A_S é o endereço da origem; A_B é o endereço de difusão; *seq* é um número para identificar a requisição. Os nós que não são misturadores, ao receberem a requisição, descartam-na. Já os misturadores enviam um pacote MREP (*Mix Reply*), cujo formato é [*MREP*, A_M, A_S, *seq*], onde A_M é o endereço do misturador; A_S é o endereço da origem; *seq* é o número de seqüência, contido no pacote MREQ.

O nó origem, ao receber o primeiro MREP, envia um pacote de requisição, cujo formato é

$$[RREQ, E_{PK_M}(ident, R), E_{PK_M}(A_D, R), E_{PK_D}(A_S, R), A_X,$$

$$path(E_{PK_S}(A_S, R), \ldots)],$$

onde PK_M é a chave pública dos misturadores; *ident* é um valor gerado pela origem, que identifica a requisição; A_D e PK_D são, respectivamente, o endereço e a chave pública do destino; A_S é o endereço da origem; A_X é o endereço do próximo salto (no caso do pacote enviado pela origem, é o endereço do misturador do qual recebeu o pacote *Mix Reply*); $path(E_{PK_S}(A_S, R), \ldots)$ contém os endereços da origem e dos misturadores intermediários (são acrescentados ao longo da rota); R é um valor aleatório.

Quando um nó misturador recebe RREQ, decifra o endereço do destino e verifica se é seu vizinho. Caso não seja, cifra novamente o endereço com PK_M (assume-se que os misturadores compartilham uma chave) e envia o pacote, por difusão, até que seja recebido por um misturador vizinho ao destino. Este misturador cifra o endereço do destino com sua chave pública PK_D e envia o pacote por difusão.

Quando RREQ chega ao destino, este gera um pacote RREP, cujo formato é

$$[RREP, E_{PK_M}(ident, R), E_{PK_S}(A_D, R), E_{PK_S}(A_S, R), A_X, path(E_{PK_S}(A_D, R),$$

$$A_X, \ldots, E_{PK_S}(A_S, R))],$$

onde PK_M é a chave pública dos misturadores; *ident* identifica a requisição a que se refere a resposta; PK_S é a chave pública da origem; A_D é o endereço do destino; A_S é o endereço da origem; A_X é o endereço do próximo salto (no caso do pacote

enviado pelo destino, é o endereço do misturador do qual recebeu o pacote RREQ); $path(E_{K_S}(A_D), A_M, \ldots, E_{PK_S}(A_S))$ indica o caminho reverso até a origem; R é um valor aleatório. Esse pacote é encaminhado pela rota reversa até chegar ao último misturador, que o envia por difusão, pois o endereço da origem está cifrado.

Após receber RREP, o nó origem pode enviar os pacotes de dados, cujo formato é

$$[DATA, E_{PK_D}(A_S, R), E_{PK_M}(A_D, R), A_X, E_{PK_D}(data, R)],$$

onde PK_D é a chave pública do destino; A_S é o endereço da origem; R é um valor aleatório; PK_M é a chave pública dos misturadores; A_D é o endereço do destino; A_X é o endereço do próximo salto na rota; *data* são os dados a serem transmitidos. Quando esse pacote chega ao misturador vizinho ao destino, ele decifra $E_{PK_M}(A_D, R)$ e substitui por $E_{PK_D}(A_D, R)$, antes de encaminhá-lo por difusão.

No caso de quebra de enlace, o misturador deve enviar um pacote *route error* para o nó do qual recebeu um pacote RREQ e assim, sucessivamente, até o pacote chegar ao misturador vizinho à origem (o último misturador envia por difusão).

ANDSR é especificado em LOTOS - *Language of Temporal Ordering Specification*, verificado e validado formalmente (ausência de *deadlocks* - impasses e *livelocks* - estados não alcançados) com o pacote de ferramentas CADP - *CAESAR/ALDEBARAN Development Package* (VASY-INRIA, 2007). O algoritmo é especificado de forma genérica, ou seja, sem considerar topologia e quantidade de nós. São modelados grupos de nós: origem, misturador, destino, comum e intruso. Realiza dois estudos de caso, nos quais, há um nó comum intruso e um conluio de misturadores. No primeiro caso, o intruso não obtém informações e no segundo, descobrem a identidade do destino e possuem suspeitas quanto à origem, mas não as confirmam. Segundo ARAUJO (2005), provê anonimato dos nós origem e destino e dificulta ataques de análise de tráfego.

3.3.6 *Certificate-free Anonymous Routing Protocol* - CARP

BANERJEE et al. (2006) propõe o protocolo CARP - *Certificate-free Anonymous Routing Protocol*, cujo objetivo é prover anonimato da identidade, da localização e da rota. Utiliza IBE - *Identity Based Encryption* para geração de chaves

(a identidade de um nó é utilizada como sua chave pública e a chave privada correspondente é gerada por um servidor de chaves), e é constituído por quatro fases: descoberta e resposta de rota, transferência de dados e manutenção.

Quando um nó precisa descobrir uma rota, encaminha, por difusão, um pacote de requisição, cujo formato é

$$[RREQ, TTL, r.P, \textit{route-request-token}, E_K(m_S), \textit{node-pseudonym-list}],$$

onde TTL é o número máximo de saltos que a requisição pode percorrer; $r.P$ é o produto de r (número aleatório escolhido pela origem) e P (ponto público da curva elíptica); *route-request-token* é o identificador da requisição; K é a chave criptográfica IBE, gerada pela origem usando a identidade do destino, os parâmetros da curva elíptica (P e $s.P$, s é a chave secreta mestre, conhecida somente pelo servidor de chaves) e a função *pairing*; m_S é igual a [ID_S, ID_D, $nonce_S$, *init-vector*] (ID_S é a identidade da origem; ID_D é a identidade do destino; $nonce_S$ é um número aleatório escolhido pela origem; *init-vector* é o vetor de inicialização, utilizado pelas operações criptográficas); *node-pseudonym-list* é a lista de pseudônimos dos nós na rota (são acrescentados ao longo da rota). Com $nonce_S$, cifra *init-vector*; o resultado é usado na fase de resposta, para conferir se o pacote foi enviado pelo destino.

Quando um nó intermediário recebe RREQ, verifica se é a primeira vez que o recebe (por meio de *route-request-token*). Com sua chave privada, $r.P$ e os parâmetros da curva elíptica, tenta recuperar K e decifrar $E_K(m_S)$, para verificar se é o destino. Se não foi recebido anteriormente e como não é o destino, decrementa TTL, acrescenta um pseudônimo único ao campo *node-pseudonym-list* e re-encaminha RREQ.

O nó destino, ao receber RREQ, encaminha, por difusão, um pacote de resposta RREP, cujo formato é

$$[RREP, TTL, \textit{route-reply-token}, uid, E_{nonce_S}(m_D), \textit{node-pseudonym-list}],$$

onde TTL é o número máximo de saltos que a resposta pode percorrer; *route-reply-token* é o identificador da resposta; *uid* é igual a E_{nonce_S}(*init-vector*); m_D é igual a [ID_S, ID_D, $nonce_D$] ($nonce_D$ é um número aleatório escolhido pelo destino; o vetor de inicialização utilizado é *init-vector*); *node-pseudonym-list* contém os pseudônimos de todos os nós na rota, inclusive o do destino. Este envia RREP, até receber um pacote RREPACK. Gera a chave de sessão $K_{session}$, através do *hash* de [ID_S, ID_D,

$nonce_S$, $nonce_D$].

Quando um nó recebe RREP, verifica *node-pseudonym-list*, para saber se faz parte da rota. Se faz, verifica se já o encaminhou anteriormente, através de *route-reply-token*. Se não, decrementa TTL e o reenvia. Além disso, envia RREPACK [$RREPACK$, $TTL = 1$, *route-reply-token*]. Quando os nós recebem esse pacote, não encaminham o RREP correspondente.

O nó origem, ao receber RREP, verifica *uid* e $E_{nonce_S}(m_D)$, para saber se RREP foi enviado pelo destino. Gera a chave de sessão $K_{session}$ do mesmo modo que o destino. Então, pode encaminhar os pacotes de dados, cujo formato é

$$[DATA, TTL, \textit{data-token}, \textit{uid}, E_{K_{session}}(\textit{data}), \textit{node-pseudonym-list}],$$

onde TTL é o número máximo de saltos que o pacote pode percorrer; *data-token* é o identificador do pacote; uid é copiado de RREP; *data* são os dados a serem transmitidos; *node-pseudonym-list* é copiado de RREP.

Quando um nó recebe um pacote de dados, verifica *node-pseudonym-list* e *data-token*, para saber se pertence à rota e se não o processou anteriormente. Se pertence e se é um pacote atual, verifica *uid*, para saber se é o destino pretendido. Se não é, decrementa TTL e re-encaminha. Além disso, envia DATAACK, para evitar que outros nós o re-encaminhem desnecessariamente. O destino, ao receber $DATA$, recupera $K_{session}$ e decifra os dados. Após encaminhar DATA, os nós esperam por um DATAACK. Caso não o recebam, enviam um RRER, [$RERR$, *data-token*, *node-pseudonym-list*].

CARP é analisado através de simulação e comparado ao DSR padrão e otimizado. Os aspectos avaliados são: taxa de entrega dos pacotes, *overhead* de pacotes e de *bytes* e latência de dados. Os resultados indicam que a taxa de entrega é menor do que a do DSR otimizado, mas possui resultado semelhante em relação ao DSR padrão. O *overhead* e a latência são maiores, mas são valores aceitáveis.

3.3.7 *Anonymous Secure Routing Protocol* - ASRP

CHENG & AGRAWAL (2005) propõem o protocolo ASRP - *Anonymous Secure Routing Protocol*, cujo objetivo é prover segurança e anonimato da identidade e da localização, com bom desempenho. É constituído por três fases: requisição e

resposta de rota e transmissão de dados.

Quando um nó precisa descobrir uma rota, encaminha, por difusão, um pacote de requisição, cujo formato é

$$[RREQ, seq, PID_i, TPK_i, E_{PK_D}(ID_S, ID_D, N_S, K_{SD}), TPK_S, E_{TPK_S}(N_S)],$$

onde *seq* é o número de seqüência que identifica o pacote; PID_i e TPK_i são o pseudônimo aleatório e a chave pública temporária do emissor i; PK_D é a chave pública do destino; ID_S e ID_D são, respectivamente, a identidade da origem e a do destino; N_S é um *nonce* aleatório gerado pela origem; K_{SD} é uma chave simétrica aleatória gerada pela origem e utilizada na fase de transmissão de dados; TPK_S é uma chave pública temporária gerada pela origem.

Quando um nó intermediário recebe RREQ, verifica *seq*, para descobrir se é a primeira vez que o recebe. Tenta decifrar $E_{PK_D}(ID_S, ID_D, N_S, K_{SD})$ com sua chave privada, para verificar se é o destino. Se não foi recebido anteriormente e como não é o destino, armazena *seq*, PID_i, TPK_i, TPK_S e $E_{TPK_S}(N_S)$, gera PID_{i+1} e TPK_{i+1}, substitui os campos correspondentes e re-encaminha RREQ.

O nó destino, ao receber RREQ, encaminha, por difusão, um pacote de resposta, cujo formato é

$$[RREP, PID_i, E_{TPK_i}(N_S, PID'_{i+1}, K_{i+1})],$$

onde PID_i e TPK_i são o pseudônimo e a chave pública temporária do próximo nó na rota reversa; N_S é o *nonce* recebido em RREQ; PID'_{i+1} e K_{i+1} são um novo pseudônimo e uma chave de sessão simétrica, a serem compartilhados com o próximo nó na rota reversa.

Quando um nó recebe RREP, verifica PID_i, para saber se é o próximo destinatário. Se é, tenta decifrar $E_{TPK_i}(N_S, PID'_{i+1}, K_{i+1})$ com sua chave privada temporária TSK_i, para autenticar o emissor. Verifica TPK_S e $E_{TPK_S}(N_S)$, para garantir que foi enviado pelo destino. Se foi, armazena PID'_{i+1} e K_{i+1}; gera um novo pseudônimo PID'_i e uma chave de sessão K_i, para serem compartilhados com o próximo nó na rota reversa; cifra [N_S, PID_i, K_i] com a chave pública temporária do destinatário TPK_{i-1}; recupera PID_{i-1}, armazenado durante a fase de requisição; substitui os campos correspondentes e re-encaminha RREP.

O nó origem, ao receber RREP, pode encaminhar os pacotes de dados, cujo

formato é

$$[DATA, PID'_i, N_{i-1}, E_{K_i}(N_{i-1}), E_{K_i}(E_{K_{SD}}(data))],$$

onde PID'_i é o pseudônimo do próximo nó na rota; N_{i-1} é um *nonce* aleatório gerado pelo emissor; K_i é a chave de sessão estabelecida, durante a fase de resposta, com o próximo nó; K_{SD} é a chave de sessão compartilhada pelos nós origem e destino; *data* são os dados a serem transmitidos.

Quando um nó recebe DATA, verifica PID'_i, para saber se é o destinatário pretendido. Se é, decifra $E_{K_i}(N_{i-1})$, para verificar se foi enviado pelo nó correto. Se sim, substitui PID'_i, N_{i-1}, $E_{K_i}(N_{i-1})$ e $E_{K_i}(E_{K_{SD}}(data))$ por PID'_{i+1}, N_i, $E_{K_{i+1}}(N_i)$ e $E_{K_{i+1}}(E_{K_{SD}}(data))$, respectivamente, e re-encaminha. O destino, ao receber DATA, decifra os dados com as chaves compartilhadas com o nó anterior e a origem.

ASRP é analisado intuitivamente em relação a anonimato e segurança. De acordo com os pesquisadores, provê anonimato da identidade, da localização e da rota, além de ser seguro contra a maioria dos ataques ativos e passivos.

3.3.8 *Anonymous Dynamic Source Routing Protocol* - AnonDSR

SONG et al. (2005) propõe o protocolo AnonDSR - *Anonymous Dynamic Source Routing Protocol*, cujo objetivo é prover anonimato, segurança e escalabilidade. É composto por três protocolos: estabelecimento de parâmetros de segurança, descoberta de rota e transferência de dados.

O protocolo de estabelecimento de parâmetros de segurança é utilizado para estabelecer a chave secreta e o *nonce* aleatório (índice secreto para identificar a chave) que serão usados pelos nós origem e destino em comunicações anônimas e seguras. Pode ser usado também para construir uma rota para comunicações não seguras.

O nó origem envia, por difusão, um pacote [*RREQ*, *SecType*, *seq*, ID_S, ID_D, *RRec*, *SecPara*], onde *SecType* pode ser não seguro, seguro ou anônimo; *seq* é um número de seqüência único global; ID_S e ID_D são, respectivamente, o identificador da origem e o do destino; *RRec* é um registro para armazenar a rota; *SecPara* contém parâmetros de segurança estabelecidos pela origem. Se *SecType* é seguro ou anônimo, *SecPara* é igual a $[E_{PK_D}(N_K, K, Para), Sign_{SK_S}(M)]$, onde PK_D é

a chave pública do destino; N_K é o índice secreto da chave secreta K; *Para* são parâmetros criptográficos, tais como algoritmo criptográfico e versão; $Sign_{SK_S}(M)$ é o *hash* do pacote, assinado pela origem; M é igual a $H(seq, ID_S, ID_D, N_K, K, Para)$. Os nós intermediários, ao receberem esse pacote pela primeira vez (verificam *seq*), inserem seu endereço em *RRec* e re-encaminham o pacote por difusão. O destino, ao recebê-lo, verifica *SecType*. Se é não seguro, inicia a fase RREP. Se é seguro ou anônimo, decifra *SecPara*, verifica $Sign_{SK_S}(M)$, armazena N_K, ID_S, K e *Para* e inicia a fase RREP.

Então, o nó destino envia, por difusão, o pacote [*RREP*, *SecType*, *seq*, ID_S, ID_D, *RRec*, *SecPara*], onde *RRec* é o registro com a rota inteira; *SecPara* são os parâmetros que o destino gostaria de mudar; os demais campos são copiados do pacote RREQ. Nós intermediários, ao recebê-lo, verificam se estão na rota e se não o receberam anteriormente. Se estão e não o receberam, re-encaminham. Se *SecType* é não seguro ou seguro, adicionam a rota em sua tabela.

O protocolo de descoberta de rota anônima é executado quando *SecType* é anônimo. Utiliza o esquema *onion routing* anônimo entre origem e destino, e para criar um *trapdoor*, utiliza o índice e a chave gerados na fase de estabelecimento de parâmetros de segurança.

O nó origem envia, por difusão, um pacote ANON-RREQ, cujo formato é

$$[ANON\text{-}RREQ, TPK_S, tr_{dest}, PDO_i],$$

onde TPK_S é uma chave pública temporária criada pela origem, funciona também como número de seqüência; tr_{dest} é um *trapdoor* criptográfico de chave simétrica, que pode ser aberto somente pelo destino ($[N_K, E_K(ID_D, TSK_S)]$, onde N_K é o índice secreto da chave K; ID_D é a identidade do destino; TSK_S é a chave privada temporária correspondente a TPK_S); PDO_i (*Protected Path Discovery Onion*) é uma estrutura *onion*, que registra o caminho anônimo. No caso da origem, o *onion* é igual a $[E_{TPK_S}(K_{SD}), E_{K_{SD}}(N_S, ID_S, PK_S, N_{K'}, K', PL, P, Sign_{SK_S}(M))]$, onde K_S é a chave de sessão a ser compartilhada com o destino; N_S é o seu pseudônimo de rota local; $N_{K'}$ e K' servem para substituir o índice e a chave antigos; PL é o tamanho do *padding* P; $Sign_{SK_S}(M)$ é o *hash* do pacote, assinado pela origem; M é igual a $H(TPK_S, TSK_S, K_{SD}, ID_S, ID_D, PK_S, N_K, K, N_{K'}, K', PL, P)$.

Um nó intermediário, ao receber ANON-RREQ pela primeira vez, adiciona uma

nova camada ao *onion*: cria uma chave de sessão K_i para ser compartilhada com origem e destino e a cifra com TPK_S, cria um pseudônimo de rota N_i e o cifra com K_i, junto com o *onion* recebido ($PDO_i = [E_{TPK_S}(K_i),\ E_{K_i}(N_i, PDO_{i-1})]$). Após inserir o novo *onion*, re-encaminha o pacote por difusão e tenta abrir tr_{dest}, para verificar se é o destino.

O nó destino, ao receber ANON-RREQ, decifra o *onion* com a chave TSK_S, obtida através do *trapdoor* e verifica se os dados estão corretos. Com os pseudônimos de rota (N_i) e correspondentes chaves de sessão (K_i), gera PRO_D (*Path Reverse Onion*) - $[E_{K_i}(N_{i-1},\ E_{K_{i-1}}\ldots(E_{K_1}(N_S,\ E_{K_{SD}}(N_1,\ K_1,\ \ldots,\ N_{i-1},\ K_{i-1},\ N_i,\ K_i,\ PL,\ P,\ Sign_{SK_D}(M)))))]$, onde $Sign_{SK_D}(M)$ é o *hash* do pacote, assinado pelo destino; M é igual a $H(N_S,\ K_{SD},\ N_{i-1},\ K_{i-1},\ N_i,\ K_i,\ PL,\ P)$. Adiciona a rota $[N_S,\ K_{SD},\ N_i,\ K_i,\ N_{i-1},\ K_{i-1},\ \ldots,\ N_1,\ K_1]$ em sua tabela de roteamento e envia, por difusão, um pacote *ANON-RREP*, cujo formato é

$$[ANON\text{-}RREP, N_i, PRO_i].$$

Quando um nó recebe *ANON-RREP*, verifica se o pseudônimo de rota N_i lhe pertence. Se pertence, remove uma camada de PRO_i usando sua chave de sessão K_i, substitui N_i por N_{i-1} (pseudônimo de rota do nó do qual recebeu o pacote de requisição), adiciona a rota $[N_i,\ N_{i-1},\ K_i]$ à sua tabela de rotas e envia o pacote por difusão. Quando o pacote chega à origem, esta remove a última camada, obtém os pseudônimos de rota e as chaves de sessão e os armazena em sua tabela de rotas.

Na transferência de dados, os nós que querem se comunicar usam as chaves de sessão compartilhadas entre si e com os nós intermediários para cifrar a comunicação com o método *onion*. Cada nó, ao longo da rota, remove uma camada e encaminha o pacote de dados, cujo formato é

$$[ANON\text{-}DATA, N_i, ADO_i],$$

onde N_i é o pseudônimo de rota do próximo nó; ADO_i (*Anonymous Communication Data Onion*) é o *onion* dos dados - $[E_{K_i}(E_{K_{i+1}}\ldots(E_{K_{SD}}(data)))]$. Um nó, ao recebê-lo, verifica se o pseudônimo de rota lhe pertence. Se pertence, substitui o pseudônimo pelo do próximo salto, remove uma camada de ADO_i e encaminha o pacote. Quando chega ao destino, este remove a última camada e obtém os dados. O destino procede da mesma forma para se comunicar com a origem.

O protocolo é analisado em relação a questões de segurança, anonimato e escalabilidade, e comparado aos protocolos ANODR e SDAR. São apresentados resultados de desempenho, obtidos através de simulação. Segundo os autores, AnonDSR é seguro contra ataques de modificação e de repetição, possui boa escalabilidade e provê anonimato da rota e das identidades dos nós origem e destino. Porém, durante o estabelecimento de parâmetros de segurança, sabe-se a identidade dos nós origem e destino, mas os adversários não podem descobrir a chave e o índice estabelecidos; o monitoramento do pacote ANON-RREQ por um adversário global pode revelar origem e destino, uma vez que contém a mesma chave pública, TPK_S, durante a fase de descoberta, mas não sabe os pseudônimos de rota e chaves de sessão criados pelos intermediários. Ataques de análise de tráfego são evitados com o envio de pacotes de resposta e de dados falsos. ANODR possui melhor desempenho; nós origem e destino não sabem toda a rota, assim não podem criar um *onion* para proteção dos dados; o *trapdoor* usado não é prático, porque o destino não sabe diretamente qual chave de sessão compartilhada usar. SDAR possui pobre escalabilidade, pois exige um alto custo computacional, mas provê anonimato forte do processo de descoberta e de transferência de dados.

3.3.9 *On-Demand Anonymous Routing* - ODAR

SY et al. (2006) propõe o protocolo ODAR - *On-Demand Anonymous Routing*, cujo objetivo é prover anonimato da identidade, da localização/topologia e da rota/caminho. A geração de chaves secretas entre origens e destinos é realizada através de *Diffie-Hellman* e com o auxílio de um servidor de chaves. É constituído por quatro fases: requisição do valor público do destino, requisição e resposta de rota e transferência de dados.

Quando um nó precisa se comunicar com outro, envia uma requisição ao servidor de chaves solicitando o valor público do destino (Y_D). Com esse valor e o seu valor privado X_S, gera uma chave secreta, $K_{SD} = Y_D^{X_S}$ (Y_i é igual a $g^{X_i} \, mod \, q$, onde g e q são os parâmetros públicos *Diffie-Hellman*; X_i é o valor privado de i, menor do que q). O servidor envia, constantemente, um pacote que contém sua chave pública e um campo, no qual os nós inserem suas identidades, antes de encaminhá-lo.

Após calcular K_{SD}, a origem encaminha, por difusão, um pacote de requisição,

cujo formato é

$$[RREQ, Y_S, Dst, bSize, bRoute],$$

onde Y_S é o valor público temporário da origem, que funciona como pseudônimo; *Dst* é a identidade do destino, protegida por *hash* seguro - $H_{K_{SD}}(ID_D+1)$; *bSize* indica o tamanho de *bRoute*, um *Bloom Filter*, no qual os nós, ao longo da rota, armazenam suas identidades de forma segura. A origem também insere seu pseudônimo no filtro - $H_{K_S}(ID_S + 1)$. *Bloom Filter* é um vetor de tamanho m e com k funções *hash*; o resultado da aplicação de cada uma dessas funções ao elemento a ser inserido indica a posição no vetor que deve ser modificada para 1.

Quando um nó intermediário recebe um RREQ, verifica se já o recebeu, através de Y_S e *Dst*, e tenta decifrar *Dst* para verificar se é o destino. Como não é, armazena Y_S, *Dst* e *bRoute*, insere um pseudônimo temporário em *bRoute* - $H_{K_i}(ID_i)$, incrementa *bSize* e encaminha RREQ.

O nó destino, ao receber RREQ, calcula $K_{SD} = Y_S^{X_D}$ e verifica *Dst* para confirmar que é o destino pretendido. Depois, encaminha, por difusão, um pacote de resposta RREP, cujo formato é

$$[RREP, bSize, bDest, Y_S, Dst],$$

onde *bSize* indica o tamanho de *bDest*, que contém as identidades dos nós na rota, inclusive a do destino - $H_{K_D}(ID_D + 2)$; Y_S é copiado de RREQ; *Dst* é a identidade do destino, protegida por *hash* seguro - $H_{K_{SD}}(ID_D + 2)$.

Quando um nó recebe RREP, verifica *bDest* para saber se faz parte da rota. Se faz, re-encaminha RREP. O nó origem, ao recebê-lo, pode encaminhar os pacotes de dados, cujo formato é

$$[DATA, bSize, bDest, data],$$

onde *data* são os dados a serem transmitidos.

Quando um nó recebe um pacote de dados, verifica *bDest* para saber se faz parte da rota ou se é o destino. Se faz, encaminha DATA. O processo é repetido, até chegar ao destino.

ODAR é analisado intuitivamente e através de simulação, e comparado ao AODV. De acordo com os pesquisadores, provê anonimato da identidade, da localização e da rota, além de ser seguro contra ataques ativos, como *man-in-the-middle* e *spoofing*.

Como limitação, citam ataques de negação de serviço e falsos positivos no *Bloom Filter*. Os resultados da simulação indicam que, apesar do *overhead*, o desempenho é comparável ao AODV.

3.4 Considerações

Este capítulo apresentou uma visão geral de anonimato, destacando-se as definições e os protocolos de roteamento anônimo para redes sem fio ad hoc móveis. Há outras propostas de protocolos, baseadas em posição. Não são descritas, porque preferiu-se apresentar apenas os protocolos com características e classificação semelhantes. Os aspectos de anonimato em redes sem fio ad hoc móveis abordados restringem-se à camada de rede, mais especificamente, a roteamento e encaminhamento de dados. Ou seja, não se abordaram temas como técnicas de localização (ex.: triangulação de sinal) e de proteção contra interceptação do sinal (ex.: modulação de espectro de dispersão). O próximo capítulo discute e compara os protocolos de roteamento em relação a anonimato e a outros aspectos.

4 *Comparativo entre Protocolos de Roteamento Anônimo para Redes Sem Fio Ad Hoc Móveis*

Neste capítulo, realiza-se uma comparação entre os protocolos estudados, quanto à segurança contra ataques de análise de tráfego, ao anonimato da identidade, do *venue* e da rota, à privacidade da localização e do padrão de movimento, aos mecanismos utilizados para prover anonimato, às técnicas de validação e algoritmos criptográficos, ao número de operações criptográficas e a características como mecanismo de atualização, topologia, suporte a enlaces assimétricos e a múltiplas rotas, e capacidade de multidifusão.

4.1 Definição de Anonimato Adotada, Aspectos Analisados e Suposições

Após estudar diferentes definições de propriedades de anonimato, adotou-se a definição apresentada por PFITZMANN & HANSEN (2006), especificamente, anonimato em termos de não-correlação entre itens de interesse (por exemplo, partes comunicantes, transmissão de um pacote, localização e padrão de movimento dos nós), e as propriedades citadas por KONG (2004), ZHU et al. (2004), KONG et al. (2005) e HONG et al. (2006) (seção 3.1):

- Anonimato da identidade do nó origem (destino): um atacante não pode relacionar um pacote à identidade do nó origem (destino).
- Anonimato da identidade dos nós intermediários: um atacante não pode rela-

cionar um pacote às identidades dos nós emissor e destinatário.

- Anonimato do *venue* do nó origem (destino): um atacante não pode correlacionar o nó origem (destino) e seu *venue*.

- Privacidade dq localização ou privacidade do *venue*: um atacante não pode relacionar a identidade de um nó à sua localização exata ou aproximada - *venue* (1), nem correlacionar pacotes transmitidos por um nó em sua localização atual (2). Se um protocolo garante (1), mas não (2), há privacidade fraca. Se ambas as propriedades são válidas, há privacidade forte.

- Privacidade do padrão de movimento: um atacante não pode relacionar a identidade de um nó aos pacotes que encaminha ou recebe (1), nem correlacionar pacotes transmitidos por um nó ou grupo de nós em suas localizações atuais e anteriores (2). Se um protocolo garante (1), mas não (2), há privacidade fraca. Se ambas as propriedades são válidas, há privacidade forte.

- Anonimato da rota: um atacante não pode correlacionar os nós pertencentes à rota (origem, intermediários e destino) e as transmissões, ao longo da rota, referentes ao mesmo pacote.

Anonimato do *venue* e privacidade da localização são conceitos próximos. O primeiro termo refere-se à não-identificação do *venue* do nó origem (ou destino), dado um conjunto de *venues* identificados após várias transmissões. O segundo refere-se à não-correlação entre a identidade de um nó e sua localização (pode ser exata ou aproximada) após uma transmissão.

Neste trabalho, assume-se que os enlaces são simétricos e omni-direcionais. Os adversários são passivos, podem ser externos ou internos; internos podem pertencer ou não à rota; possuem capacidade de comunicação entre si ilimitada e computacional restrita (não são capazes de quebrar os protocolos criptográficos nem de distinguir pacotes falsos de verdadeiros). Nos protocolos SDAR, ANDSR e ASRP, há pacotes que possuem identidades de nós (junto com outras informações), cifradas com chaves públicas conhecidas. Supõe-se que em SDAR, o tamanho da identidade é inferior ao tamanho do bloco usado para criptografia; em ANDSR, o valor aleatório que é cifrado junto com a identidade é inserido no início ou o tamanho da identidade é inferior ao tamanho do bloco; em ASRP, o tamanho do bloco é maior

do que o dobro do tamanho de uma identidade. Se tais suposições não são feitas, é possível descobrir as identidades comparando todas as combinações de chaves e identidades ao valor capturado; são feitas, porque são questões de implementação e não de projeto.

4.2 Comparação em Relação a Ataques de Análise de Tráfego

Ao analisar as descrições dos protocolos, é possível indicar suas possíveis vulnerabilidades a ataques de análise de tráfego. Consideram-se as seguintes perspectivas: o encaminhamento de um mesmo pacote ou de pacotes distintos do mesmo fluxo; um grupo de nós ou um único nó, que pode ser externo ou interno (pertencente à rota ou não) à rede. O Quadro 4.1 compara os protocolos quanto à proteção contra os ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo (seção 2.6). Considera-se que o protocolo é seguro contra o ataque, quando os pacotes não são vulneráveis sob o ponto de vista de qualquer nó; é vulnerável, quando todos os pacotes são propensos ao ataque sob o ponto de vista de qualquer nó; é parcialmente seguro, quando nem todos são vulneráveis ou se são sob o ponto de vista de apenas um tipo de nó ou sob determinadas condições.

Mecanismos como difusão e técnicas *mixing* podem tornar o ataque mais difícil. Por exemplo, quando um pacote é vulnerável a ataques de análise de tráfego, como conteúdo e volume, mas é difundido, é mais difícil seguir e ordenar seqüencialmente as suas transmissões, exceto se há um campo, como lista de nós, ou do ponto de vista de nós, em conluio, pertencentes à rota; quando um pacote é vulnerável ao ataque de volume, adversários fora da rota nem sempre podem segui-lo, porque pode haver outros (falsos ou verdadeiros), na mesma área, com tamanho idêntico.

O protocolo ANODR é parcialmente seguro contra os ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo:

- Ataque de análise de tempo: a fim de prevenir esse ataque, utiliza técnicas *mixing*. Porém, somente os pacotes de resposta e de dados são protegidos.
- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há dois campos, seq e tr_{dest}, que permanecem iguais durante sua transmissão, porém esse

Quadro 4.1: Segurança contra Ataques de Análise de Tráfego

	Análise de tempo	**Conteúdo do pacote**	**Volume do pacote**	**Reconhecimento de fluxo**
ANODR	P	P	P	P
SDAR	X	P	P	X
ASR	P	P	P	P
MASK	P	P	P	P
ANDSR	X	X	X	X
CARP	X	X	X	X
ASRP	X	P	P	X
AnonDSR	P	P	P	X
ODAR	P	X	P	X

NOTA: X - vulnerável ao ataque, P - parcialmente seguro contra o ataque

pacote é difundido por toda a rede. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados. Entretanto, são vulneráveis em relação a nós, em conluio, na rota, porque o primeiro possui uma informação, pr_{dest}, que apesar de estar cifrada, pode ser vista por eles, e a carga útil do segundo é cifrada salto a salto.

- Ataque de volume do pacote: a fim de prevenir esse ataque, cada salto preenche o *onion* da requisição e da resposta, até completar 400 bits. Assim, possuem tamanho único e fixo, apesar de o tamanho real do *onion* mudar, de modo padrão, a cada salto. Na requisição, cada nó acrescenta uma informação com tamanho fixo (um *nonce* - N_i) ao *onion* recebido e o resultado é cifrado com uma chave simétrica (K_i). No caso do nó origem, o *onion* é a sua identidade, cifrada com uma chave simétrica. Na resposta, decifra-se o *onion* e remove-se o *nonce*. Esse método protege somente contra ataques externos. Nós internos precisam saber o tamanho real do *onion* recebido, para que possam modificá-lo. Porém, com exceção dos nós pertencentes à rota, os demais internos não têm acesso ao *onion* da resposta, cifrado enlace a enlace. O pacote de dados é vulnerável, porque seu tamanho depende da carga útil,

porém é possível que haja outros pacotes, falsos ou verdadeiros, com o mesmo tamanho (nós pertencentes à rota, em conluio, podem confirmar, através do ataque de conteúdo, que é o mesmo pacote).

- Ataque de reconhecimento de fluxo: pacotes pertencentes à mesma rota não podem ser correlacionados, porque não há informações em comum, em texto claro, e não são vulneráveis à análise de tempo. Inclusive, os pseudônimos de cada enlace mudam dinamicamente a cada pacote encaminhado. Porém, um nó pertencente à rota pode correlacioná-los. Na fase de requisição, cada nó gera um par de chaves assimétricas temporárias, cuja chave pública (TPK_i) é inserida no pacote RREQ. Essa chave é utilizada no pacote RREP para cifrar a semente criptográfica ($E_{TPK_i}(K_{i+1})$), que gera a seqüência de pseudônimos utilizados na fase de transferência de dados.

O protocolo SDAR é vulnerável aos ataques de análise de tempo e de reconhecimento de fluxo, e parcialmente seguro contra os ataques de conteúdo e de volume:

- Ataque de análise de tempo: não cita mecanismos de proteção contra esse ataque. No caso de baixa carga de tráfego, a possibilidade de correlação entre os pacotes é maior.

- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há dois campos, *TRUST-REQ* e TPK_S, que permanecem iguais durante sua transmissão, porém esse pacote é difundido por toda a rede. Os demais campos são cifrados salto a salto com a chave da comunidade de cada nó. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados (uma camada do *onion* é decifrada).

- Ataque de volume do pacote: a requisição é vulnerável, porque seu tamanho depende do *padding* escolhido pela origem e aumenta, de modo padrão, durante sua transmissão. Cada nó acrescenta informações com tamanho fixo, a identidade do nó (ID_i), uma chave de sessão (K_i), um identificador de sessão ($SN_{Session\_ID_i}$) e a assinatura digital do *hash* do pacote ($Sign_{SK_i}(M_i)$). A resposta é vulnerável, porque seu tamanho depende do *padding* escolhido pelo destino e cada nó na rota remove uma camada da estrutura *onion* e outras informações com tamanho fixo. O pacote de dados é vulnerável, porque seu

tamanho depende da quantidade de dados e do número de nós na rota, e, a cada salto, é decifrada uma camada da estrutura *onion*. Porém, é possível que haja outros pacotes de resposta ou de dados com o mesmo tamanho (nós pertencentes à rota, em conluio, podem confirmar, através do ataque de conteúdo, que é o mesmo pacote).

- Reconhecimento de fluxo: pacotes pertencentes à mesma rota podem ser correlacionados através do identificador de sessão local - $SN_{Session\_ID_i}$; no caso de um nó pertencente à rota, através da chave de sessão K_i também. Em situações de pouco tráfego, podem ser correlacionados através de análise de tempo.

O protocolo ASR é parcialmente seguro contra os ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo:

- Ataque de análise de tempo: a fim de prevenir esse ataque, utiliza técnicas *mixing*. Porém, somente os pacotes de dados são protegidos.

- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há dois campos, *seq* e $E_{K_{SD}}(ID_D, K_S, U_0)$, $E_{K_S}(seq, END)$, que permanecem iguais durante sua transmissão, porém esse pacote é difundido por toda a rede. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados. Entretanto, são vulneráveis em relação a nós, em conluio, na rota, porque o primeiro possui duas informações, *seq* e K'_s, que apesar de estarem cifradas, podes ser vistas por eles, e a carga útil do segundo é cifrada salto a salto.

- Ataque de volume do pacote: a requisição é vulnerável, porque seu tamanho depende do tamanho de U_0 (p_s), que é calculado a partir de H_{max} (valor escolhido pela origem) e de p_x (valor fixo conhecido por todos os nós). A resposta não é vulnerável, porque seu tamanho é único e fixo, ou seja, todo pacote RREP possui, independentemente de rota, o mesmo tamanho, o qual não se altera durante sua transmissão. O pacote de dados é vulnerável, porque seu tamanho depende da carga útil, porém é possível que haja outros pacotes, falsos ou verdadeiros, com o mesmo tamanho (nós pertencentes à rota, em conluio, podem confirmar, através do ataque de conteúdo, que é o mesmo pacote).

- Ataque de reconhecimento de fluxo: pacotes pertencentes à mesma rota não podem ser correlacionados, porque não há informações em comum, em texto claro, e não são vulneráveis a análise de tempo. Porém, um nó pertencente à rota pode correlacioná-los. Na fase de requisição, cada nó gera uma chave pública (TPK_i), que é utilizada na fase de resposta, pelo próximo nó, para cifrar a chave a ser compartilhada pelos dois ($E_{TPK_i}(K_{i+1})$). Essas chaves secretas são utilizadas nos pacotes de dados, para cifrar uma informação através da qual, os nós verificam se são os destinatários (N, $H_{K_{i+1}}(N)$). São utilizadas também para cifrar salto a salto a carga útil.

O protocolo MASK é parcialmente seguro contra os ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo:

- Ataque de análise de tempo: a fim de prevenir esse ataque, utiliza técnicas *mixing*, quando a carga de tráfego é baixa. Porém, somente os pacotes de dados são protegidos.

- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há três campos, *seq*, ID_D e *destSeq*, que permanecem iguais durante sua transmissão, porém esse pacote é difundido por toda a rede. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados. Entretanto, são vulneráveis em relação a nós, em conluio, na rota, porque o primeiro possui duas informações, ID_D e *destSeq*, que apesar de estarem cifradas, podem ser vistas por eles, e a carga útil do segundo é cifrada salto a salto.

- Ataque de volume do pacote: requisição e resposta não são vulneráveis, porque possuem tamanho único e fixo. O pacote de dados não é vulnerável em relação a nós fora da rota, porque os intermediários acrescentam *paddings* aleatórios à carga útil (nós pertencentes à rota, em conluio, podem confirmar, através do ataque de conteúdo, que é o mesmo pacote).

- Reconhecimento de fluxo: pacotes pertencentes à mesma rota não podem ser correlacionados, porque não há informações em comum, em texto claro, e não são vulneráveis a análise de tempo. Inclusive, os identificadores de cada enlace mudam dinamicamente a cada pacote encaminhado e um nó pode participar de múltiplas rotas para o destino. Assim, um nó pertencente à rota pode correlacionar os pacotes que encaminha por destino, mas não por rota (se bem que

a possibilidade de um conjunto de pacotes, enviados após uma requisição, ser da mesma rota é alta), porque as tabelas de encaminhamento são organizadas por destino e não por rota.

O protocolo ANDSR é vulnerável aos ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo:

- Ataque de análise de tempo: não cita mecanismos de proteção contra esse ataque, nem fica claro se, como e quando os misturadores utilizam técnicas *mixing*.
- Ataque de conteúdo do pacote: requisição, resposta e dados são vulneráveis, porque possuem campos que permanecem inalterados, durante sua transmissão. No caso do primeiro, os campos $E_{PK_M}(ident, R)$, $E_{PK_M}(A_D, R)$ (este é substituído por $E_{PK_D}(A_D, R)$, somente quando chega ao misturador vizinho ao destino), $E_{PK_D}(A_S, R)$ e $path(E_{PK_S}(A_S, R), \ldots)$ (a cada salto, são acrescentados nós à lista), no do segundo, $E_{PK_M}(ident, R)$, $E_{PK_S}(A_D, R)$, $E_{PK_S}(A_S, R)$ e $path(E_{PK_S}(A_D, R)$, A_X, $\ldots$, $E_{PK_S}(A_S, R)$, e no do último, $E_{PK_D}(A_S, R)$, $E_{PK_M}(A_D, R)$ (este é substituído por $E_{PK_D}(A_D, R)$, somente quando chega ao misturador vizinho ao destino) e $E_{PK_D}(data, R)$.
- Ataque de volume do pacote: a requisição é vulnerável, porque, a cada salto, é acrescentada uma informação com tamanho fixo, o endereço do nó misturador intermediário (A_M). Resposta (e dados) é vulnerável, porque seu tamanho depende do número de nós na rota (e da quantidade de dados também, no caso do pacote de dados).
- Reconhecimento de fluxo: pacotes pertencentes à mesma rota pode ser correlacionados através dos campos $E_{PK_D}(A_S, R)$ e $E_{PK_M}(A_D, R)$. Aliás, através desses campos, pode-se correlacionar todas as transmissões pertencentes a uma rota.

O protocolo CARP é vulnerável aos ataques de análise de tempo, de conteúdo, de volume e de reconhecimento de fluxo:

- Ataque de análise de tempo: não cita mecanismos de proteção contra esse ataque. Assim, quando há pouco tráfego, a possibilidade de correlação entre os pacotes é maior.

- Ataque de conteúdo do pacote: requisição, resposta e dados são vulneráveis, porque possuem campos que permanecem inalterados durante sua transmissão. No caso da requisição, os campos *r.P*, *route-request-token*, $E_K(m_S)$ e *node-pseudonym-list* (a cada salto, são acrescentados nós à lista), da resposta, *route-reply-token*, *uid*, $E_{nonces}(m_D)$ e *node-pseudonym-list*, e dos dados, *data-token*, *uid*, $E_{K_{session}}(data)$ e *node-pseudonym-list*.

- Ataque de volume do pacote: a requisição é vulnerável, porque, a cada salto, é acrescentada uma informação com tamanho fixo, o pseudônimo do nó. A resposta é vulnerável, porque seu tamanho depende do número de nós na rota. O pacote de dados é vulnerável, porque seu tamanho depende da carga útil e do número de nós na rota.

- Ataque de reconhecimento de fluxo: pacotes pertencentes à mesma rota pode ser correlacionados através dos campos *uid* e *node-pseudonym-list*. Aliás, através desses campos, pode-se correlacionar todas as transmissões pertencentes a uma rota.

O protocolo ASRP é vulnerável aos ataques de análise de tempo e de reconhecimento de fluxo, e parcialmente seguro contra os ataques de conteúdo e de volume:

- Ataque de análise de tempo: não cita mecanismos de proteção contra esse ataque. Assim, quando há pouco tráfego, a possibilidade de correlação entre os pacotes é maior.

- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há quatro campos, *seq*, $E_{PK_D}(ID_S, ID_D, N_S, K_{SD})$, TPK_S e $E_{TPK_S}(N_S)$, que permanecem iguais durante sua transmissão, porém esse pacote é difundido por toda a rede. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados. Entretanto, são vulneráveis em relação a nós, em conluio, na rota, porque o primeiro possui uma informação, N_S, que apesar de estar cifrada, pode ser vista por eles, e a carga útil do segundo é cifrada salto a salto.

- Ataque de volume do pacote: requisição e resposta não são vulneráveis, porque possuem tamanho único e fixo. O pacote de dados é vulnerável, porque seu tamanho depende da carga útil, porém é possível que haja outros pacotes com

o mesmo tamanho (nós pertencentes à rota, em conluio, podem confirmar, através do ataque de conteúdo, que é o mesmo pacote).

- Reconhecimento de fluxo: pacotes de dados podem ser correlacionados através do pseudônimo de cada nó - PID'_i; no caso de um nó pertencente à rota, através da chave de sessão K_i também. Em situações de pouco tráfego, pacotes pertencentes à mesma rota podem ser correlacionados através de análise de tempo.

O protocolo AnonDSR é vulnerável ao ataque de reconhecimento de fluxo e, é parcialmente seguro contra os ataques de análise de tempo, de conteúdo e de volume:

- Ataque de análise de tempo: a fim de prevenir esse ataque, são enviados pacotes de resposta e de dados falsos.
- Ataque de conteúdo do pacote: a requisição é vulnerável, porque há dois campos, TPK_S e tr_{dest}, que permanecem iguais durante sua transmissão, porém esse pacote é difundido por toda a rede. Resposta e dados não são vulneráveis, porque, a cada salto, todos os campos são modificados (uma camada do *onion* é decifrada).
- Ataque de volume do pacote: a requisição é vulnerável, porque seu tamanho depende do *padding* (P) escolhido pela origem e aumenta, de modo padrão, a cada salto. O *onion* (PDO_i) é substituído por duas informações, uma com tamanho fixo (a chave de sessão gerada pelo nó, cifrada com a chave pública temporária do nó origem, $E_{TPK_S}(K_i)$) e outra com tamanho previsível (um pseudônimo de rota com tamanho fixo e o *onion* recebido, cifrados com a chave de sessão - $E_{K_i}(N_i, PDO_{i-1})$). A resposta é vulnerável, porque seu tamanho depende do *padding* (P) escolhido pelo destino e da quantidade de nós na rota, e diminui, de modo padrão, a cada salto, pois é decifrada uma camada do *onion* (PRO_i) e removida uma informação com tamanho fixo (um pseudônimo de rota - N). O pacote de dados é vulnerável, porque seu tamanho depende da carga útil e diminui, de modo padrão, a cada salto, pois é decifrada uma camada do *onion* (ADO_i), que contém os dados. Porém, é possível que haja outros pacotes de resposta ou de dados, falsos ou verdadeiros, com o mesmo tamanho.

- Reconhecimento de fluxo: pacotes pertencentes à mesma rota podem ser correlacionados através do pseudônimo de rota local - N_i; no caso de um nó pertencente à rota, através da chave de sessão K_i também.

O protocolo ODAR é parcialmente seguro contra os ataques de análise de tempo e de volume, e vulnerável aos ataques de conteúdo e de reconhecimento de fluxo:

- Ataque de análise de tempo: a fim de prevenir esse ataque, cita a utilização de técnicas *mixing*. Porém, somente os pacotes de dados são protegidos.
- Ataque de conteúdo do pacote: requisição, resposta e dados são vulneráveis, porque possuem campos que permanecem inalterados durante sua transmissão. No caso da requisição, os campos Y_S e *Dst* (porém esse pacote é difundido por toda a rede), da resposta, *bSize*, *bDest*, Y_S e *Dst*, e dos dados, *bSize*, *bDest* e carga útil. Qualquer nó pode identificar os pacotes de um mesmo fluxo. Requisição e resposta possuem, em comum, o campo Y_S, resposta e dados, *bSize* e *bDest*. Além disso, esses dois campos são idênticos em todos os pacotes de dados.
- Ataque de volume do pacote: requisição e resposta não são vulneráveis, porque possuem tamanho único e fixo. O pacote de dados é vulnerável, porque seu tamanho depende da carga útil (apesar da possibilidade de haver outros pacotes, falsos ou verdadeiros, com o mesmo tamanho, pode-se confirmar, através do ataque de conteúdo, que é o mesmo pacote).
- Ataque de reconhecimento de fluxo: pacotes pertencentes à mesma rota pode ser correlacionados através dos campos *bSize* e *bDest*. Aliás, através desses campos, pode-se correlacionar todas as transmissões pertencentes a uma rota.

4.3 Análise em Relação a Anonimato

Esta seção analisa e compara os protocolos em relação a anonimato da identidade, do *venue*, da localização, do padrão de movimento e da rota. O comparativo apresentado baseia-se na descrição dos protocolos e na análise em relação aos ataques de análise de tráfego. Considera-se que o anonimato é parcial quando o protocolo não garante, em determinadas situações ou quando se consideram pacotes que não

pertencem aos processos de roteamento e de encaminhamento, todos os itens da propriedade de anonimato analisada.

4.3.1 Anonimato da Identidade

O Quadro 4.2 compara os protocolos em relação a anonimato das identidades dos nós origem, destino e intermediários. Para a verificação dessas propriedades, as seguintes questões são analisadas: se os pacotes possuem as identidades dos nós origem, destino, emissor e destinatário ou quaisquer informações que possam ser associadas a eles; no caso de tais informações serem utilizadas, se elas estão em texto claro ou se são criptografadas, mas a chave correspondente é revelada para algum nó; se as identidades dos nós emissor e destinatário são conhecidas e o número de saltos pode ser calculado (o atacante pode descobrir a identidade da origem ou do destino pelo número de saltos). Os protocolos ANODR e ANDSR usam *tags* públicas que identificam os nós origem e destino, e endereços dos nós, respectivamente; do ponto de vista deste trabalho, tais *tags* e endereços funcionam como identidades, uma vez que são identificadores únicos.

Quadro 4.2: Anonimato da Identidade

	Anonimato do nó origem	**Anonimato do nó destino**	**Anonimato dos nós intermediários**
ANODR	√	X	√
SDAR	√	√	X
ASR	√	√	√
MASK	√	X	√
ANDSR	P	X	X
CARP	√	√	√
ASRP	√	√	√
AnonDSR	P	P	P
ODAR	√	√	P

NOTA: √ - provê, X - não provê, P - provê parcialmente

O protocolo ANODR provê anonimato das identidades dos nós origem e intermediários e não provê do nó destino:

- Anonimato da identidade do nó origem: está presente no *onion* (TBO_i) da requisição e da resposta, mas cifrada com uma chave conhecida apenas por ele. Além disso, não há informações que o identifiquem.
- Anonimato da identidade do nó destino: está presente no *trapdoor* da requisição ($E_{K_{SD}}(dest, K_c), E_{K_c}(dest)$) e pode ser visualizado pelos nós pertencentes à rota, após receberem a resposta. Esse pacote contém a chave K_c.
- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Nas fases de requisição e de resposta, utilizam chaves públicas temporárias e estabelecem pseudônimos de rota dinâmicos (cada nó, na rota, estabelece uma seqüência de pseudônimos com o próximo), que são utilizados na fase de transferência de dados.

O protocolo SDAR provê anonimato das identidades dos nós origem e destino, e não provê dos intermediários:

- Anonimato da identidade do nó origem: está presente na requisição, mas cifrada com uma chave conhecida apenas pelos nós origem e destino ($E_{K_{SD}}(ID_S$, PK_S, TPK_S, TSK_S, $SN_{Session\_ID_S})$, $Sign_{SK_S}(M_S)$).
- Anonimato da identidade do nó destino: apesar de estar presente na requisição, está cifrada com a sua chave pública ($E_{PK_D}(ID_D, K_{SD}, PL_S)$). Não há outras informações que o identifiquem.
- Anonimato das identidades dos nós intermediários: o destino sabe as identidades de todos os intermediários da rota, porque eles as cifram com uma chave pública temporária (par é gerado pela origem e compartilhado com destino) e as inserem no pacote de requisição ($E_{TPK_S}(ID_i,\ K_i,\ SN_{Session\_ID_i}$, $Sign_{SK_i}(M_i))$).

O protocolo ASR provê anonimato das identidades dos nós origem, destino e intermediários:

- Anonimato da identidade do nó origem: nenhum pacote possui a identidade do nó origem nem qualquer informação que o identifique.
- Anonimato da identidade do nó destino: apesar de estar presente no *trapdoor* da requisição, está cifrada com uma chave conhecida apenas pelos nós origem e destino ($E_{K_{SD}}(ID_D, K_S, U_0)$). Não há outras informações que o identifiquem.
- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Nas fases de requisição e de resposta, utilizam chaves públicas temporárias e estabelecem chaves secretas com os vizinhos, que são utilizadas na fase de transferência de dados.

O protocolo MASK provê anonimato das identidades dos nós origem e intermediários e não provê do destino:

- Anonimato da identidade do nó origem: nenhum pacote possui a identidade do nó origem nem qualquer informação que o identifique.
- Anonimato da identidade do nó destino: está presente na requisição, em texto claro.
- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Utilizam pares de chaves e de identificadores de enlace dinâmicos e únicos para cada vizinho. Os nós utilizam pseudônimos, que são trocados apenas quando se movem para outro local. No caso de redes estáticas, esse pseudônimo funcionaria como sua identidade. Mas, neste trabalho assume-se que as redes são móveis. Além disso, não é possível associar os pseudônimos de um nó entre si e à sua identidade real.

O protocolo ANDSR provê parcialmente anonimato da identidade do nó origem e não provê do destino e dos intermediários:

- Anonimato da identidade do nó origem: está presente em todos os pacotes, mas cifrada. Na requisição e nos dados, está cifrada com a chave pública do destino ($E_{PK_D}(A_S, R)$). Na requisição e na resposta, está cifrada com a sua

chave pública ($E_{PK_S}(A_S, R)$). Não há outras informações que o identifiquem. Porém, durante a fase de requisição de misturador, a sua identidade e a do primeiro misturador são reveladas. Assim, quando requisições são enviadas a esse misturador, pode-se descobrir os possíveis nós origem.

- Anonimato da identidade do nó destino: está presente em todos os pacotes, mas nós comuns não podem descobri-la, porque está cifrada. Porém, qualquer misturador pode. Nos pacotes de requisição e de dados, está cifrada com a chave pública dos misturadores ($E_{PK_M}(A_D, R)$) e, após passar pelo último misturador, com sua própria chave ($E_{PK_D}(A_D)$). Na resposta, está cifrada com a chave pública da origem ($E_{PK_S}(A_D)$).
- Anonimato das identidades dos nós intermediários: a identidade de todos os intermediários (misturadores) é inserida no campo *path* dos pacotes de requisição e de resposta.

O protocolo CARP provê anonimato das identidades dos nós origem, destino e intermediários:

- Anonimato das identidades dos nós origem e destino: estão presentes nos pacotes de requisição e de resposta, mas cifradas com uma chave conhecida somente por eles ($E_K(ID_S, ID_D, nonce_S,$ *init-vector*) e $E_{nonce_S}(ID_S, ID_D, nonce_D)$, respectivamente). Não há outras informações que os identifiquem.
- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Utilizam pseudônimos específicos para a rota.

O protocolo ASRP provê anonimato das identidades dos nós origem, destino e intermediários:

- Anonimato das identidades dos nós origem e destino: estão presentes na requisição, mas cifradas com a chave pública do destino ($E_{PK_D}(ID_S, ID_D, N_S, K_S)$). Não há outras informações que os identifiquem.
- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Utilizam chaves e pseudônimos específicos para a rota.

O protocolo AnonDSR provê parcialmente anonimato das identidades dos nós origem, destino e intermediários:

- Anonimato da identidade do nó origem: apesar de estar presente no *onion* (PDO_i) da requisição, está cifrada com uma chave conhecida apenas pelos nós origem e destino. Não há outras informações que o identifiquem. Porém, durante o estabelecimento dos parâmetros de segurança, sua identidade é revelada. Assim, no caso de baixa carga de tráfego, pode-se identificá-lo.

- Anonimato da identidade do nó destino: apesar de estar presente no *trapdoor* da requisição, está cifrada com uma chave conhecida apenas pelos nós origem e destino ($N_K, E_K(ID_D, TSK_S)$). Não há outras informações que o identifiquem. Porém, durante o estabelecimento dos parâmetros de segurança, sua identidade é revelada. Assim, no caso de baixa carga de tráfego, pode-se identificar o nó destino.

- Anonimato das identidades dos nós intermediários: nenhum pacote possui as identidades dos nós emissor e destinatário nem qualquer informação que possa ser associada a eles. Utilizam chaves e pseudônimos específicos para a rota. Porém, durante o estabelecimento dos parâmetros de segurança, as identidades dos nós entre origem e destino são reveladas. A possibilidade de os nós serem os mesmos quando a rota for estabelecida depende de fatores como, por exemplo, mobilidade.

O protocolo ODAR provê anonimato das identidades dos nós origem e destino e parcialmente dos intermediários:

- Anonimato da identidade do nó origem: um *hash* seguro da identidade do nó origem ($H_{K_S}(ID_S + 1)$) é inserido no campo *Bloom Filter* (*bRoute* ou *bDest*) de todos os pacotes. Assim, somente ele é capaz de saber se sua identidade está presente no *Bloom Filter*.

- Anonimato da identidade do nó destino: está presente no campo *Dst* da requisição ($H_{K_{SD}}(ID_D + 1)$) e da resposta ($H_{K_{SD}}(ID_D + 2)$) e no campo *bDest* da resposta e dos dados, mas protegida por *hash* seguro.

- Anonimato das identidades dos nós intermediários: um *hash* seguro da identidade de cada salto ($H_{K_i}(ID_i)$) é inserido no *Bloom Filter*. Porém, o servidor de chaves envia um pacote, que contém as identidades de todos os nós que o recebem.

4.3.2 Anonimato dos *Venues* dos Nós Origem e Destino

O Quadro 4.3 compara os protocolos em relação a anonimato dos *venues* dos nós origem e destino. Para a verificação dessas propriedades, as seguintes questões são analisadas: se os nós sabem a sua distância em relação aos nós origem e destino, ou seja, se os pacotes possuem contadores de saltos ou se o seu tamanho indica o número de saltos percorridos; se os pacotes podem ser seguidos, através de ataques de análise de tráfego, até o nó origem ou destino.

Quadro 4.3: Anonimato dos *Venues* dos Nós Origem e Destino

	Anonimato do nó origem	**Anonimato do nó destino**
ANODR	X	√
SDAR	P	P
ASR	X	P
MASK	P	√
ANDSR	X	X
CARP	X	X
ASRP	P	P
AnonDSR	P	√
ODAR	X	X

NOTA: √ - provê, X - não provê, P - provê parcialmente

O protocolo ANODR não provê anonimato do *venue* do nó origem e provê do nó destino:

- Anonimato do *venue* do nó origem: atacantes internos, ao receberem a requisição original, sabem que são vizinhos da origem, através do tamanho

da requisição, mais especificamente do tamanho do *onion*; considerando, por exemplo, os valores utilizados pelos pesquisadores na simulação - *nonce* de 40 bits, chave simétrica de 128 bits, *tag* de 128 bits e algoritmo simétrico AES, o tamanho do *onion* enviado pela origem ($E_{K_S}(src)$) é igual a 32 bytes, pelos próximos nós a 37 bytes ($E_{K_A}(N_A, E_{K_S}(src))$), a 53 bytes ($E_{K_B}(N_B, E_{K_A}(N_A, E_{K_S}(src))))$ e assim por diante. Nós externos não sabem o tamanho real, porque cada nó acrescenta um *padding* para que o pacote possua sempre o mesmo tamanho. Se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo à requisição, é possível descobrir o nó que a enviou pela primeira vez.

- Anonimato do *venue* do nó destino: apesar de o tamanho da resposta mudar a cada salto de modo padrão, os nós não sabem qual é a sua distância do destino (o tamanho depende da distância da origem). Não é possível descobrir o venue do destino, seguindo o pacote de requisição ou de dados, ou descobrindo a origem da resposta, porque o primeiro pacote é difundido e os demais não são vulneráveis aos ataques de volume (com exceção dos dados, mas há técnicas *mixing*), de análise de tempo e de conteúdo.

Os protocolos SDAR e ASRP provêem parcialmente anonimato dos *venues* dos nós origem e destino:

- Anonimato do *venue* do nó origem: em SDAR, apesar de o tamanho da requisição aumentar de forma padrão a cada salto, não é possível saber a distância do nó origem, porque este insere um *padding* de tamanho aleatório no pacote; em ASRP, o tamanho da requisição é fixo. Porém, se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo à requisição, é possível descobrir o nó que a enviou pela primeira vez. Além disso, em situações de pouco tráfego, é possível descobrir a origem dos pacotes de dados, porque são vulneráveis aos ataques de análise de tempo e de volume, e se há somente uma requisição, é possível seguir o pacote de resposta do destino até a origem.

- Anonimato do *venue* do nó destino: em SDAR, o destino acrescenta um *pad-*

ding de tamanho aleatório à resposta e apesar de o seu tamanho mudar a cada salto de modo padrão, os nós não sabem qual é a sua distância do destino; em ASRP, o tamanho da resposta é fixo. Se há um adversário com capacidade global ou atacantes monitorando toda a rede, e apenas uma requisição, pode-se identificar o *venue* do destino, quando este enviar a resposta. Em situações de pouco tráfego, é possível seguir os dados até o destino, porque não há técnicas *mixing* e esse pacote é vulnerável ao ataque de volume.

O protocolo ASR não provê anonimato do *venue* do nó origem e provê parcialmente do nó destino:

- Anonimato do *venue* do nó origem: atacantes, ao receberem a requisição original, sabem que são vizinhos da origem, através do campo U_i. A cada salto, o seu tamanho não se altera e ocorre um deslocamento à direita de p_x bits, assim, pela quantidade de grupos de p_x bits iguais a zero à esquerda, descobre-se o número de saltos percorridos. Considerando, por exemplo, $H_{max} = 3$ e $p_x = 3$, obtém-se $p_s = 12$; supondo, então, $U_0 = 111011011001$, $S_1 = 101$ e $S_2 = 111$, obtém-se $U_1 = 000111011011$, $U_2 = 000000111011$. Se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo à requisição, é possível descobrir o nó que a enviou pela primeira vez. Além disso, se há apenas uma requisição na rede, pode-se seguir a resposta até a origem.

- Anonimato do *venue* do nó destino: não é possível descobrir o *venue* do destino através do tamanho da resposta, porque seu tamanho é fixo, nem seguindo os dados, porque esse pacote não é vulnerável aos ataques de análise de tempo e de conteúdo (é vulnerável ao ataque de volume, mas há técnicas *mixing*). Porém, se há um adversário com capacidade global ou atacantes monitorando toda a rede, e apenas uma requisição, pode-se identificar o *venue* do destino, quando este enviar a resposta.

Os protocolos MASK e AnonDSR provêem parcialmente anonimato do *venue* do nó origem e provê do nó destino:

- Anonimato do *venue* do nó origem: em MASK, não é possível descobrir o *venue* da origem através do tamanho da requisição, porque seu tamanho é fixo; em

AnonDSR, apesar de o tamanho da requisição aumentar, de forma padrão, a cada salto, não é possível saber a distância do nó origem, porque este insere um *padding* de tamanho aleatório no pacote. Não é possível descobrir a origem dos dados, porque esse pacote não é vulnerável aos ataques de análise de tempo, de conteúdo e de volume (exceto em AnonDSR, mas há envio de pacotes falsos). Porém, se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo à requisição, é possível descobrir o nó que a enviou, pela primeira vez. Além disso, no caso de MASK, se há apenas uma requisição na rede, pode-se seguir a resposta até a origem.

- Anonimato do *venue* do nó destino: em MASK, não é possível descobrir o *venue* do destino através do tamanho da resposta, porque seu tamanho é fixo; em AnonDSR, o destino acrescenta um *padding* de tamanho aleatório à resposta e apesar de o seu tamanho mudar a cada salto de modo padrão, os nós não sabem qual é a sua distância do destino. Não é possível descobrir o venue do destino seguindo o pacote de requisição ou de dados, ou descobrindo a origem da resposta, porque o primeiro pacote é difundido a toda a rede e os outros não são vulneráveis aos ataques de análise de tempo (em MASK, a resposta é vulnerável, mas intermediários podem enviá-la também), de conteúdo e de volume (em AnonDSR, resposta e dados são vulneráveis ao ataque de volume, mas há envio de pacotes falsos).

Os protocolos ANDSR e CARP não provêem anonimato dos *venues* dos nós origem e destino:

- Anonimato do *venue* do nó origem: atacantes, ao receberem a requisição original, sabem que são vizinhos da origem, através, respectivamente, dos campos *path* e *node-pseudonym-list*, que contém os endereços e pseudônimos dos nós que encaminham esse pacote. O pacote de resposta (e o de dados) é vulnerável aos ataques de conteúdo e de volume e portanto, pode ser seguido até (de volta) à origem. Em ANDSR, como a resposta possui o endereço dos misturadores, os vizinhos do primeiro misturador sabem que são vizinhos da origem. Se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo

à requisição ou aos dados, é possível descobrir o nó que os enviou pela primeira vez.

- Anonimato do *venue* do nó destino: atacantes, ao receberem a resposta original, sabem que são vizinhos do destino, através, respectivamente, dos campos *path* e *node-pseudonym-list* da requisição e da resposta. Os pacotes de dados são vulneráveis aos ataques de conteúdo e de volume e portanto, podem ser seguidos até o destino. Em ANDSR, como a resposta possui o endereço dos misturadores, os vizinhos do último misturador sabem que são vizinhos do destino.

O protocolo ODAR não provê anonimato dos *venues* dos nós origem e destino:

- Anonimato do *venue* do nó origem: atacantes, ao receberem a requisição original, sabem que são vizinhos da origem, através do campo *bSize*, que indica o número de saltos percorridos. O pacote de resposta (e de dados) é vulnerável ao ataque de conteúdo e portanto, pode ser seguido até (de volta) à origem. Se há um adversário com capacidade global ou atacantes monitorando toda a rede, pode-se identificar o *venue* da origem, porque, através do ataque de conteúdo à requisição ou aos dados, é possível descobrir o nó que os enviou pela primeira vez.
- Anonimato do *venue* do nó destino: atacantes, ao receberem a resposta original, sabem que são vizinhos do destino, através do campo *bSize* da requisição e da resposta. Os pacotes de dados são vulneráveis aos ataques de conteúdo e de volume e portanto, podem ser seguidos até o destino.

4.3.3 Privacidade da Localização e do Padrão de Movimento

O Quadro 4.4 compara os protocolos em relação à privacidade da localização e do padrão de movimento. Para a verificação de ambas as propriedades, deve-se verificar se os pacotes possuem informações que possam identificar emissor ou destinatário; se pacotes transmitidos (em uma mesma localização, no caso da primeira ou em qualquer localização, no caso da segunda) pelo mesmo nó (ou um grupo de nós, no caso da segunda) podem ser correlacionados, ou seja, se possuem informações em comum.

Quadro 4.4: Privacidade da Localização e do Padrão de Movimento

	Localização	**Padrão de movimento**
ANODR	forte	forte
SDAR	parcialmente fraca	parcialmente fraca
ASR	forte	forte
MASK	fraca	forte
ANDSR	não há	não há
CARP	forte	fraca
ASRP	fraca	fraca
AnonDSR	parcialmente fraca	parcialmente fraca
ODAR	parcialmente fraca	parcialmente fraca

Os protocolos ANODR e ASR provêem privacidade forte da localização e do padrão de movimento, enquanto CARP, forte e fraca. Não é possível relacionar a identidade de um nó a qualquer pacote, porque há anonimato do emissor e do destinatário. Não é possível correlacionar quaisquer pacotes transmitidos por um nó ou grupo de nós (com exceção do protocolo CARP, em que pacotes pertencentes à mesma rota podem ser correlacionados), porque não são vulneráveis ao ataque de reconhecimento de fluxo.

Os protocolos SDAR, AnonDSR e ODAR provêem privacidade parcialmente fraca da localização e do padrão de movimento. Não é possível relacionar a identidade de um nó a qualquer pacote, porque há anonimato do emissor e do destinatário. Porém, em SDAR, todo nó conhece seus vizinhos, uma vez que mantém uma lista de vizinhos e níveis de confiança, e como discutido na seção 4.3.1, não há anonimato caso se considerem o processo de estabelecimento dos parâmetros de segurança (AnonDSR) e o pacote do servidor de chaves (ODAR). Todos os pacotes pertencentes a uma rota, encaminhados por um nó (SDAR e AnonDSR) ou grupo de nós (ODAR), podem ser correlacionados através do ataque de reconhecimento de fluxo.

O protocolo ANDSR não provê privacidade da localização e do padrão de movimento. É possível relacionar a identidade de um misturador a todos os pacotes que encaminha ou recebe, porque não há anonimato do emissor e do destinatário; nós comuns revelam suas identidades ao requisitarem misturadores. Os pacotes pertencentes a uma rota, encaminhados por todos os nós, podem ser correlacionados através do ataque de reconhecimento de fluxo.

Os protocolos MASK e ASRP provêem privacidade fraca da localização, e privacidade forte e fraca, respectivamente, do padrão de movimento. Não é possível relacionar a identidade de um nó a qualquer pacote, porque há anonimato do emissor e do destinatário. Em ASRP, os pacotes pertencentes a uma rota, encaminhados por um nó, podem ser correlacionados através do ataque de reconhecimento de fluxo. Já em MASK, somente requisições transmitidas por um nó em sua localização atual podem ser correlacionados, porque possuem seu pseudônimo ativo.

4.3.4 Anonimato da Rota

O Quadro 4.5 compara os protocolos em relação a anonimato da rota, sob o ponto de vista de nós fora e dentro da rota. Para a verificação dessa propriedade, as seguintes questões são analisadas: se informações capazes de identificar os nós são usadas e reveladas; se quaisquer (ou algumas) transmissões referentes ao mesmo pacote podem ser correlacionadas, ou seja, se os pacotes são vulneráveis a ataques de análise de tráfego.

Quadro 4.5: Anonimato da Rota

	Anonimato da rota (nós fora da rota)	**Anonimato da rota (nós na rota)**
ANODR	$\surd$	P
SDAR	P	P
ASR	P	P
MASK	P	P
ANDSR	X	X
CARP	X	X
ASRP	P	P
AnonDSR	P	P
ODAR	X	X

NOTA: $\surd$ - provê, X - não provê, P - provê parcialmente

O protocolo ANODR provê anonimato da rota em relação a nós fora da rota, e

parcialmente, em relação a pertencentes:

- Anonimato da rota em relação a nós fora da rota: não são capazes de associar os nós pertencentes a cada rota, porque há anonimato da identidade. Não correlacionam as transmissões referentes ao mesmo pacote, porque não podem realizar ataques de análise de tempo, de conteúdo e de volume.

- Anonimato da rota em relação a nós pertencentes à rota: são capazes de identificar somente o destino. Em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota, através dos ataques de conteúdo e de volume aos pacotes que cada um correlacionou pelo ataque de reconhecimento de fluxo. Pelo tamanho dos pacotes de requisição e de resposta, podem descobrir sua distância relativa.

O protocolo SDAR provê parcialmente anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: não são capazes de associar os nós pertencentes a cada rota, porque há anonimato da identidade. Como discutido na seção 4.3.2, transmissões referentes ao mesmo pacote podem ser correlacionadas em situações de pouco tráfego.

- Anonimato da rota em relação a nós pertencentes à rota: assim como os nós fora da rota, podem correlacionar as transmissões referentes ao mesmo pacote, em situações de pouco tráfego. O destino sabe a identidade de todos os intermediários. Pelo tamanho dos pacotes de requisição e de resposta, nós na rota sabem a distância relativa entre eles.

Os protocolos ASR e MASK provêem parcialmente anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: não podem correlacionar transmissões referentes ao mesmo pacote através dos ataques de análise de tempo, de conteúdo e de volume. Porém, como discutido na seção 4.3.2, o pacote de resposta pode ser seguido em certas situações. Em ASR, não podem correlacionar os nós pertencentes a cada rota, porque há anonimato da identidade; em MASK, qualquer nó pode identificar o destino.

- Anonimato da rota em relação a nós pertencentes à rota: em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota, através dos ataques de conteúdo e de volume aos pacotes que cada um correlacionou pelo ataque de reconhecimento de fluxo. Em ASR, pelo tamanho do pacote de requisição, podem descobrir sua distância relativa; em MASK, podem identificar o destino.

O protocolo ANDSR não provê anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: são capazes de associar as identidades de todos os intermediários na rota, porque não há anonimato das identidades dos nós emissor e destinatário, e correlacionar transmissões referentes ao mesmo pacote através dos ataques de análise de tempo, de conteúdo e de volume. Podem supor os possíveis nós origens, porque, no processo de requisição de misturador, a sua identidade e a do misturador são reveladas.
- Anonimato da rota em relação a nós pertencentes à rota: assim como os nós fora da rota, podem descobrir os demais intermediários, supor os possíveis nós origens e correlacionar as transmissões referentes ao mesmo pacote. Além disso, sabem a identidade do nó destino, presente em qualquer pacote, e a distância entre eles, através do campo *path* da resposta.

O protocolo CARP não provê anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: não são capazes de associar os nós pertencentes a cada rota, porque há anonimato da identidade. Porém, podem correlacionar transmissões referentes ao mesmo pacote através dos ataques de análise de tempo, de conteúdo e de volume.
- Anonimato da rota em relação a nós pertencentes à rota: assim como os nós fora da rota, podem correlacionar as transmissões referentes ao mesmo pacote. Em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota e a distância relativa entre eles, através do campo *node-pseudonym-list*, que contém os pseudônimos dos nós na rota.

O protocolo ASRP provê parcialmente anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: não são capazes de associar os nós pertencentes a cada rota, porque há anonimato da identidade. Em situações de pouco tráfego, transmissões referentes ao mesmo pacote de dados podem ser correlacionadas através dos ataques de análise de tempo e de volume. Como discutido na seção 4.3.2, o pacote de resposta e o de dados podem ser seguidos em certas situações.

- Anonimato da rota em relação a nós pertencentes à rota: em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota, através dos ataques de conteúdo e de volume aos pacotes que cada um correlacionou pelo ataque de reconhecimento de fluxo.

O protocolo AnonDSR provê parcialmente anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: considerando apenas o processo de roteamento, não são capazes de associar os nós pertencentes a cada rota, porque há anonimato da identidade. Porém, durante o estabelecimento dos parâmetros de segurança entre dois nós, a rota entre eles é revelada. Não podem correlacionar as transmissões referentes ao mesmo pacote, porque não podem realizar os ataques de análise de tempo, de conteúdo e de volume.

- Anonimato da rota em relação a nós pertencentes à rota: em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota, através do ataque de conteúdo à requisição referente aos pacotes correlacionados, individualmente, através do ataque de reconhecimento de fluxo. Pelo tamanho do pacote de resposta ou de dados, podem descobrir sua distância relativa.

O protocolo ODAR não provê anonimato da rota em relação a nós fora e a pertencentes à rota:

- Anonimato da rota em relação a nós fora da rota: considerando apenas o processo de roteamento, não são capazes de associar os nós pertencentes a

cada rota, porque há anonimato da identidade. Porém, o servidor de chaves envia, periodicamente, um pacote, que contém um campo, no qual os nós inserem suas identidades, antes de encaminhá-lo. Assim, é possível descobrir a topologia da rede. Podem correlacionar as transmissões referentes ao mesmo pacote, através do ataque de conteúdo.

- Anonimato da rota em relação a nós pertencentes à rota: assim como os nós fora da rota, podem correlacionar as transmissões referentes ao mesmo pacote. Em conluio, consecutivos ou não, podem descobrir que pertencem à mesma rota, através do campo *bDest*, e sabem a distância relativa entre eles e a quantidade de nós na rota, através do campo *bSize* dos pacotes de requisição e de resposta.

4.4 Outras Comparações

Esta seção compara os protocolos em relação aos mecanismos utilizados para prover anonimato, às técnicas de validação e algoritmos criptográficos usados pelos pesquisadores, à escalabilidade (número de operações criptográficas) e a outras características.

4.4.1 Mecanismos para Prover Anonimato

O Quadro 4.6 cita os mecanismos utilizados por cada protocolo, a fim de prevenir ataques de análise de tráfego e prover anonimato. Nota-se que, em geral, utilizam os mesmos mecanismos, diferindo apenas nos detalhes de como os empregam.

O mecanismo de criptografia salto a salto protege contra ataques de conteúdo, porém não em relação a nós pertencentes à rota, em conluio. Além disso, há o problema de gerenciar as chaves compartilhadas. *Paddings* aleatórios protegem contra o ataque de volume, porém podem causar *overhead* de carga. A técnica de difusão é utilizada para proteger os destinatários e a de *mixing*, os pacotes verdadeiros e evitar que sejam correlacionados e/ou seguidos até origem ou destino. Porém, causam *overhead* de tráfego. Pseudônimos dinâmicos são utilizados para proteger emissores e destinatários e evitar a descoberta de padrões de comunicação e de movimento. O problema é manter e atualizar os pseudônimos. *Trapdoor* protege destinatários,

Quadro 4.6: Mecanismos Utilizados para Prover Anonimato

Protocolo	**Mecanismos**
ANODR	Criptografia salto a salto do conteúdo do pacote de dados, difusão, *Onion Routing*, pseudônimos dinâmicos, requisição e resposta com tamanho único e fixo, técnicas *mixing*, *trapdoor* de chaves simétricas
SDAR	Difusão, *Onion Routing*, *paddings* aleatórios, *trapdoor* de chaves assimétricas
ASR	Criptografia salto a salto do conteúdo do pacote de dados, difusão, técnicas *mixing*, resposta com tamanho único e fixo, *trapdoor* de chaves simétricas
MASK	Criptografia salto a salto do conteúdo do pacote de dados, difusão, identificadores de enlace dinâmicos, pseudônimos, técnicas *mixing*, requisição e resposta com tamanho único e fixo
ANDSR	Difusão, misturadores
CARP	Difusão, pseudônimos, *trapdoor* de chaves simétricas
ASRP	Criptografia salto a salto do conteúdo do pacote de dados, difusão, pseudônimos, requisição e resposta com tamanho único e fixo, *trapdoor* de chaves assimétricas
AnonDSR	Difusão, *Onion Routing*, *paddings* aleatórios, pseudônimos, técnicas *mixing*, *trapdoor* de chaves simétricas
ODAR	*Bloom Filter*, difusão, requisição e resposta com tamanho único e fixo, técnicas *mixing*, *trapdoor* de HMAC

mas é preciso compartilhar um segredo e há o problema de *overhead* computacional, imposto aos nós, para que possam descobrir se os pacotes são destinados a eles.

4.4.2 Técnicas de Validação e Algoritmos Criptográficos

O Quadro 4.7 cita as técnicas utilizadas pelos pesquisadores para validar suas propostas e os algoritmos criptográficos usados na simulação. Nota-se que a maioria opta por simular os protocolos e os resultados, normalmente, referem-se apenas a desempenho.

Quadro 4.7: Técnicas Utilizadas para Validação e Algoritmos Criptográficos

Protocolo	**Validação**	**Algoritmos Criptográficos**
ANODR	Simulação - QualNet	AES/Rijndael 128 bits ECAES 160 bits SHA-1 160 bits
SDAR	Simulação - ns-2	RSA 512 bits
ASR	—	—
MASK	Simulação - GloMoSim	RC-6
ANDSR	Verificação formal - LOTOS	—
CARP	Simulação - ns-2	Blowfish
ASRP	—	—
AnonDSR	Simulação - ns-2	AES/Rijndael 128 bits RSA 2048 bits SHA-1 160 bits
ODAR	Simulação - J-Sim	—

4.4.3 Escalabilidade

Os Quadros 4.8 a 4.13 citam o número de operações criptográficas realizadas durante as fases de requisição, de resposta e de transferência dos dados. Em alguns protocolos, apesar de os nós não saberem diretamente qual a chave correta, considera-se o melhor caso. Há protocolos em que algumas operações são utilizadas para garantir integridade e autenticidade; essas operações também são contadas. Nos protocolos ANODR, ASR e MASK, pode-se ter mais de um pacote de resposta para cada rota, mas o cálculo é feito considerando-se apenas um.

Em relação à fase de requisição, os protocolos MASK, CARP, ASR, ANODR e ODAR se destacam. O primeiro não realiza operações, mas isso ocorre, porque há um mecanismo de estabelecimento de chaves entre vizinhos, anterior ao processo de roteamento. Os próximos três, porque utilizam operações simétricas e o número de nós na rota não influencia. O último realiza apenas *hash* e o cálculo das posições a serem marcadas no filtro é feito apenas uma vez, e é utilizado para qualquer rota. Quanto à fase de resposta, os protocolos MASK, CARP e ODAR se destacam, porque realizam apenas operações simétricas e/ou *hash* e não dependem da quantidade

Quadro 4.8: Número de Operações Criptográficas na Fase de Requisição (1)

		ANODR	**SDAR**	**ASR**	**MASK**	**ANDSR**
Nó Origem	Simétricas	$3^*/0$	$2/0$	$2^*/0$	$0/0$	$0/0$
	Assimétricas	$0/0$	$2^*/0$	$0/0$	$0/0$	$4/0$
	Hash	0	1^*	0	0	0
Nó Intermediário	Simétricas	$1/1$	$1/1$	$0/1$	$0/0$	$0/0$
	Assimétricas	$0/0$	$2^*/1$	$0/0$	$0/0$	$1^\diamond/1$
	Hash	0	1^*	0	0	0
Nó Destino	Simétricas	$0/2$	$1/2$	$0/2$	$0/0$	$0/0$
	Assimétricas	$0/0$	$0/2h+2^*$	$0/0$	$0/0$	$0/2$
	Hash	0	$h+1$	0	0	0

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; * indica que há operações, cujo objetivo é integridade e autenticidade; $^\diamond$ indica que a operação é realizada somente por um nó.

Quadro 4.9: Número de Operações Criptográficas na Fase de Requisição (2)

		CARP	**ASRP**	**AnonDSR**	**ODAR**
Nó Origem	Simétricas	$2/0$	$0/0$	$2/0$	$0/0$
	Assimétricas	$0/0$	$2^*/0$	$2^*/0$	$0/0$
	Hash	0	0	1^*	1
Nó Intermediário	Simétricas	$0/1$	$0/0$	$1/1$	$0/0$
	Assimétricas	$0/0$	$0/1$	$1/0$	$0/0$
	Hash	0	0	0	1
Nó Destino	Simétricas	$0/1$	$0/0$	$0/h+2$	$0/0$
	Assimétricas	$0/0$	$1^*/1$	$0/h+2^*$	$0/0$
	Hash	0	0	1^*	1

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; $*$ indica que há operações, cujo objetivo é integridade e autenticidade.

Quadro 4.10: Número de Operações Criptográficas na Fase de Resposta (1)

		ANODR	**SDAR**	**ASR**	**MASK**	**ANDSR**
Nó Origem	Simétrica	$0/2^{\dagger}$	0/1	$0/1^{\dagger}$	0/1	0/0
	Assimétrica	$0/1^{\dagger}$	0/0	$0/1^{\dagger}$	0/0	0/1
	Hash	0	1	0	0	0
Nó Intermediário	Simétrica	$2^{*}/3^{\dagger}_{*}$	0/1	$2^{*}/2^{\dagger}_{*}$	1/1	0/0
	Assimétrica	$1/1^{\dagger}$	0/0	$1/1^{\dagger}$	0/0	0/0
	Hash	0	1^{*}	0	0	0
Nó Destino	Simétrica	1/0	h+1/0	1/0	1/0	0/0
	Assimétrica	1/0	0/0	1/0	0/0	1/0
	Hash	0	$2h+2^{*}$	0	0	0

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; * indica que há operações, cujo objetivo é integridade e autenticidade; † indica que há operações contabilizadas, considerando-se o melhor caso.

Quadro 4.11: Número de Operações Criptográficas na Fase de Resposta (2)

		CARP	**ASRP**	**AnonDSR**	**ODAR**
Nó Origem	Simétrica	0/1	0/0	0/1	0/0
	Assimétrica	0/0	0/1	$0/1^{*}$	0/0
	Hash	1	0	1^{*}	1
Nó Intermediário	Simétrica	0/0	0/0	0/1	0/0
	Assimétrica	0/0	$2^{*}/1$	0/0	0/0
	Hash	0	0	0	0
Nó Destino	Simétrica	2/0	0/0	h+1/0	0/0
	Assimétrica	0/0	1/0	$1^{*}/0$	0/0
	Hash	1	0	1^{*}	1

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; * indica que há operações, cujo objetivo é integridade e autenticidade.

Quadro 4.12: Número de Operações Criptográficas na Fase de Transferência (1)

		ANODR	**SDAR**	**ASR**	**MASK**	**ANDSR**
Nó Origem	Simétrica	1/0	h+1/0	1/0	1/0	0/0
	Assimétrica	0/0	0/0	0/0	0/0	3/0
	Hash	0	0	1	0	0
Nó Intermediário	Simétrica	1/1	0/1	1/1	1/1	0/0
	Assimétrica	0/0	0/0	0/0	0/0	$1^{\diamond}/1$
	Hash	0	0	$2^{\dagger}$	0	0
Nó Destino	Simétrica	0/1	0/1	0/1	0/1	0/0
	Assimétrica	0/0	0/0	0/0	0/0	0/3
	Hash	0	0	1	0	0

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; $^{\diamond}$ indica que a operação é realizada somente por um nó; † indica que há operações contabilizadas, considerando-se o melhor caso.

Quadro 4.13: Número de Operações Criptográficas na Fase de Transferência (2)

		CARP	**ASRP**	**AnonDSR**	**ODAR**
Nó Origem	Simétrica	1/0	$3^{*}/0$	h+1/0	0/0
	Assimétrica	0/0	0/0	0/0	0/0
	Hash	0	0	0	0
Nó Intermediário	Simétrica	0/0	$2^{*}/2^{*}$	0/1	0/0
	Assimétrica	0/0	0/0	0/0	0/0
	Hash	0	0	0	0
Nó Destino	Simétrica	0/1	$0/3^{*}$	0/1	0/0
	Assimétrica	0/0	0/0	0/0	0/0
	Hash	0	0	0	0

NOTA: h é o número de nós entre origem e destino; x/y indica cifragens/decifragens; * indica que há operações, cujo objetivo é integridade e autenticidade.

de nós na rota. Quanto à fase de transferência, a maioria realiza apenas operações simétricas; os protocolos ODAR, CARP, ANODR, MASK e ASRP se destacam. O primeiro não realiza operações e os últimos, realizam apenas simétricas. ASRP possui mais operações do que os demais, mas provê mecanismos de segurança de integridade e de autenticidade.

O protocolo MASK possui uma fase de autenticação entre vizinhos, na qual cada nó realiza duas funções *hash*, uma para gerar a chave master de sessão e a outra, para o autenticador, e mais duas para cada par de chave e identificador de enlace. O protocolo AnonDSR possui uma fase de estabelecimento de parâmetros de segurança, na qual a origem realiza duas cifragens assimétricas e um *hash*, e o destino, 2 decifragens assimétricas e um *hash*. O protocolo ODAR usa um *Bloom Filter* para esconder as identidades; cada nó realiza *1+k hashs*, para gerar o HMAC de sua identidade e inseri-lo no filtro (k é o número de funções *hash* associadas ao *Bloom Filter*).

4.4.4 Outras Características

Os Quadros 4.14 e 4.15 comparam os protocolos em relação a mecanismo de atualização, organização da topologia, enlaces assimétricos, múltiplas rotas e multidifusão. Todos são reativos, *unicast* e supõem que os enlaces são simétricos. A maioria é do tipo *flat*, com exceção do ANDSR, que possui misturadores e nós comuns. Somente os protocolos ANODR, ASR e AnonDSR possuem suporte a múltiplas rotas. Na realidade, os projetistas de ANODR deixam essa questão para os implementadores. No segundo, o destino envia uma resposta para cada requisição distinta e válida que recebe. No último, nós intermediários que possuem rotas válidas podem enviar resposta.

4.5 Considerações

Este capítulo apresentou uma análise qualitativa dos protocolos de roteamento anônimo para redes sem fio ad hoc móveis; verificações formais, simulações e testes são necessários para demonstrar e analisar quantitativamente as vulnerabilidades identificadas por este trabalho. Porém, é importante mencionar que todos os passos da análise são detalhados e realizados com base em uma única definição. A análise

Quadro 4.14: Comparação de Outras Características (1)

	ANODR	**SDAR**	**ASR**	**MASK**	**ANDSR**
Mecanismo de atualização	reativo	reativo	reativo	reativo	reativo
Organização da topologia	*flat*	*flat*	*flat*	*flat*	não uniforme
Suporte a enlaces assimétricos	não	não	não	não	não
Suporte a múltiplas rotas	sim	não	sim	sim	não
Capacidade de multidifusão	não	não	não	não	não

Quadro 4.15: Comparação de Outras Características (2)

	CARP	**ASRP**	**AnonDSR**	**ODAR**
Mecanismo de atualização	reativo	reativo	reativo	reativo
Organização da topologia	*flat*	*flat*	*flat*	*flat*
Suporte a enlaces assimétricos	não	não	não	não
Suporte a múltiplas rotas	não	não	não	não
Capacidade de multidifusão	não	não	não	não

considera apenas anonimato em relação à camada de rede, mais especificamente das informações de roteamento, apesar de ser possível inferir informações importantes da rede através das outras camadas. Não são analisadas questões como segurança do protocolo contra ataques ativos, nem análise de tráfego usando ataques ativos (são aspectos importantes e alguns protocolos os consideram). Verificou-se que nenhum protocolo garante todas as propriedades analisadas e que, em geral, possuem características em comum e utilizam os mesmos mecanismos para prover anonimato, diferindo apenas nos detalhes de como os empregam, e que nem todos se preocupam com outras questões de segurança, como ataques de repetição e de modificação; também é difícil conciliar desempenho e segurança. O próximo capítulo apresenta uma análise formal do protocolo ANODR.

5 *Especificação Formal do Protocolo ANODR*

Neste capítulo, o protocolo ANODR é especificado formalmente usando a técnica de descrição formal LOTOS - *Language of Temporal Ordering Specification* e o pacote de ferramentas CADP - *CAESAR/ALDEBARAN Development Package*. Além disso, algumas de suas limitações são discutidas e modificações são sugeridas para melhorar sua segurança quanto a anonimato. Escolheu-se o protocolo ANODR por ter sido a primeira proposta de protocolo de roteamento anônimo para redes sem fio ad hoc móveis e porque é o mais seguro, de acordo com os resultados da análise realizada neste trabalho e apresentada no capítulo anterior. Pelas pesquisas realizadas, é a primeira especificação formal do protocolo ANODR e a segunda de protocolos de roteamento anônimo para redes sem fio ad hoc móveis (dos protocolos estudados, somente ANDSR é especificado formalmente).

5.1 Análise Formal

Esta seção apresenta uma visão geral da técnica de descrição formal LOTOS e do pacote de ferramentas CADP, a modelagem e a verificação do protocolo ANODR.

5.1.1 LOTOS e CADP

A fim de verificar formalmente o protocolo ANODR, utilizou-se a técnica de descrição formal LOTOS - *Language of Temporal Ordering Specification* (IS8807, 1988) e o pacote de ferramentas CADP - *CAESAR/ALDEBARAN Development Package* (VASY-INRIA, 2007). Optou-se por LOTOS, porque é uma técnica consolidada e padronizada, desenvolvida pela ISO - *International Organization for Standardization*, e de acordo com LEDUC & GERMEAU (2000), é apropriada para especificar protoco-

los de segurança, devido à sua flexibilidade; além disso, um dos protocolos estudados, ANDSR, foi especificado em LOTOS. CADP é um dos pacotes de ferramentas mais completos e utilizados para verificação, teste e simulação de especificações LOTOS.

A técnica LOTOS é constituída por dois componentes: LOTOS básica e LOTOS completa. O primeiro possui recursos para a descrição do comportamento dos processos e suas interações; é baseado nas álgebras de processos CCS - *Calculus of Communicating System* e CSP - *Communicating Sequential Process*. O segundo inclui, além do primeiro, recursos para a definição de estruturas de dados e de valores; é baseado na linguagem de tipos de dados abstratos ACT ONE - *Abstract Data Type Formalism*.

A Figura 5.1 apresenta a sintaxe básica para definir um tipo abstrato de dados, que é uma álgebra formada por *sorts* (nome dado a um conjunto de elementos pertencentes ao mesmo domínio; corresponde ao conceito de tipo das linguagens de programação (MAÑAS, 1988)) com operações e propriedades: <identificador> é o nome do tipo; <lista de identificadores> contém o(s) nome(s) do(s) *sort(s)* definido(s) no tipo; <lista de definições de tipos importadas> contém tipos, cujos sorts e/ou operações são utilizados no tipo que está sendo definido; <lista de operações> contém as operações que atuam sobre os *sorts*; <lista de equações> contém propriedades associadas às operações; <lista de variáveis> contém as variáveis que são utilizadas nas equações para o <sort>.

```
TYPE <identificador> IS <lista de definições de tipos importadas>
SORTS <lista de identificadores>
OPNS   <lista de operações>
EQNS
          FORALL <lista de variáveis>
          OFSORT <sort>
                    <lista de equações>
ENDTYPE
```

Figura 5.1: Sintaxe de um Tipo Abstrato de Dados (BOLOGNESI; BRINKSMA, 1987)

Uma especificação em LOTOS é constituída por um ou mais processos que modelam o comportamento do sistema. Um processo é composto por ações (eventos) internas não observáveis e externas observáveis - interações (sincronizações) com outros processos, através de pontos de interação (portas).

As Figuras 5.2 e 5.3 mostram a sintaxe em LOTOS completa para definir uma especificação e um processo típicos: <identificador> é o nome da especi-

ficação ou do processo; <lista de portas> contém as portas através das quais há interação com o meio externo; <lista de parâmetros> contém declarações de variáveis; <funcionalidade> indica o tipo de comportamento, que pode ser *exit* - término com sucesso e habilitação de outro processo ou *noexit* - não há término; <definição de tipos> contém as definições de estruturas de dados ou o nome da biblioteca que as define; <expressão de comportamento> contém as ações internas e externas de um processo. O Quadro 5.1 apresenta os operadores LOTOS utilizados na especificação.

```
SPECIFICATION  <identificador> [lista de portas] (lista de parâmetros): <funcionalidade>
        <definição de tipos>
BEHAVIOUR
        <expressão de comportamento>
WHERE
        <definição de tipos>
        <definição de processos>
ENDSPEC
```

Figura 5.2: Sintaxe de uma Especificação em LOTOS (BOLOGNESI; BRINKSMA, 1987)

```
PROCESS <identificador> [lista de portas] (lista de parâmetros): <funcionalidade> :=
        <expressão de comportamento>
WHERE
        <definição de tipos>
        <definição de processos>
ENDPROC
```

Figura 5.3: Sintaxe de um Processo em LOTOS (BOLOGNESI; BRINKSMA, 1987)

O pacote CADP possui uma interface gráfica, chamada *Eucalyptus*, através da qual é possível acessar todos os arquivos (especificações, arquivos BCG - *Binary Coded Graphs*, entre outros) e executar rápida e facilmente suas ferramentas, dentre as quais, destacam-se:

- Caesar: compila e verifica especificações escritas em LOTOS; a partir da especificação, gera um programa em C e um arquivo que contém o autômato de estados finitos correspondente (LTS - *Labelled Transition System*), em formato textual - AUT ou de grafo - BCG.
- Caesar.adt: compila tipos abstratos de dados LOTOS em uma biblioteca C.
- BCG_edit: edita representações PS de grafos em formato BCG.
- BCG_draw: gera representações PS de grafos em formato BCG.

Quadro 5.1: Principais Operadores LOTOS

Nome	**Sintaxe**	**Significado**
ação interna	i	ação não observável
prefixo de ação	i;a;B	seqüência de eventos
escolha	A[]B	escolha não determinística entre os processos A e B
declaração de valor	g!E	porta g oferece o valor da expressão E
declaração de variável	g?x:t	porta g recebe, através da variável x, valores do *sort* t
paralelismo independente	A|||B	processos A e B são executados paralelamente e são independentes
paralelismo geral - sincronização parcial	A|[$g_1, ..., g_n$]|B	processos A e B são executados paralelamente e devem sincronizar através das portas $g_1, ..., g_n$
paralelismo dependente - sincronização total	A||B	processos A e B são executados paralelamente e devem sincronizar através de todas as portas
ocultação	hide $g_1, ..., g_n$ in B	portas $g_1, ..., g_n$ do processo B não são visíveis
instanciação de processo	A[$g_1, ..., g_n$]	instanciação do processo A, cujas portas são $g_1, ..., g_n$
término com sucesso	exit	processo termina com sucesso
término com sucesso com oferecimento de valor	exit(x)	processo termina com sucesso e oferece o valor x
composição seqüencial (habilitação)	A>>B	o processo B é executado após o término com sucesso do processo A
composição seqüencial (habilitação com passagem de valor)	A >> accept x:t in B	o processo B é executado após o término com sucesso do processo A e recebe o valor x do *sort* t de A

Fonte: BOLOGNESI & BRINKSMA (1987).

- BCG_min: minimiza LTSs.
- Bisimulator: verifica e compara LTSs. Equivalência forte (toda ação de um LTS possui uma ação correspondente no outro LTS) e equivalência de observação (somente ações visíveis devem corresponder) são algumas das opções de relação de equivalência que podem ser escolhidas.
- Executor: executa aleatoriamente a especificação a fim de testá-la. Há três estratégias de teste: determinística, não determinística com início aleatório e não determinística com escolha do início.
- Evaluator: detecta *livelocks*.
- Exhibitor e Terminator: detectam *deadlocks* (impasses).
- XSimulator e Interactive Simulator: simulam interativamente a execução de uma especificação.

5.1.2 Modelagem

Para a modelagem do protocolo ANODR, adotou-se o estilo de especificação orientado para recursos, no qual eventos internos e externos são modelados; para a modelagem do serviço, o estilo orientado para restrições, no qual há somente eventos observáveis. A especificação usa LOTOS completa, ou seja, inclui além da descrição de seu comportamento, tipos abstratos de dados para a modelagem de operações criptográficas, pacotes e mensagens de *status*, por exemplo. Os tipos abstratos de dados utilizados na especificação são definidos em uma biblioteca chamada ANODRLIB (Apêndice A), que utiliza duas bibliotecas predefinidas (BOOLEAN e NATURAL), incluídas no pacote CADP.

A modelagem baseia-se no modelo genérico desenvolvido por ARAUJO (2005) ao especificar o protocolo ANDSR: ao invés de modelar cada nó considerando diferentes topologias, modela-se o comportamento apresentado por um grupo de nós. Outras idéias inspiradas no trabalho de ARAUJO são: utilização de portas para representar a interação com a camada superior e com a tabela de roteamento, e representação do tempo através de uma mensagem *synchronization*; quando um nó A, vizinho dos nós B e C (sendo que B está mais perto de A do que C), envia um pacote em difusão, B o recebe e envia a mensagem *synchronization* a C, para que este esteja apto a

recebê-lo também. Duas idéias inspiradas no trabalho de LEDUC & GERMEAU (2000) são a utilização de eventos especiais e de portas para envio de mensagens de *status*, quando pontos importantes são atingidos, e a modelagem do conhecimento do intruso como uma estrutura de dados.

A Figura 5.4 representa graficamente a especificação de ANODR. O processo *communication establishment* (serviço oferecido pela camada de rede, mais especificamente por ANODR) é composto por três subprocessos: *route discovery* (descoberta de rota), *data forwarding* (transmissão de dados com sucesso) e *maintenance* (manutenção, no caso de detecção de quebra de enlace). Cada um deles, por sua vez, é composto por seis subprocessos: *SN* (nó origem), *IIN* (nó intermediário pertencente à rota), *ION* (nó intermediário não pertencente à rota), *EN* (nó externo), *DN* (nó destino) e *channel* (canal de comunicação), que modelam o comportamento de cada tipo de nó e suas interações (envio e recebimento de pacotes) através do canal (portas *out* e *in*). As portas *SL*, *RT* e *AS* são utilizadas para representar a interação dos nós com a camada superior (além de emitir mensagens de *status* relacionadas ao funcionamento do protocolo) e com suas tabelas de roteamento, e emitir mensagens de *status* relacionadas a anonimato, respectivamente.

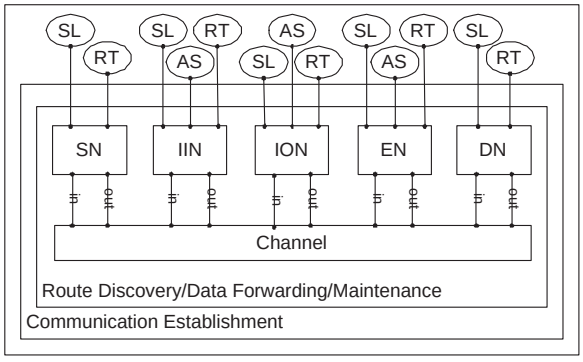

Figura 5.4: Representação Gráfica da Especificação do Protocolo ANODR

A Figura 5.5 mostra parte da especificação em LOTOS do protocolo (a especificação completa pode ser vista no Apêndice B). Quatro cenários são modelados: o nó origem conhece (processo *RouteDiscoveryProcess1*) ou não a rota (processo *RouteDiscoveryProcess2*), e o encaminhamento dos pacotes de dados termina com sucesso (processo *DataForwardingProcess*) ou há a detecção de quebra de enlace (processo *MaintenanceProcess*). Os processos *DataForwardingProcess* e *MaintenanceProcess* possuem como parâmetros, valores estabelecidos durante os processos *RouteDiscoveryProcess1* e *RouteDiscoveryProcess2*: as chaves secretas estabelecidas pelos nós

consecutivos na rota (intermediário/origem e destino/intermediário), durante a descoberta de rota, que são utilizadas para gerar a seqüência de pseudônimos e cifrar salto a salto a carga útil dos pacotes de dados, e três estruturas que representam o conhecimento dos nós intermediários pertencente e não pertencente à rota, e externo.

```
specification anodr_spe[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]: noexit

library ANODRLIB endlib

behaviour

CommunicationEstablishmentProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]

where

process CommunicationEstablishmentProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,
sl_DN,rt_DN]: noexit :=
(
 RouteDiscoveryProcess1[sl_SN,rt_SN]
 >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge: Knowledge_Sort
 in
 DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN] (k1, k2,
 IINKnowledge, IONKnowledge, ENKnowledge)
)
[]
(
 RouteDiscoveryProcess1[sl_SN,rt_SN]
 >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge: Knowledge_Sort
 in
 MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN] (k1, k2, IINKnowledge,
 IONKnowledge, ENKnowledge)
)
[]
(
 RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
 >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge: Knowledge_Sort
 in
 DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN] (k1, k2,
 IINKnowledge, IONKnowledge, ENKnowledge)
)
[]
(
 RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
 >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge: Knowledge_Sort
 in
 MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN] (k1, k2, IINKnowledge,
 IONKnowledge, ENKnowledge)
)
>>
CommunicationEstablishmentProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]

where

process RouteDiscoveryProcess1[sl_SN, rt_ SN]: exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort,
Knowledge_Sort, Knowledge_Sort) :=
…
endproc

process RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]:
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
...
endproc

process DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
(kseed_SN_IIN, kseed_IIN_DN: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
Knowledge_Sort): exit :=
…
endproc

process MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN]
(kseed_SN_IIN, kseed_IIN_DN: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
Knowledge_Sort): exit :=
...
endproc

endspec
```

Figura 5.5: Especificação do Protocolo ANODR

A Figura 5.6 mostra parte da especificação do processo *RouteDiscoveryProcess2*, que modela a descoberta de rotas. Os subprocessos *Source*, *IntermediateInRoute*, *IntermediateOutRoute*, *External* e *Destination*, que modelam os tipos de nós, são paralelos e independentes (operador |||) e sincronizam (operador |[]|) com o subprocesso *Channel*, que modela o canal de comunicação. Tal sincronização ocorre através das portas *in_SN*, *out_SN*, *in_IIN*, *out_IIN*, *in_ION*, *out_ION*, *in_EN*, *out_EN*, *in_DN* e *out_DN*, onde *in_XN* e *out_XN* são as portas de entrada e saída de pacotes do nó X; essas portas não são visíveis externamente (operador *hide*). O subprocesso *Source* possui como parâmetros a identidade do nó origem (ID_{SN}), a identidade do nó destino (ID_{DN}) e a chave secreta compartilhada por ambos (K_{SD}); os subprocessos *IntermediateInRoute*, *IntermediateOutRoute* e *External*, uma estrutura que representa o conhecimento inicial de cada um dos nós (*knowledge_IIN*, *knowledge_ION* e *knowledge_EN*); o subprocesso *Destination*, a chave secreta compartilhada com a origem (K_{SD}). Inicialmente, essas estruturas estão vazias, mas a cada pacote recebido, há a inserção de cada um de seus campos (campos cifrados, mas com chaves conhecidas, são armazenados em claro). As interações do nó X com a camada superior e com a sua tabela de roteamento e as emissões de *status* quanto a anonimato ocorrem através das portas *SL_XN*, *RT_XN* e *AS_XN*, únicas portas visíveis externamente. Os processos *DataForwardingProcess* e *MaintenanceProcess*, que representam, respectivamente, o encaminhamento de dados e a detecção de quebra de enlace, possuem a mesma estrutura. O processo *RouteDiscoveryProcess1* contém apenas a solicitação da rota, a verificação da tabela de roteamento e a constatação de que a rota já é conhecida pelo nó origem.

A Figura 5.7 mostra parte da especificação em LOTOS do serviço que deve ser realizado por ANODR (a especificação completa pode ser vista no Apêndice C), ou seja, o estabelecimento anônimo de rotas (processo *RouteDiscoveryProcess*), o encaminhamento anônimo dos dados (processo *DataForwardingProcess*) e a manutenção no caso de detecção de quebra de enlace (processo *MaintenanceProcess*). De forma detalhada:

- Quando um nó origem X deseja se comunicar com um nó destino Y, verifica sua tabela de roteamento e caso não conheça uma rota, inicia o processo de descoberta enviando um pacote de requisição.

- Esse pacote deve ser recebido pelo nó destino e não pode revelar as identidades

```
Process RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN,
rt_DN]: exit( SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
hide in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN in
(
    Source[sl_SN, rt_SN, in_SN, out_SN](ID_SN, ID_DN, K_SD)
    |||
    IntermediateInRoute[sl_IIN, rt_IIN, in_IIN, out_IIN](knowledge_IIN)
    |||
    IntermediateOutRoute[sl_ION, rt_ION, in_ION, out_ION](knowledge_ION)
    |||
    External[sl_EN, rt_EN, in_EN, out_EN](knowledge_EN)
    |||
    Destination[sl_DN, rt_DN, in_DN, out_DN](K_SD)
)
|[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]|
    Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]

where

    process Source[sl_SN, rt_SN, in_SN, out_SN](ID_SN, ID_DN: Identity_Sort, K_SD:       SecretKey_Sort):
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    …
    endproc

    process IntermediateInRoute[sl_IIN, rt_IIN, as_IIN, in_IIN, out_IIN](IINKnowledge: Knowledge_Sort):
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort)  :=
    …
    endproc

    process IntermediateOutRoute[sl_ION, as_ION, in_ION, out_ION](IONKnowledge: knowledge_Sort):
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    …
    endproc

    process External[sl_EN, as_EN, in_EN, out_EN](ENKnowledge: Knowledge_Sort):
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    …
    endproc

    process Destination[sl_DN, rt_DN, in_DN, out_DN](K_SD: SecretKey_Sort):
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort):=
    …
    endproc

    process Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out DN]:
    exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    …
    endproc
endproc
```

Figura 5.6: Especificação do Processo *RouteDiscoveryProcess2*

dos nós origem e destino nem o *venue* do primeiro a qualquer nó que o receba.

- O nó destino, após recebê-lo, envia um pacote de resposta.
- Esse pacote deve ser recebido pelo nó origem e não pode revelar as identidades dos nós origem e destino nem o *venue* do último a qualquer nó que o receba.
- Com a rota estabelecida, o nó origem pode enviar pacotes de dados.
- Esses pacotes devem ser recebidos pelo nó destino e não podem revelar as identidades dos nós origem e destino nem os seus *venues* a qualquer nó que o receba.
- Ao detectar a quebra de um enlace (após enviar um pacote de dados, todo

nó espera, por determinado período de tempo, a confirmação local de recebimento), um nó intermediário pertencente à rota deve enviar um pacote de erro, que deve chegar até a origem.

- Todos os nós devem descartar pacotes duplicados ou que não são destinados (localmente) a eles.

```
specification anodr_serv[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]: noexit

library ANODRLIB endlib

behaviour

Service[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]

where

process Service[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,
sl_DN,rt_DN]: noexit :=
(
 RouteDiscoveryProcess1[sl_SN,rt_SN]
 >> DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
)
[]
(
 RouteDiscoveryProcess1[sl_SN,rt_SN]
 >> MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN]
)
[]
(
 RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
 >> DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
)
[]
(
 RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]
 >> MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN]
)
>> Service[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]

where

process RouteDiscoveryProcess1[sl_SN, rt_ SN]: exit :=
...
endproc

process RouteDiscoveryProcess2[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]:
exit :=
...
endproc

process DataForwardingProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,as_IIN,sl_ION,as_ION,sl_EN,as_EN,sl_DN,rt_DN]: exit
:=
...
endproc

process MaintenanceProcess[sl_SN,rt_SN,sl_IIN,rt_IIN,sl_ION,sl_EN,sl_DN,rt_DN]: exit :=
...
endproc

endspec
```

Figura 5.7: Especificação do Serviço

Nesta fase do trabalho, modelou-se o anonimato das identidades e dos *venues* dos nós origem e destino. Questões como mobilidade, envio de pacotes fictícios e diferentes pares origem/destino não foram modelados, por isso anonimato da rota e privacidade da localização e do padrão de movimento não foram verificados. A especificação em relação a anonimato da identidade verifica se está presente nos pacotes

e é revelada. Em relação a *venue*, considera somente a existência de contadores de saltos explícitos ou implícitos; a capacidade de os atacantes seguirem os pacotes não é modelada.

Na especificação do serviço, há a ocorrência de eventos especiais nas portas *AS* dos nós *IIN* (intermediário pertencente à rota), *ION* (intermediário não pertencente à rota) e *EN* (externo): mensagens de *status* relacionadas às propriedades de anonimato que devem ser garantidas pelo protocolo (*does_not_discover_IDsource*, *does_not_discover_IDdestination*, *does_not_discover_venue_source*, *does_not_discover_venue_destination*).

Na especificação do protocolo, há operações que verificam se as identidades dos nós origem e destino são conhecidas pelos nós *IIN*, *ION* e *EN* (*verify_anonymityIDsrc (knowledge_X)* e *verify_anonymityIDdest(knowledge_X)*, onde *knowledge_X* é o conhecimento do nó *X*); o resultado (*does_not_discover_IDsource*, *does_not_discover_ID destination*, *knows_IDsource* ou *knows_IDdestination*) é enviado na porta *AS*. Há também uma operação que indica o número de saltos percorridos pelo pacote de requisição, ou seja, a distância em relação ao nó origem (*hopcounter(tbo)*, onde *tbo* é o *onion* da requisição, cujo tamanho aumenta a cada salto); essa operação pode ser realizada somente por nós internos (*IIN* e *ION*), pois são os únicos que podem realizar o ataque de volume.

5.1.3 Verificação

Através do pacote CADP, três tipos de verificação foram realizadas: ausência de impasses, vivacidade do sistema e equivalência de observação entre a especificação do protocolo e a do serviço esperado.

Utilizando as ferramentas Exhibitor e Evaluator, verificou-se que a especificação do protocolo ANODR (Apêndice B) não contém impasses e é reinicializável (Figura 5.8).

Para a verificação de equivalência entre a especificação do protocolo e a do serviço, utilizaram-se as ferramentas Caesar.adt e Caesar para a compilação das especificações e geração dos LTSs em formato BCG e Bisimulator para a comparação dos LTSs. A verificação foi dividida em duas partes: funcionamento e anonimato.

Para a verificação do funcionamento, retiraram-se os itens das especificações

```
------------------------------------------------ Cleared -----------------------------------------------
bcg_open anodr_spe.bcg exhibitor -bfs  -depth "0" < /usr/local/cadp/src/eucalyptus/deadlock.seq | tee "dead
lock.seq"
bcg_open: using ``/usr/local/cadp/bin.iX86/exhibitor.a''
bcg_open: running ``exhibitor -bfs -depth 0'' for ``./anodr_spe.bcg''
*** searching for sequence of the form:

<any>* . <deadlock>

*** using breadth-first search algorithm

*** no sequence found

*** no prefix of the sequence has been recognized

-------------------------------------------------------------------------------------------------------
bcg_open anodr_spe.bcg evaluator  -verbose  -diag "livelock.bcg" "/usr/local/cadp/src/xtl/livelock.mcl"
bcg_open: using ``/usr/local/cadp/bin.iX86/evaluator.a''
bcg_open: running ``evaluator -verbose -diag livelock.bcg /usr/local/cadp/src/xtl/livelock.mcl'' for ``./an
odr_spe.bcg''
-- evaluator 3.5 -- R. Mateescu and M. Sighireanu (INRIA Rhone-Alpes) --

evaluator: preprocessing of ``livelock''
evaluator: syntax analysis of ``livelock''
evaluator: semantic analysis of ``livelock''
evaluator:    - variable binding
evaluator:    - translation in positive normal form
evaluator:    - test of alternation 1
evaluator:    - translation into regular modal equation systems
evaluator:    - regular expression elimination
evaluator:    - translation into modal equation systems
evaluator: resolution of ``livelock''
evaluator: diagnostic of ``livelock''

FALSE
(consult diagnostic in file ``livelock.bcg'')
-------------------------------------------------------------------------------------------------------
```

Figura 5.8: Verificação de Impasses e de *Livelocks*

relacionados a anonimato. O LTS correspondente à especificação do protocolo possui 166 estados e 170 transições e o do serviço, 57 estados e 60 transições. Através da ferramenta Bisimulator, verificou-se a equivalência de observação entre as duas especificações (resultado *TRUE*), ou seja, o protocolo funciona corretamente em todos os cenários modelados, isto é, é capaz de estabelecer rotas, encaminhar os dados e detectar quebras de enlace (Figura 5.9).

```
-------------------------------------------------------------------------------------------------------
bcg_open anodr_spe.bcg bisimulator   -equal -observational  -bfs -diag "bisimulator.bcg" ./anodr_serv
ice.bcg
bcg_open: using ``/usr/local/cadp/bin.iX86/bisimulator.a''
bcg_open: running ``bisimulator -equal -observational -bfs -diag bisimulator.bcg ./anodr_service.bcg'
' for ``./anodr_spe.bcg''

TRUE
-------------------------------------------------------------------------------------------------------
```

Figura 5.9: Verificação de Equivalência de Observação - Funcionamento

O processo de verificação de anonimato foi dividido em doze etapas: verificação do anonimato das identidades e dos *venues* dos nós origem e destino, em relação a nós intermediários pertencentes à rota (*IIN*), a não pertencentes (*ION*) e a externos (*EN*). O LTS completo correspondente à especificação do protocolo possui

214 estados e 218 transições e o do serviço, 105 estados e 108 transições. Todas as verificações, exceto *venue* da origem e identidade do destino, em relação a *IIN* e *ION*, e *IIN*, respectivamente, obtiveram o resultado *TRUE* (a tela resultante é semelhante à da Figura 5.9), ou seja, o anonimato da identidade da origem e o do *venue* do destino em relação a todos os nós, o anonimato da identidade do destino em relação a *ION* e a *EN* e o anonimato do *venue* da origem em relação a *EN* são assegurados.

A Figura 5.10 mostra o resultado obtido ao se verificar anonimato da identidade do destino em relação a *IIN*, anonimato do *venue* da origem em relação a *IIN* e a *ION*. Bisimulator indica que não há equivalência de observação entre as especificações (resultado *FALSE*), ou seja, a propriedade de anonimato analisada não é assegurada.

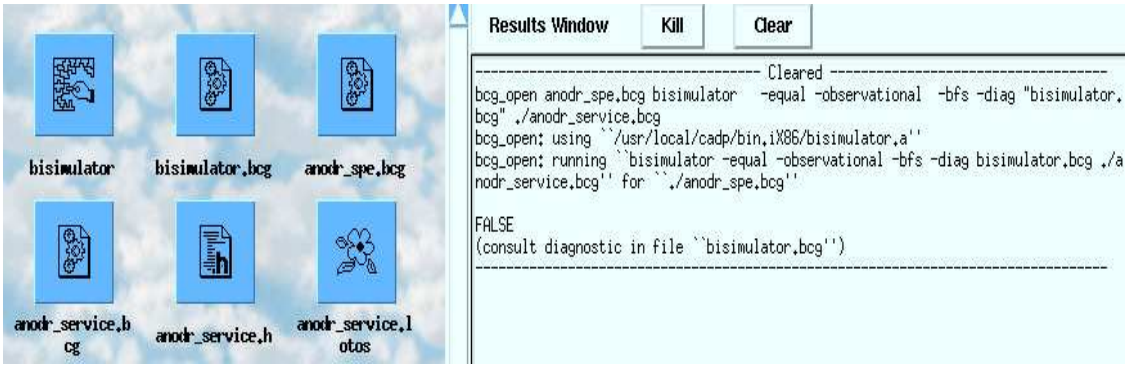

Figura 5.10: Verificação de Equivalência de Observação - Anonimato da Identidade do Destino em Relação a *IIN*

Além de indicar que não há equivalência, Bisimulator gera um arquivo (no caso, bisimulator.bcg) com o problema encontrado ao se realizar a verificação. Esse arquivo pode ser visualizado através da ferramenta BCG_draw (Figura 5.11) ou ser executado através do XSimulator (Figura 5.12). Essas figuras referem-se à verificação de anonimato do *venue* da origem em relação a *ION*: a especificação do serviço espera o evento *AS_ION ! does_not_discover_venue_source*, porém a especificação do protocolo gera o evento *AS_ION ! knows_venue_source !2* quando *ION* recebe a requisição, onde o valor *2* indica a distância em relação à origem (Figura 5.13). Na verificação do anonimato da identidade do destino em relação a *IIN*, o evento esperado é *as_IIN ! does_not_discover_IDdestination* e o evento gerado quando *IIN* recebe a resposta é *as_IIN ! knows_IDdestination*. No caso do anonimato do *venue* da origem em relação a *IIN*, espera-se o evento *AS_IIN ! does_not_discover_venue_source*,

porém o evento gerado é *AS_IIN ! knows_venue_source !1*.

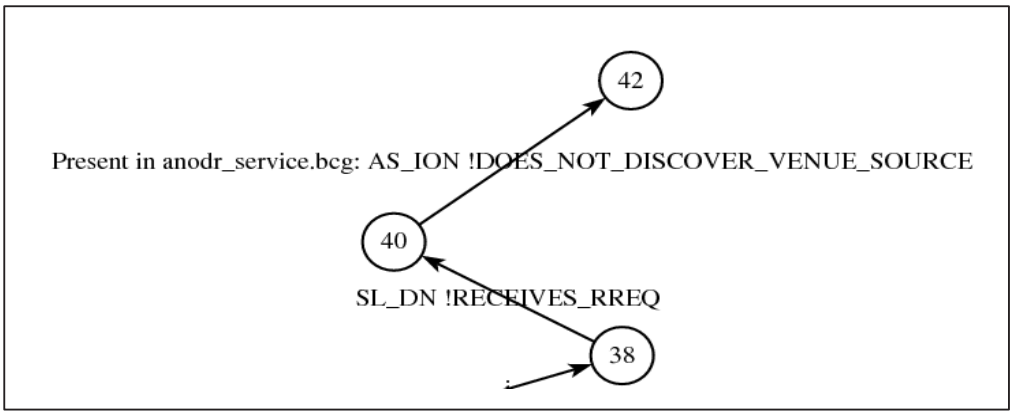

Figura 5.11: Parte do Arquivo com a Indicação do Problema Encontrado na Verificação de Anonimato do *Venue* da Origem em Relação a *ION*

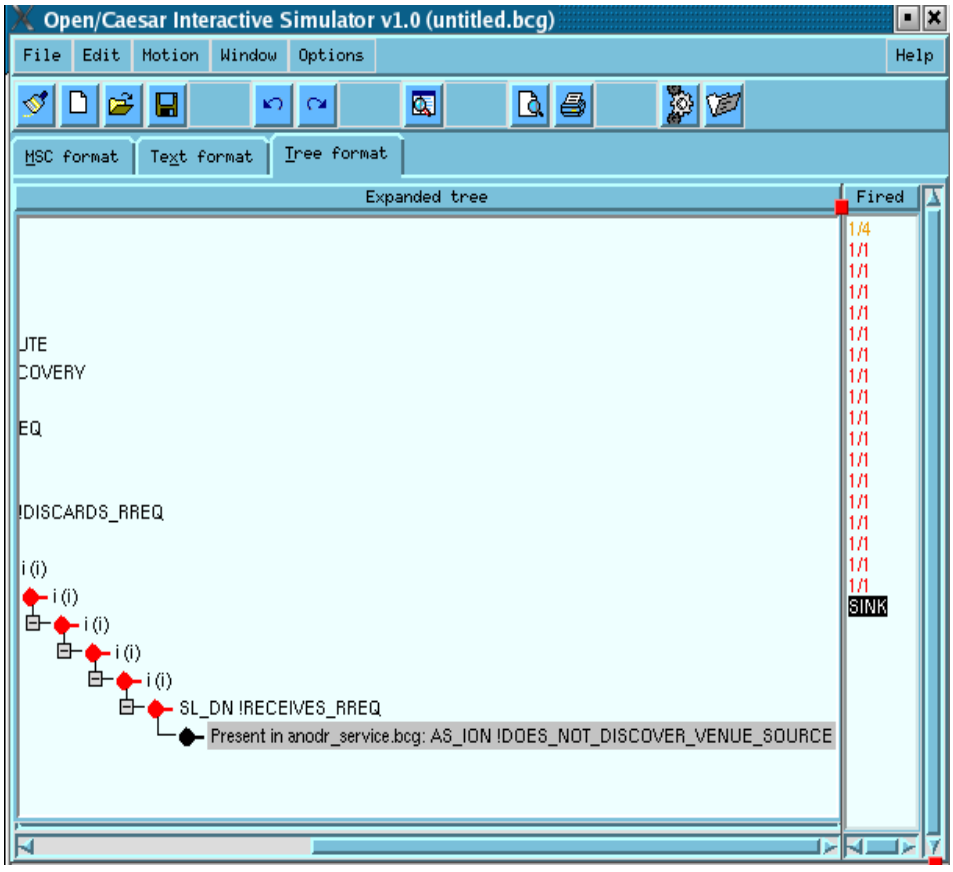

Figura 5.12: Simulação de Execução do Arquivo com a Indicação do Problema Encontrado na Verificação de Anonimato do *Venue* da Origem em Relação a *ION*

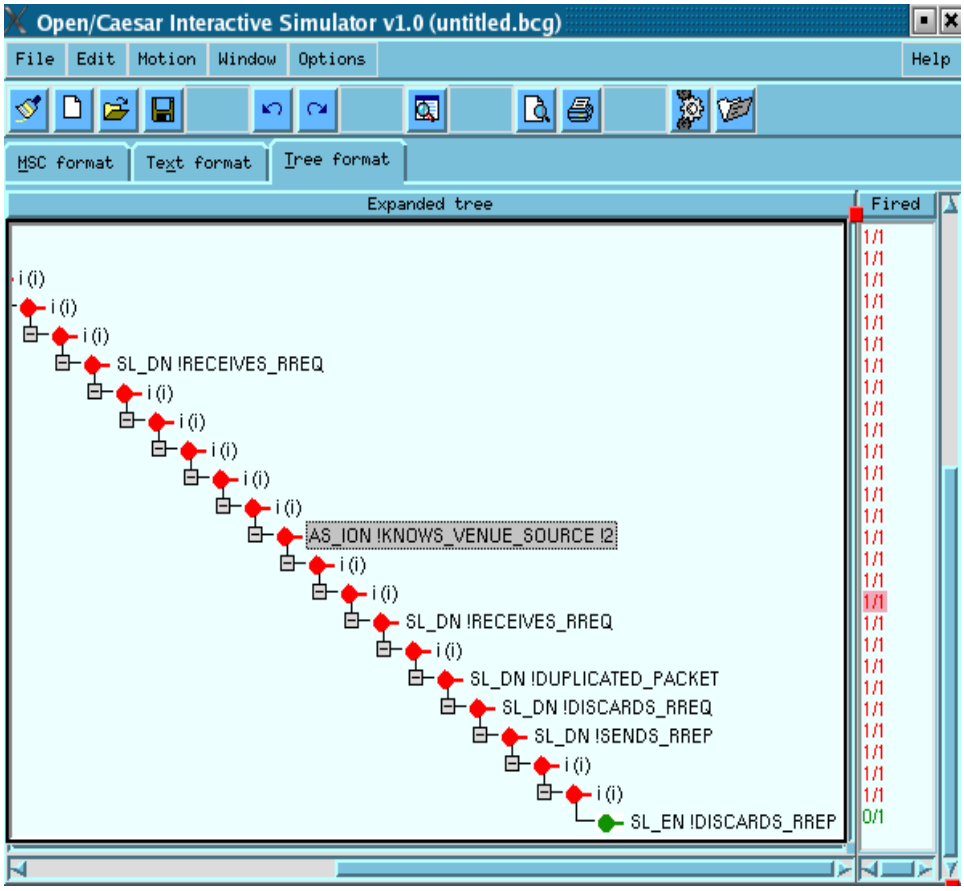

Figura 5.13: Simulação de Execução do LTS correspondente à Especificação do Protocolo

5.2 Algumas Limitações

Uma limitação, mencionada pelos próprios autores, é a sensibilidade à mobilidade dos nós. Estratégias tradicionais para manutenção de enlaces não podem ser utilizadas, porque, em ANODR, somente caminhos anônimos estabelecidos para a atual conexão podem ser usados e os nós não sabem as identidades dos nós origem e destino.

Os nós precisam compartilhar uma chave simétrica que é usada para gerar um *trapdoor* antes de se comunicarem. Porém, o protocolo não provê um mecanismo para estabelecer essas chaves.

Como discutido por SONG et al. (2005), quando um nó recebe um pacote de requisição de rota, precisa testar todas as chaves secretas que compartilha para tentar abrir o *trapdoor*; nem o próprio destino sabe diretamente qual é a chave correta. O mesmo problema ocorre com o pacote de resposta; quando um nó recebe esse pa-

cote, precisa testar todos os conjuntos de chaves estabelecidos para cada requisição recebida (três decifragens - uma assimétrica e duas simétricas) para descobrir se pertence à rota. Conseqüentemente, escalabilidade é um problema em ANODR.

Para manter os pacotes de requisição e de resposta com tamanho único e fixo, cada nó acrescenta *paddings* aos *onions* destes pacotes até completar 400 bits. Esse valor é calculado, considerando-se que a cada salto, o tamanho real do *onion* aumenta 40 bits e que 10 saltos é um tamanho de diâmetro razoável para as redes. Assim, é preciso estimar o diâmetro da rede para se escolher um valor e configurar todos os nós.

Anonimato compromete o desempenho do roteamento, uma vez que são necessárias mais operações criptográficas e o gerenciamento torna-se mais complexo. Para cada par de requisição e resposta, todo nó mantém um par de chaves pública e privada, uma chave simétrica e um *nonce*; a origem faz três cifragens simétricas, duas decifragens simétricas e uma decifragem assimétrica, no mínimo; um intermediário, três cifragens simétricas, quatro decifragens simétricas, uma cifragem assimétrica e uma decifragem assimétrica, no mínimo; o destino, uma cifragem simétrica, uma cifragem assimétrica e duas decifragens simétricas, no mínimo. Para cada rota estabelecida, todo nó na rota mantém duas seqüências de pseudônimos de rota dinâmicos e uma chave secreta; a origem faz uma cifragem simétrica; um intermediário, uma cifragem simétrica e uma decifragem simétrica; o destino, uma decifragem simétrica.

5.3 Possíveis Modificações

As seguintes modificações podem ser realizadas nos pacotes de requisição e de resposta do protocolo ANODR, a fim de garantir anonimato da identidade do destino e do *venue* da origem, evitar que nós pertencentes à rota, em conluio, saibam sua distância relativa e diminuir o número de operações criptográficas realizadas pelos nós para descobrir se um pacote de resposta é destinado a eles:

- Substituir a *tag dest*, que identifica o destino, presente no *trapdoor* da requisição ($[E_{K_{SD}}(dest, K_c), E_{K_c}(dest)]$) por um *nonce N*. Assim, quando a chave K_c for informada aos nós intermediários pertencentes à rota através da resposta, a identidade do destino não será revelada.

- Remover o *onion* da requisição e da resposta, para que esses pacotes possuam tamanhos únicos e fixos. No projeto de ANODR, essa estrutura é utilizada para que os nós verifiquem se o pacote é destinado a eles, porém seu tamanho muda de forma padrão, indicando a distância da origem aos nós internos. Na resposta, pode ser substituído pelo número de seqüência da requisição (*seq*), como ocorre no protocolo ASR.

5.4 Considerações

Este capítulo aprofundou a discussão sobre o protocolo ANODR, indicado como o mais seguro pela análise apresentada no capítulo 4. Suas limitações foram discutidas e pequenas modificações foram sugeridas para melhorar sua segurança em relação a anonimato da identidade do destino e do *venue* da origem. Demonstrou-se formalmente os pontos em que tais propriedades de anonimato não são asseguradas, e que o protocolo funciona corretamente. É importante destacar que é a primeira especificação formal de ANODR e que apesar de não abordar todas as propriedades de anonimato, há a modelagem de envio e recebimento de pacotes (operadores ? e !), do conhecimento do intruso (estrutura com funções para inserção de cada um dos campos dos pacotes; campos criptografados com chaves conhecidas são armazenados em claro), de passagem de parâmetros entre os processos e funções para manipulação dos campos de cada pacote. Os LTSs das especificações do protocolo e do serviço não são ilustrados, porque a quantidade de estados e transições dificulta a sua visualização. Na modelagem, os seguintes itens foram abstraídos: a verificação do número de seqüência e a verificação se o nó é o destinatário são modeladas como ações internas i; como os enlaces são simétricos, quando um nó vizinho ao nó origem encaminha um pacote enviado por ele, este deveria recebê-lo também, porém para evitar repetições, isto não foi modelado. Os trabalhos relacionados à formalização de anonimato mencionados no primeiro capítulo não foram utilizados, porque suas definições e modelagens são específicas para os sistemas verificados. Além disso, no caso dos protocolos de roteamento, a verificação de anonimato está relacionada à análise do conteúdo dos pacotes e não somente ao seu envio e recebimento, como é o caso desses trabalhos. O próximo capítulo apresenta as conclusões e os trabalhos futuros.

6 Conclusões e Trabalhos Futuros

Privacidade e anonimato são características necessárias em diversas aplicações de redes sem fio ad hoc móveis para inibirem os efeitos de ataques de captura e de análise de tráfego contra pacotes de roteamento. Porém, não são objetivos fáceis de serem alcançados, devido, principalmente, às próprias características dessas redes, como meio sem fio compartilhado e ausência de infra-estrutura fixa. Outras dificuldades são: garantir anonimato e desempenho e manter a rede escalável, pois é necessário utilizar técnicas *mixing*, que aumentam o tráfego e impõem atrasos, realizar um número maior de operações criptográficas, utilizadas, por exemplo, para manter em segredo a identidade das entidades comunicantes, e gerenciar um número maior de chaves.

A análise e a comparação dos protocolos de roteamento anônimo realizadas nesta dissertação propiciaram a verificação de diversas propriedades associadas a anonimato - identidade, localização, *venue*, padrão de movimento e rota - segundo uma definição padronizada, e apontaram o ANODR como um dos mais seguros em relação a essas propriedades. Além disso, o funcionamento e o formato dos pacotes de todas as propostas de roteamento anônimo encontradas com a mesma classificação e mais relevantes foram detalhados.

Verificou-se que nenhum dos protocolos garante todas as propriedades e que, em geral, possuem características em comum e utilizam os mesmos mecanismos para prover anonimato, diferindo apenas nos detalhes de como os empregam. Além disso, nem todos se preocupam com outras questões de segurança, como ataques de repetição e de modificação.

A definição de anonimato que serviu como base para a análise é resultado da organização das diversas definições utilizadas em redes estruturadas e apresentadas

nos artigos que propuseram os protocolos, e inclui as informações que devem ser mantidas em segredo mais relevantes. Para cada uma das propriedades definidas, foram identificadas as principais questões que devem ser analisadas para verificar se o protocolo a garante. Essas questões foram identificadas, após analisar quais seriam os meios explícitos e implícitos através das quais as informações a serem protegidas poderiam ser descobertas.

O protocolo ANODR foi especificado formalmente em LOTOS e verificado com o pacote de ferramentas CADP. Optou-se por LOTOS, porque é uma técnica consolidada e padronizada, desenvolvida pela ISO - *International Organization for Standardization*, e de acordo com LEDUC & GERMEAU (2000), é apropriada para especificar protocolos de segurança, devido à sua flexibilidade; além disso, um dos protocolos estudados neste trabalho, ANDSR, foi especificado em LOTOS. CADP constitui um dos pacotes de ferramentas mais completos e utilizados para verificação, teste e simulação de especificações em LOTOS. Pelas pesquisas realizadas, é a primeira especificação formal do protocolo ANODR e a segunda de protocolos de roteamento anônimo para redes sem fio ad hoc móveis.

Através da verificação formal, foi possível demonstrar formalmente os pontos e em relação a quais nós, as propriedades anonimato da identidade do destino e anonimato do *venue* da origem não são asseguradas, e que o protocolo funciona corretamente. Porém, verificou-se que a técnica de descrição formal LOTOS e a ferramenta CADP não são ideais para a verificação automática de anonimato. São ótimas para a verificação do comportamento e da interação entre os nós, porém a verificação de anonimato, principalmente, em um ambiente tão dinâmico e constituído por diferentes entidades como redes sem fio ad hoc móveis é uma questão complexa, pois envolve muitos detalhes. Há a necessidade de se modelar questões como mobilidade, comunicação de grupos de nós, fluxo de informações em diferentes níveis, verificação de todos os campos de todos os pacotes e análise estatística. É importante ressaltar que não basta analisar, por exemplo, se a informação a ser protegida está presente no pacote, mas também se há meios não explícitos de inferi-la.

As principais limitações do trabalho são: não considera ataques ativos de análise de tráfego; a análise não é probabilística; a especificação formal não aborda todas as propriedades de anonimato, pois não modela questões como mobilidade e técnicas *mixing*, necessárias à sua verificação; ataques como negação de serviço, personi-

ficação e modificação dos pacotes não são modelados.

Como trabalhos futuros, sugerem-se:

- Simular os protocolos em diferentes cenários (diferentes pares origem/destino, graus de mobilidade e número de intrusos, por exemplo) e verificar a estatística quanto a anonimato. A dificuldade é implementar todos os protocolos, pois nem todos são simulados, e os que são, utilizam simuladores diferentes.
- Desenvolver um protocolo que contemple as principais vantagens de cada uma das propostas analisadas.
- Modelar as características que não foram especificadas: mobilidade, técnicas *mixing* e diferentes pares origem/destino.
- Investigar a vulnerabilidade dos protocolos a ataques ativos de análise de tráfego e a ataques contra a sua própria segurança.
- Implementar as modificações sugeridas para o protocolo ANODR.
- Aplicar outras técnicas de descrição e verificação formal.

Referências

ARAUJO, A. M. de. *ANDSR - Protocolo de Roteamento Anônimo para Redes Ad Hoc*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2005.

BANERJEE, A.; JULIEN, C.; SHMATIKOV, V. *Certificate Free Anonymous Routing for Mobile Ad Hoc Networks*. [S.l.], 2006. Technical Report, TR-UTEDGE-2005-005.

BEYER, D. A. Accomplishments of the DARPA SURAN Program. In: *Military Communications Conference*. [S.l.: s.n.], 1990.

BHARGAVAN, K.; OBRADOVIC, D.; GUNTER, C. A. Formal Verification of Standards for Distance Vector Routing Protocols. *Journal of the ACM*, v. 49, n. 4, p. 538–576, July 2002.

BLUETOOTH SIG. *Bluetooth SIG Membership Website*. Disponível em: <http://www.bluetooth.org>. Acesso em: jul. 2006.

BOLOGNESI, T.; BRINKSMA, E. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, v. 14, n. 1, p. 25–59, January 1987.

BOUKERCHE, A. et al. A Novel Solution for Achieving Anonymity in Wireless Ad hoc Networks. In: *1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. [S.l.: s.n.], 2004. p. 30–38.

BOUKERCHE, A. et al. Anonymity Enabling Scheme for Wireless Ad hoc Networks. In: *IEEE Global Telecommunications Conference*. [S.l.: s.n.], 2004. p. 136–140.

BOUKERCHE, A. et al. SDAR: A Secure Distributed Anonymous Routing Protocol for Wireless and Mobile Ad Hoc Networks. In: *29th Annual IEEE International Conference on Local Computer Networks*. [S.l.: s.n.], 2004. p. 618–324.

BUTTYAN, L.; HUBAUX, J.-P. Report on Working Session on Security in Wireless Ad Hoc Networks. *Mobile Computing and Communications Review*, v. 7, n. 1, p. 74–94, January 2003.

CHAKERES, I.; PERKINS, C. *Dynamic MANET On-demand (DYMO) Routing*. July 2007. IETF Internet-Draft. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-10.txt>.

CHAUM, D. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, v. 24, n. 2, p. 84–88, 1981.

CHAUM, D. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, v. 1, n. 1, p. 65–75, 1988.

CHENG, Y.; AGRAWAL, D. P. Distributed Anonymous Secure Routing Protocol in Wireless Mobile Ad Hoc Networks. In: *OPNETWORK*. [S.l.: s.n.], 2005.

CHLAMTAC, I.; CONTI, M.; LIU, J. J.-N. Mobile Ad Hoc Networking: Imperatives and Challenges. *Ad Hoc Networks*, v. 1, p. 13–64, July 2003.

CLAUSEN, T.; DEARLOVE, C.; JACQUET, P. *The Optimized Link State Routing Protocol version 2*. July 2007. IETF Internet-Draft. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-manet-olsrv2-04.txt>.

CLAUSEN, T.; JACQUET, P. *Optimized Link State Routing Protocol (OLSR)*. IETF, October 2003. RFC 3626 (Experimental). (Request for Comments, 3626). Disponível em: <http://www.ietf.org/rfc/rfc3626.txt>.

COMMON CRITERIA. *ISO/IEC 15408: Common Criteria for Information Security Evaluation*. August 1999. Disponível em: <http://www.commoncriteriaportal.org/>.

CORSON, S.; MACKER, J. *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. IETF, January 1999. RFC 2501 (Informational). (Request for Comments, 2501). Disponível em: <http://www.ietf.org/rfc/rfc2501.txt>.

EL-KHATIB, K. et al. Secure Dynamic Distributed Algorithm for Ad Hoc Wireless Networks. In: *International Conference on Parallel Processing Workshops*. [S.l.: s.n.], 2003. p. 359–366.

ETSI. *BRAN - Broad Band Radio Access Networks Website*. Disponível em: <http://portal.etsi.org/bran>. Acesso em: jul. 2006.

FEENEY, L. M. *A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks*. [S.l.], October 1999. Technical Report, T1999:07. Disponível em: <http://www.sics.se/cna/publications/survey-tax.ps>.

FIFER, W. C.; BRUNO, F. J. The Low-Cost Packet Radio. In: *IEEE*. [S.l.: s.n.], 1987. v. 75, n. 1, p. 33–42.

HAAS, Z. J. et al. Guest Editorial Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, v. 17, n. 8, p. 1329–1332, August 1999.

HALPERN, J. Y.; NEILL, K. R. O. Anonymity and Information Hiding in Multiagent Systems. In: *16th IEEE Computer Security Foundations Workshop*. [S.l.: s.n.], 2003. p. 75–88.

HONG, X.; KONG, J.; GERLA, M. Mobility Changes Anonymity: New Passive Threats in Mobile Ad Hoc Networks. *Wireless Communications and Mobile Computing*, v. 6, n. 3, p. 281–293, May 2006.

HU, Y.-C.; PERRIG, A. A Survey of Secure Wireless Ad Hoc Routing. *IEEE Security and Privacy*, v. 2, n. 3, p. 28–39, May-June 2004.

IEEE 802.11 WG. *IEEE 802.11, The Working Group Setting the Standards for Wireless LANs Website*. Disponível em: <http://www.ieee802.org/11/>. Acesso em: jul. 2006.

IEEE 802.15 WG. *IEEE 802.15 Working Group for Wireless Personal Area Networks (WPANs) Website*. Disponível em: <http://www.ieee802.org/15/>. Acesso em: jul. 2006.

IS8807. *ISO/IEC: Information Processing Sytems - Open Systems Interconnection, LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behavior*. 1988.

JIANG, S.; VAIDYA, N. H.; ZHAO, W. A Dynamic Mix Method for Wireless Ad Hoc Networks. In: *Military Communications Conference*. [S.l.: s.n.], 2001. p. 873–877.

JIANG, S.; VAIDYA, N. H.; ZHAO, W. A Mix Route Algorithm for Mix-net in Wireless Mobile Ad Hoc Networks. In: *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*. [S.l.: s.n.], 2004. p. 406–415.

JOHNSON, D.; HU, Y.; MALTZ, D. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. February 2007. RFC 4728 (Experimental). Disponível em: <http://www.ietf.org/rfc/rfc4728.txt>.

JOHNSON, D. B.; MALTZ, D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In: IMIELINSKI, T.; KORTH, H. (Ed.). *Mobile Computing*. [S.l.]: Kluwer Academic Publishers, 1996, (The Kluwer International Series in Engineering and Computer Science, v. 353). cap. 5, p. 153–181.

JUBIN, J.; TORNOW, J. D. The DARPA Packet Radio Network Protocols. In: *IEEE*. [S.l.: s.n.], 1987. v. 75, n. 1, p. 21–32.

KAHN, R. E. et al. Advances in Packet Radio Technology. In: *IEEE*. [S.l.: s.n.], 1978. v. 66, n. 11, p. 1468–1496.

KARP, B.; KUNG, H. T. GPSR: Greedy Perimeter Stateless Routing for Wireless Network. In: *Annual International Conference on Mobile Computing and Networking*. [S.l.: s.n.], 2000.

KONG, J. *Anonymous and Untraceable Communications in Mobile Wireless Networks*. Tese (Doutorado) — University of California, July 2004.

KONG, J.; HONG, X. ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. In: *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. [S.l.: s.n.], 2003. p. 291–302.

KONG, J. et al. Mobility Changes Anonymity: Mobile Ad Hoc Networks Need Efficient Anonymous Routing. In: *10th IEEE Symposium on Computers and Communications*. [S.l.: s.n.], 2005. p. 57–62.

KUOSMANEN, P. Classification of Ad Hoc Routing Protocols. Finnish Defence Forces, Naval Academy, Finland. Unpublished. June 2002. Disponível em: <http://www.netlab.tkk.fi/opetus/s38030/k02/Papers/12-Petteri.pdf>.

LEDUC, G.; GERMEAU, F. Verification of security protocols using lotos-method and application. *Computer Communications*, v. 23, p. 1089–1103, 2000.

LEINER, B.; RUTH, R.; SASTRY, A. R. Goals and Challenges of the DARPA GloMo Program. *IEEE Personal Communications*, v. 3, n. 6, p. 34–43, December 1996.

LIU, J. et al. Performance Evaluation of Anonymous Routing Protocols in MANETs. In: *Wireless Communications and Networking Conference*. [S.l.: s.n.], 2006. v. 2, p. 646–651.

MAÑAS, J. A. *A Tutorial on ADT semantics for LOTOS users. Part I: Fundamental Concepts*. November 1988.

MANET WG. *Mobile Ad-hoc Networks (manet) Charter*. Disponível em: <http://www.ietf.org/html.charters/manet-charter.html>. Acesso em: jun. 2007.

MARSHALL, J. D. *An Analysis of the Secure Routing Protocol for Mobile Ad Hoc Networks Route Discovery: Using Intuitive Reasoning and Formal Verification to Identify Flaws*. Dissertação (Mestrado) — Florida State University, 2003.

MESHNETWORKS. *Meshnetworks.com Website*. Disponível em: <http://www.meshnetworks.com>. Acesso em: jul. 2006.

MICHIARDI, P.; MOLVA, R. Ad Hoc Networks Security. In: BASAGNI, S. et al. (Ed.). *Mobile Ad Hoc Networking*. [S.l.]: IEEE Press and Wiley-Interscience, 2004. cap. 12, p. 329–354.

MISHRA, A.; NADKARNI, K.; PATCHA, A. Intrusion Detection in Wireless Ad Hoc Networks. *IEEE Wireless Communications*, v. 11, n. 1, p. 48–60, Feburary 2004.

MISHRA, A.; NADKARNI, K. M. Security in Wireless Ad Hoc Networks. In: ILYAS, M. (Ed.). *The Handbook of Ad Hoc Wireless Networks*. [S.l.]: CRC Press, 2003, (The Electrical Engineering Handbook Series). cap. 30, p. 466–528.

MURTHY, C. S. R.; MANOJ, B. S. *Ad Hoc Wireless Networks. Architectures and Protocols*. [S.l.]: Prentice Hall Professional Technical Reference, 2004.

OGIER, R.; TEMPLIN, F.; LEWIS, M. *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*. IETF, February 2004. RFC 3684 (Experimental). (Request for Comments, 3684). Disponível em: <http://www.ietf.org/rfc/rfc3684.txt>.

PERKINS, C.; BELDING-ROYER, E.; DAS, S. *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF, July 2003. RFC 3561 (Experimental). (Request for Comments, 3561). Disponível em: <http://www.ietf.org/rfc/rfc3561.txt>.

PERKINS, C. E.; ROYER, E. M. Ad-hoc On-Demand Distance Vector Routing. In: *Second IEEE Workshop on Mobile Computing Systems and Applications*. [S.l.: s.n.], 1999. p. 90–100.

PFITZMANN, A.; HANSEN, M. *Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management - A Consolidated Proposal for Terminology*. May 2006. Version v0.28. Disponível em: <http://dud.inf.tu-dresden.de/Anon_Terminology.shtml>.

RAHMAN, S. M. M. et al. An Anonymous On-Demand Position-based Routing in Mobile Ad Hoc Networks. In: *International Symposium on Applications and the Internet*. [S.l.: s.n.], 2006. p. 300–306.

RAMANATHAN, R.; REDI, J. A Brief Overview of Ad Hoc Networks: Challenges and Directions. *IEEE Communications Magazine*, v. 40, n. 5, p. 20–22, May 2002.

RAYMOND, J.-F. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In: FEDERRATH, H. (Ed.). *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. [S.l.]: Springer-Verlag, LNCS 2009, 2000. p. 10–29.

REED, M. G.; SYVERSON, P. F.; GOLDSCHLAG, D. M. Proxies for Anonymous Routing. In: *12th Annual Computer Security Applications Conference*. [S.l.: s.n.], 1996. p. 95–104.

REED, M. G.; SYVERSON, P. F.; GOLDSCHLAG, D. M. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communication*, v. 16, n. 4, p. 482–494, May 1998.

REITER, M. K.; RUBIN, A. D. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, v. 1, n. 1, p. 66–92, 1998.

REITER, M. K.; RUBIN, A. D. Anonymous Web Transactions with Crowds. *Communications of the ACM*, v. 42, n. 2, p. 32–48, February 1999.

RENESSE, R. de; AGHVAMI, A. H. Formal Verification of Ad-Hoc Routing Protocols Using SPIN Model Checker. In: *12th IEEE Mediterranean Electrotechnical Conference*. [S.l.: s.n.], 2004. v. 3, p. 1177–1182.

ROYER, E. M.; TOH, C.-K. A Review of Current Routing Protocol for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, v. 6, n. 2, p. 46–55, April 1999.

SCHNEIDER, S.; SIDIROPOULOS, A. CSP and Anonymity. In: *ESORICS*. [S.l.: s.n.], 1996. p. 198–218.

SHMATIKOV, V. Probabilistic Model Checking of an Anonymity Sytem. *Journal of Computer Security*, v. 12(3/4), p. 355–377, 2004.

SONG, R.; KORBA, L. Review of Network-based Approaches for Privacy. In: *14th Annual Canadian Information Technology Security Symposium*. [S.l.: s.n.], 2002.

SONG, R.; KORBA, L.; YEE, G. AnonDSR: Efficient Anonymous Dynamic Source Routing for Mobile Ad-Hoc Networks. In: *ACM Workshop on Security Ad Hoc and Sensor Networks*. [S.l.: s.n.], 2005. p. 33–42.

SPANWORKS INC. *SPANworks Home*. Disponível em: <http://www.spanworks.com>. Acesso em: jul. 2006.

SY, D.; CHEN, R.; BAO, L. ODAR: On-Demand Anonymous Routing in Ad Hoc Networks. In: *Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems*. [S.l.: s.n.], 2006. p. 267–276.

VASY-INRIA. *CADP Home Page*. Disponível em: <http://www.inrialpes.fr/vasy/cadp.html>. Acesso em: jun. 2007.

WRIGHT, J. et al. *Formalizing Anonymity: A Review*. [S.l.], June 2005.

WU, X.; BHARGAVA, B. AO2P: Ad Hoc On-Demand Position-Based Private Routing Protocol. *IEEE Transactions on Mobile Computing*, v. 4, n. 4, p. 335–348, July/August 2005.

YANG, H. et al. Security in Mobile Ad Hoc Networks: Challenges and Solutions. *IEEE Wireless Communications*, v. 11, n. 1, p. 38–47, February 2004.

ZHANG, Y.; LIU, W.; LOU, W. Anonymous Communications in Mobile Ad Hoc Networks. In: *Annual Joint Conference of the IEEE Computer and Communications Societies*. [S.l.: s.n.], 2005. v. 3, p. 1940–1951.

ZHANG, Y. et al. Anonymous Handshakes in Mobile Ad Hoc Networks. In: *IEEE Military Communications Conference*. [S.l.: s.n.], 2004. v. 3, p. 1193–1999.

ZHU, B. et al. Anonymous Secure Routing in Mobile Ad-Hoc Networks. In: *29th Annual International Conference on Local Computer Networks*. [S.l.: s.n.], 2004. p. 102–108.

Apêndice A

Especificação da Biblioteca - ANODRLIB.lib

Neste apêndice, apresenta-se a especificação em ACT ONE da biblioteca ANODRLIB.lib. Esta biblioteca, desenvolvida nesta dissertação, contém a declaração de tipos e funções necessários para a especificação do protocolo ANODR e do serviço.

```
library Boolean, Natural endlib (* Predefined libraries *)

(* Actions and status messages *)
type ActionStatus is
sorts Action_Sort (*! implementedby Action comparedby CMP_Action printedby PRINT_Action *)
opns
    route_to_destination (*! implementedby route_to_destination constructor *),
    checks_table (*! implementedby checks_table constructor *),
    knows_route (*! implementedby knows_route constructor *),
    does_not_know_route (*! implementedby does_not_know_route constructor *),
    start_route_discovery (*! implementedby start_route_discovery constructor *),
    end_route_discovery (*! implementedby end_route_discovery constructor *),
    start_transmission (*! implementedby start_transmission constructor *),
    end_transmission (*! implementedby end_transmission constructor *),
    sends_rreq (*! implementedby sends_rreq constructor *),
    sends_rrep (*! implementedby sends_rrep constructor *),
    sends_data (*! implementedby sends_data constructor *),
    receives_rreq (*! implementedby received_rreq constructor *),
    receives_rrep (*! implementedby received_rrep constructor *),
    receives_data (*! implementedby received_data constructor *),
    receives_error (*! implementedby received_error constructor *),
    discards_rreq (*! implementedby discards_rreq constructor *),
    discards_rrep (*! implementedby discards_rrep constructor *),
    discards_data (*! implementedby discards_data constructor *),
    discards_error (*! implementedby discards_error constructor *),
    duplicated_packet (*! implementedby duplicated_packet constructor *),
    timeout (*! implementedby timeout constructor *),
    broken_link (*! implementedby broken_link constructor *): -> Action_Sort
endtype

(* Acknowledgment: message that is sent when a data packet is received *)
type Ack is
sorts Ack_Sort (*! implementedby Ack comparedby CMP_Ack printedby PRINT_Ack *)
opns local_ack (*! implementedby local_ack constructor *): -> Ack_Sort
endtype

(* Syncronization: time *)
```

```
type Syncronization is
sorts Syncronization_Sort (*! implementedby Syncronization comparedby CMP_Syncronization printedby
PRINT_Syncronization *)
opns synchronization (*! implementedby synchronization constructor *): -> Syncronization_Sort
endtype

(* Element: information that is inserted into the structure knowledge *)
type Element is Boolean, Identity, SequenceNumber, PublicKey, SecretKey, SymmetricEncryptedMessage,
AsymmetricEncryptedMessage, TBO, Pseudonym, Message
sorts Element_Sort (*! implementedby Element comparedby CMP_Element printedby PRINT_Element *)
opns
    e1 (*! implementedby e1 constructor *): Identity_Sort -> Element_Sort
    e2 (*! implementedby e2 constructor *): SequenceNumber_Sort -> Element_Sort
    e3 (*! implementedby e3 constructor *): PublicKey_Sort -> Element_Sort
    e4 (*! implementedby e4 constructor *): SecretKey_Sort -> Element_Sort
    e5 (*! implementedby e5 constructor *): SymmetricEncryptedMessage_Sort -> Element_Sort
    e6 (*! implementedby e6 constructor *): AsymmetricEncryptedMessage_Sort -> Element_Sort
    e7 (*! implementedby e7 constructor *): TBO_Sort -> Element_Sort
    e8 (*! implementedby e8 constructor *): Pseudonym_Sort -> Element_Sort
    e9 (*! implementedby e9 constructor *): Message_Sort -> Element_Sort
endtype

(* Node knowledge: there are functions to insert the packets fields into the structure knowledge and to verify
whether an information is into the knowledge *)
type Knowledge is RouteRequestPacket, RouteReplyPacket, DataPacket, Boolean, Element
sorts Knowledge_Sort (*! implementedby Knowledge comparedby CMP_Knowledge printedby PRINT_Knowledge*)
opns
    knowledge_IIN (*! implementedby knowledge_IIN constructor *),
    knowledge_ION (*! implementedby knowledge_ION constructor *),
    knowledge_EN (*! implementedby knowledge_EN constructor *) : -> Knowledge_Sort
    add (*! implementedby add constructor *): Element_Sort, Knowledge_Sort -> Knowledge_Sort
    insert_rreq (*! implementedby insert_rreq *): RouteRequestPacket_Sort, Knowledge_Sort -> Knowledge_Sort
    insert_rrep (*! implementedby insert_rrep *): RouteReplyPacket_Sort, Knowledge_Sort -> Knowledge_Sort
    insert_rrepIIN (*! implementedby insert_rrepIIN *): RouteReplyPacket_Sort, RouteRequestPacket_Sort,
    PrivateKey_Sort, Knowledge_Sort -> Knowledge_Sort
    insert_pckdata (*! implementedby insert_pckdata *): DataPacket_Sort, Knowledge_Sort ->
    Knowledge_Sort
    insert_dataIIN (*! implementedby insert_data *): DataPacket_Sort, SecretKey_Sort, Knowledge_Sort ->
    Knowledge_Sort
    _IsIn_ (*! implementedby IsIn *),
    _IsNotIn_ (*! implementedby IsNotIn *): Element_Sort, Knowledge_Sort -> Bool
eqns
    forall pckrreq: RouteRequestPacket_Sort,
           pckrrep: RouteReplyPacket_Sort,
           pckdata: DataPacket_Sort,
           privk: PrivateKey_Sort,
           kseed: SecretKey_Sort,
           s: Knowledge_Sort,
           x, y: Element_Sort
    ofsort Knowledge_Sort
           insert_rreq(pckrreq, s) = add(e2(get_seqnum(pckrreq)),add(e3(get_pubkey(pckrreq)),add(
           e5(get_trapdoorpart1(pckrreq)),add(e5(get_trapdoorpart2(pckrreq)),add(e7(get_tboreq(pckrreq)),s)))));
           insert_rrep(pckrrep, s) = add(e6(get_part1(pckrrep)),add(e5(get_part2(pckrrep)),s));
```

```
        insert_rrepIIN(pckrrep, pckrreq, privk, s) = add(e1(get_id(sym_decrypt(get_trapdoorpart2(
        pckrreq), get_commitmentkey(sym_decrypt(get_part2 (pckrrep), get_keyseed(asy_decrypt(
        get_part1(pckrrep),privk)))))),add(e4(get_commitmentkey(sym_decrypt(get_part2(pckrrep),get_keyseed(
        asy_decrypt(get_part1(pckrrep),privk))))),add(e4(get_keyseed(asy_decrypt(get_part1(pckrrep),
        privk))),s)));
        insert_pckdata(pckdata, s) = add(e8(get_pseudonym(pckdata)),add(e5(get_payload(pckdata)),s));
        insert_dataIIN(pckdata, kseed, s) = add(e8(get_pseudonym(pckdata)),add(e9(sym_decrypt(
        get_payload(pckdata),kseed)),s));
    ofsort Bool
        x IsIn Knowledge_IIN = false;
        x IsIn Knowledge_ION = false;
        x IsIn Knowledge_EN = false;
        x IsIn add(x, s) = true;
        x IsIn add(y, s) = x IsIn s;
        x IsNotIn s = not(x IsIn s);
endtype

(* Anonymity status *)
type AnonymityStatus is Knowledge, Boolean, Element
sorts AnonymityStatus_Sort (*! implementedby AnonymityStatus comparedby CMP_AnonymityStatus printedby
PRINT_AnonymityStatus *)
opns
    does_not_discover_IDsource (*! implementedby does_not_discover_IDsource constructor *),
    does_not_discover_IDdestination (*! implementedby does_not_discover_IDdestination constructor *),
    knows_IDsource (*! implementedby knows_IDsource constructor *),
    knows_IDdestination (*! implementedby knows_IDdestination constructor *),
    does_not_discover_venue_source (*! implementedby does_not_discover_venue_source constructor *),
    does_not_discover_venue_destination (*! implementedby does_not_discover_venue_destination constructor *),
    knows_venue_destination (*! implementedby knows_venue_destination constructor *),
    knows_venue_source (*! implementedby knows_venue_source constructor *): -> AnonymityStatus_Sort,
    verify_anonymityIDsrc (*! implementedby verify_anonymityIDsrc *),
    verify_anonymityIDdest (*! implementedby verify_anonymityIDdest *): Knowledge_Sort ->
    AnonymityStatus_Sort
eqns
    forall s: Knowledge_Sort,
        req: RouteRequestPacket_Sort,
        rep: RouteReplyPacket_Sort,
        data: DataPacket_Sort
    ofsort AnonymityStatus_Sort
        e1(ID_SN) IsIn s => verify_anonymityIDsrc(s) = knows_IDsource;
        e1(ID_SN) IsNotIn s => verify_anonymityIDsrc(s) = does_not_discover_IDsource;
        e1(ID_DN) IsIn s => verify_anonymityIDdest(s) = knows_IDdestination;
        e1(ID_DN) IsNotIn s => verify_anonymityIDdest(s) = does_not_discover_IDdestination;
endtype

(* Packet type *)
type PacketType is
sorts PacketType_Sort (*! implementedby PacketType comparedby CMP_PacketType printedby
PRINT_PacketType *)
opns
    RREQType (*! implementedby RREQType constructor *),
    RREPType (*! implementedby RREPType constructor *),
    DATAType (*! implementedby DATAType constructor *),
```

```
    ERRORType (*! implementedby ERRORType constructor *): -> PacketType_Sort
endtype

(* Sequence number *)
type SequenceNumber is
sorts SequenceNumber_Sort (*! implementedby SequenceNumber comparedby CMP_SequenceNumber printedby
PRINT_SequenceNumber *)
opns seqnum (*! implementedby seqnum constructor *): -> SequenceNumber_Sort
endtype

(* Nonce *)
type Nonce is
sorts Nonce_Sort (*! implementedby Nonce comparedby CMP_Nonce printedby PRINT_Nonce *)
opns
    N_IIN (*! implementedby N_IIN constructor *),
    N_ION (*! implementedby N_ION constructor *): -> Nonce_Sort
endtype

(* Identity of node *)
type Identity is
sorts Identity_Sort (*! implementedby Identity comparedby CMP_Identity printedby PRINT_Identity *)
opns
    ID_SN (*! implementedby ID_SN constructor *),
    ID_IIN (*! implementedby ID_IIN constructor *),
    ID_ION (*! implementedby ID_ION constructor *),
    ID_DN (*! implementedby ID_DN constructor *): -> Identity_Sort
endtype

(* Route pseudonym *)
type Pseudonym is
sorts Pseudonym_Sort (*! implementedby Pseudonym comparedby CMP_Pseudonym printedby
PRINT_Pseudonym *)
opns
    n1_SN_IIN (*! implementedby n1_SN_IIN constructor *),
    n2_SN_IIN (*! implementedby n2_SN_IIN constructor *),
    n1_IIN_DN (*! implementedby n1_IIN_DN constructor *),
    n2_IIN_DN (*! implementedby n2_IIN_DN constructor *): -> Pseudonym_Sort
endtype

(* Public key *)
type PublicKey is
sorts PublicKey_Sort (*! implementedby PublicKey comparedby CMP_PublicKey printedby PRINT_PublicKey *)
opns
    PK_SN_one (*! implementedby PK_SN_one constructor *),
    PK_IIN_one (*! implementedby PK_IIN_one constructor *),
    PK_ION_one (*! implementedby PK_ION_one constructor *): -> PublicKey_Sort
endtype

(* Private key *)
type PrivateKey is
sorts PrivateKey_Sort (*!implementedby PrivateKey comparedby CMP_PrivateKey printedby
PRINT_PrivateKey*)
opns
    SK_SN_one (*! implementedby SK_SN_one constructor *),
    SK_IIN_one (*! implementedby SK_IIN_one constructor *),
```

```
    SK_ION_one (*! implementedby SK_ION_one constructor *): -> PrivateKey_Sort
endtype

(* Secret key *)
type SecretKey is
sorts SecretKey_Sort (*! implementedby SecretKey comparedby CMP_SecretKey printedby PRINT_SecretKey *)
opns
    K_SD (*! implementedby K_SD constructor *),
    K_c (*! implementedby K_c constructor *),
    K_SN_one (*! implementedby K_SN_one constructor *),
    K_IIN_one (*! implementedby K_IIN_one constructor *),
    K_ION_one (*! implementedby K_ION_one constructor *),
    K_SN_IIN_seed (*! implementedby K_SN_IN_seed constructor *),
    K_IIN_DN_seed (*! implementedby K_IIN_DN_seed constructor *): -> SecretKey_Sort
endtype

(* Message *)
type Message is Identity, Nonce, SecretKey, TBO
sorts Message_Sort (*! implementedby Message comparedby CMP_Message printedby PRINT_Message *)
opns
    data1 (*! implementedby data1 constructor *): -> Message_Sort
    data2 (*! implementedby data2 constructor *): -> Message_Sort
    append1 (*! implementedby append1 constructor *): Identity_Sort, SecretKey_Sort -> Message_Sort
    append2 (*! implementedby append2 constructor *): Identity_Sort -> Message_Sort
    append3 (*! implementedby append3 constructor *): SecretKey_Sort -> Message_Sort
    append4 (*! implementedby append4 constructor *): SecretKey_Sort, TBO_Sort -> Message_Sort
    get_id (*! implementedby get_id *): Message_Sort -> Identity_Sort
    get_commitmentkey (*! implementedby get_commitmentkey *): Message_Sort -> SecretKey_Sort
    get_keyseed (*! implementedby get_keyseed *): Message_Sort -> SecretKey_Sort
    get_tbo (*! implementedby get_tbo *): Message_Sort -> TBO_Sort
eqns
    forall id: Identity_Sort,
           k: SecretKey_Sort,
           tbo: TBO_Sort
    ofsort Identity_Sort
           get_id(append1(id,k)) = id
           get_id(append2(id)) = id;
    ofsort SecretKey_Sort
           get_commitmentkey(append1(id,k)) = k
           get_commitmentkey(append4(k,tbo)) = k;
           get_keyseed(append3(k)) = k;
    ofsort TBO_Sort
           get_tbo(append4(k,tbo)) = tbo;
endtype

(* Onion *)
type TBO is Identity, Nonce, SecretKey, Natural
sorts TBO_Sort (*! implementedby TBO comparedby CMP_TBO printedby PRINT_TBO *)
opns
    TBO_SN (*! implementedby TBO_SN constructor *): Identity_Sort, SecretKey_Sort -> TBO_Sort
    TBO_IN (*! implementedby TBO_IN constructor *): Nonce_Sort, TBO_Sort, SecretKey_Sort -> TBO_Sort
    peelsoff (*! implementedby peelsoff *): TBO_Sort -> TBO_Sort
    hopcounter(*! implementedby hopcounter *): TBO_Sort -> Nat
```

```
eqns
    forall n: Nonce_Sort,
           k: SecretKey_Sort,
           tbo:TBO_Sort,
           id: Identity_Sort
    ofsort TBO_Sort
           peelsoff(TBO_IN(n, tbo, k)) = tbo;
    ofsort Nat
           hopcounter(TBO_SN(id,k)) = 1;
           hopcounter(TBO_IN(n, tbo, k)) = 1 + hopcounter(tbo);
endtype

(* Public key encryption *)
type AsymmetricEncryptedMessage is Message, PublicKey, PrivateKey
sorts AsymmetricEncryptedMessage_Sort (*! implementedby AsymmetricEncryptedMessage comparedby
CMP_AsymmetricEncryptedMessage printedby PRINT_AsymmetricEncryptedMessage *)
opns
    asy_encrypt (*! implementedby asy_encrypt constructor *): Message_Sort, PublicKey_Sort ->
    AsymmetricEncryptedMessage_Sort
    asy_decrypt (*! implementedby asy_decrypt *): AsymmetricEncryptedMessage_Sort, PrivateKey_Sort ->
    Message_Sort
eqns
    forall msg: Message_Sort, pubkey: PublicKey_Sort, prikey: PrivateKey_Sort
    ofsort Message_Sort
           asy_decrypt(asy_encrypt(msg,pubkey),prikey) = msg
endtype

(* Secret key encryption *)
type SymmetricEncryptedMessage is Message, SecretKey
sorts SymmetricEncryptedMessage_Sort (*! implementedby SymmetricEncryptedMessage comparedby
CMP_SymmetricEncryptedMessage printedby PRINT_SymmetricEncryptedMessage *)
opns
    sym_encrypt (*! implementedby sym_encrypt constructor *): Message_Sort, SecretKey_Sort ->
    SymmetricEncryptedMessage_Sort
    sym_decrypt (*! implementedby sym_decrypt *): SymmetricEncryptedMessage_Sort, SecretKey_Sort ->
    Message_Sort
eqns
    forall msg: Message_Sort, seckey: SecretKey_Sort
    ofsort Message_Sort
           sym_decrypt(sym_encrypt(msg,seckey),seckey) = msg
endtype

(* Route Discovery Request Packet *)
type RouteRequestPacket is PacketType, SequenceNumber, PublicKey, SymmetricEncryptedMessage
sorts RouteRequestPacket_Sort (*! implementedby RouteRequestPacket comparedby CMP_RouteRequestPacket
printedby PRINT_RouteRequestPacket *)
opns
    pck_RREQ (*! implementedby pck_RREQ constructor *): PacketType_Sort, SequenceNumber_Sort,
    PublicKey_Sort, SymmetricEncryptedMessage_Sort, SymmetricEncryptedMessage_Sort, TBO_Sort ->
    RouteRequestPacket_Sort
    get_seqnum (*! implementedby get_seqnum *): RouteRequestPacket_Sort -> SequenceNumber_Sort
    get_pubkey (*! implementedby get_pubkey *): RouteRequestPacket_Sort -> PublicKey_Sort
    get_tbo (*! implementedby get_tbo *): RouteRequestPacket_Sort -> TBO_Sort
```

```
    get_trapdoorpart1 (*! implementedby get_trapdoorpart1 *): RouteRequestPacket_Sort ->
    SymmetricEncryptedMessage_Sort
    get_trapdoorpart2 (*! implementedby get_trapdoorpart2 *): RouteRequestPacket_Sort ->
    SymmetricEncryptedMessage_Sort
eqns
    forall seq: SequenceNumber_Sort,
           pubkey: PublicKey_Sort,
           trappart1: SymmetricEncryptedMessage_Sort,
           trappart2: SymmetricEncryptedMessage_Sort,
           tbo: TBO_Sort
    ofsort SequenceNumber_Sort
           get_seqnum(pck_RREQ(RREQType,seq,pubkey,trappart1,trappart2,tbo)) = seq;
    ofsort PublicKey_Sort
           get_pubkey(pck_RREQ(RREQType,seq,pubkey,trappart1,trappart2,tbo)) = pubkey;
    ofsort SymmetricEncryptedMessage_Sort
           get_trapdoorpart1(pck_RREQ(RREQType,seq,pubkey,trappart1,trappart2,tbo)) = trappart1;
           get_trapdoorpart2(pck_RREQ(RREQType,seq,pubkey,trappart1,trappart2,tbo)) = trappart2;
    ofsort TBO_Sort
           get_TBO(pck_RREQ(RREQType,seq,pubkey,trappart1,trappart2,tbo)) = tbo;
endtype

(* Route Discovery Reply Packet *)
type RouteReplyPacket is AsymmetricEncryptedMessage, PacketType, SymmetricEncryptedMessage
sorts RouteReplyPacket_Sort (*! implementedby RouteReplyPacket comparedby CMP_RouteReplyPacket
printedby PRINT_RouteReplyPacket *)
opns pck_RREP (*! implementedby pck_RREP constructor *): PacketType_Sort,
AsymmetricEncryptedMessage_Sort, SymmetricEncryptedMessage_Sort -> RouteReplyPacket_Sort
endtype

(* Data Packet *)
type DataPacket is PacketType, Pseudonym, SymmetricEncryptedMessage
sorts DataPacket_Sort (*! implementedby DataPacket comparedby CMP_DataPacket printedby
PRINT_DataPacket *)
opns
    pck_DATA (*! implementedby pck_DATA constructor *): PacketType_Sort, Pseudonym_Sort,
    SymmetricEncryptedMessage_Sort -> DataPacket_Sort
    get_pseudonym (*! implementedby get_pseydonym *): DataPacket_Sort -> Pseudonym_Sort
    get_payload (*! implementedby get_payload *): DataPacket_Sort -> SymmetricEncryptedMessage_Sort
eqns
    forall n: Pseudonym_Sort, payload: SymmetricEncryptedMessage_Sort
    ofsort Pseudonym_Sort
           get_pseudonym(pck_DATA(DATAType,n,payload)) = n
    ofsort SymmetricEncryptedMessage_Sort
           get_payload(pck_DATA(DATAType,n,payload)) = payload
endtype

(* Route Maintenance Packet *)
type ErrorPacket is PacketType, Pseudonym
sorts ErrorPacket_Sort (*! implementedby ErrorPacket comparedby CMP_ErrorPacket printedby
PRINT_ErrorPacket *)
opns pck_ERROR (*! implementedby pck_ERROR constructor *): PacketType_Sort, Pseudonym_Sort ->
ErrorPacket_Sort
endtype
```

Apêndice B

Especificação do Protocolo ANODR - anodr_spe.lotos

Neste apêndice, apresenta-se a especificação em LOTOS do protocolo ANODR.

```
specification anodr_spe[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN, rt_DN]: noexit

library ANODRLIB endlib

behaviour

    CommunicationEstablishmentProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION,
    sl_EN, as_EN, sl_DN, rt_DN]

    where

    process CommunicationEstablishmentProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION,
    as_ION, sl_EN, as_EN, sl_DN, rt_DN]: noexit :=
    (* Source node knows the route to the destination and the data forwarding process is finished successfully *)
    (
        RouteDiscoveryProcess1[sl_SN, rt_SN]
        >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
            Knowledge_Sort
            in
            DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
            as_EN, sl_DN, rt_DN](k1, k2, IINKnowledge, IONKnowledge, ENKnowledge)
    )
    []
    (* Source node knows the route to the destination and the data forwarding process is initiated, but there is a
    broken link detection *)
    (
        RouteDiscoveryProcess1[sl_SN, rt_SN]
        >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
            Knowledge_Sort
            in
            MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN](k1, k2,
            IINKnowledge, IONKnowledge, ENKnowledge)
    )
    []
    (* Source node does not know the route to the destination and the data forwarding process is finished
    successfully *)
    (
        RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
        >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
            Knowledge_Sort
            in
            DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
            as_EN, sl_DN, rt_DN](k1, k2, IINKnowledge, IONKnowledge, ENKnowledge)
    )
    []
    (* Source node does not know the route to the destination and the data forwarding process is initiated, but
    there is a broken link detection *)
    (
        RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
```

```
        >> accept k1, k2: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
            Knowledge_Sort
            in
            MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN](k1, k2,
            IINKnowledge, IONKnowledge, ENKnowledge)
    )
    >> CommunicationEstablishmentProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION,
    as_ION, sl_EN, as_EN, sl_DN, rt_DN]

where

(* The source knows the route to the destination *)
process RouteDiscoveryProcess1[sl_SN, rt_SN]: exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort,
Knowledge_Sort, Knowledge_Sort) :=
    sl_SN ! route_to_destination; (* Superior layer requests a route *)
    rt_SN ! checks_table; (* Routing table is verified *)
    rt_SN ! knows_route; (* Result: route is known *)
    sl_SN ! knows_route; (* Network layer is ready to transmit data *)
    exit(K_SN_IIN_seed, K_IIN_DN_seed, insert_rrepIIN(pck_RREP(RREPType,
    asy_encrypt(append3(K_IIN_DN_seed), PK_IIN_one), sym_encrypt(append4(K_c,
    TBO_IN(N_IIN, TBO_SN(ID_SN, K_SN_one), K_IIN_one)), K_IIN_DN_seed)),
    pck_RREQ(RREQType, seqnum, PK_SN_one, sym_encrypt(append1(ID_DN, K_c), K_SD),
    sym_encrypt(append2(ID_DN), K_c), TBO_SN(ID_SN, K_SN_one)), SK_IIN_one,
    insert_rreq(pck_RREQ(RREQType, seqnum, PK_SN_one, sym_encrypt(append1(ID_DN,
    K_c), K_SD), sym_encrypt(append2(ID_DN), K_c), TBO_SN(ID_SN, K_SN_one)),
    Knowledge_IIN)), insert_rrep(pck_RREP(RREPType, asy_encrypt(append3(K_IIN_DN_seed),
    PK_IIN_one), sym_encrypt(append4(K_c, TBO_IN(N_IIN, TBO_SN(ID_SN, K_SN_one),
    K_IIN_one)), K_IIN_DN_seed)), insert_rreq(pck_RREQ(RREQType, seqnum, PK_IIN_one,
    sym_encrypt(append1(ID_DN, K_c), K_SD), sym_encrypt(append2(ID_DN), K_c),
    TBO_IN(N_IIN, TBO_SN(ID_SN, K_SN_one), K_IIN_one)), Knowledge_ION)),
    insert_rrep(pck_RREP(RREPType, asy_encrypt(append3(K_IIN_DN_seed), PK_IIN_one),
    sym_encrypt(append4(K_c, TBO_IN(N_IIN, TBO_SN(ID_SN, K_SN_one), K_IIN_one)),
    K_IIN_DN_seed)), insert_rreq(pck_RREQ(RREQType, seqnum, PK_SN_one,
    sym_encrypt(append1(ID_DN, K_c), K_SD), sym_encrypt(append2(ID_DN), K_c),
    TBO_SN(ID_SN, K_SN_one)), Knowledge_EN)))
endproc

(* The source does not know the route to the destination *)
process RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN, rt_DN]:
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
hide in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN in
(
    Source[sl_SN, rt_SN, in_SN, out_SN](ID_SN, ID_DN, K_SD)
    |||
    IntermediateInRoute[sl_IIN, rt_IIN, as_IIN, in_IIN, out_IIN](knowledge_IIN)
    |||
    IntermediateOutRoute[sl_ION, as_ION, in_ION, out_ION](knowledge_ION)
    |||
    External[sl_EN, as_EN, in_EN, out_EN](knowledge_EN)
    |||
    Destination[sl_DN, rt_DN, in_DN, out_DN](K_SD)
)
|[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]|
Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]

where

process Source[sl_SN, rt_SN, in_SN, out_SN](ID_SN, ID_DN: Identity_Sort, K_SD: SecretKey_Sort):
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    sl_SN ! route_to_destination; (* Superior layer requests a route *)
    rt_SN ! checks_table; (* Routing table is verified *)
    rt_SN ! does_not_know_route; (* Result: route is not known *)
    sl_SN ! does_not_know_route; (* Network layer is not ready to transmit data *)
    (* Source initiates the route discovery process *)
    sl_SN ! start_route_discovery;
    i; (* It generates the request packet *)
    sl_SN ! sends_rreq;
    out_SN ! pck_RREQ(RREQType, seqnum, PK_SN_one, sym_encrypt(append1(ID_DN, K_c),
    K_SD), sym_encrypt(append2(ID_DN), K_c), TBO_SN(ID_SN, K_SN_one));
    (* Source receives a route reply packet *)
    in_SN ? rep:RouteReplyPacket_Sort;
```

```
    sl_SN ! receives_rrep;
    i; (* It verifies whether is the intended receiver *)
    sl_SN ! end_route_discovery; (* Route is established *)
    exit(any SecretKey_Sort, any SecretKey_Sort, any Knowledge_Sort, any Knowledge_Sort,
    any Knowledge_Sort)
endproc

process IntermediateInRoute[sl_IIN, rt_IIN, as_IIN, in_IIN, out_IIN](IINKnowledge: Knowledge_Sort):
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    (* Node in the route receives a route request packet *)
    in_IIN ? sync:Syncronization_Sort;
    in_IIN ? req:RouteRequestPacket_Sort;
    i; (* It checks whether it has received the packet and whether it is the destination, records
    information, generates others and modifies the request *)
    (* Anonymity status *)
    as_IIN ! verify_anonymityIDsrc(insert_rreq(req, IINKnowledge));
    as_IIN ! verify_anonymityIDdest(insert_rreq(req, IINKnowledge));
    as_IIN ! knows_venue_source ! hopcounter(get_tboreq(req));
    (* Node in the route forwards the route request packet *)
    out_IIN ! pck_RREQ(RREQType, get_seqnum(req), PK_IIN_one, get_trapdoorpart1(req),
    get_trapdoorpart2(req), TBO_IN(N_IIN, get_tboreq(req),K_IIN_one));
    (* Node in the route receives a route reply packet *)
    in_IIN ? sync:Syncronization_Sort;
    in_IIN ? rep:RouteReplyPacket_Sort;
    i; (* It verifies whether it is the intended receiver, records the route, generates a new key seed
    and modifies the reply *)
    (* Anonymity status *)
    as_IIN ! verify_anonymityIDsrc(insert_rrepIIN(rep, req, SK_IIN_one, insert_rreq(req,
    IINKnowledge)));
    as_IIN ! verify_anonymityIDdest(insert_rrepIIN(rep, req, SK_IIN_one, insert_rreq(req,
    IINKnowledge)));
    as_IIN ! does_not_discover_venue_destination;
    (* Node in the route forwards the route reply packet *)
    out_IIN ! pck_RREP(RREPType, asy_encrypt(append3(K_SN_IIN_seed),get_pubkey(req)),
    sym_encrypt(append4(get_commitmentkey(sym_decrypt(get_part2(rep), get_keyseed(
    asy_decrypt(get_part1(rep),SK_IIN_one)))),peelsoff(get_tbo(sym_decrypt(get_part2(rep),
    get_keyseed(asy_decrypt(get_part1(rep),SK_IIN_one)))))), K_SN_IIN_seed));
    exit(K_SN_IIN_seed, any SecretKey_Sort, insert_rrepIIN(rep, req, SK_IIN_one, insert_rreq(req,
    IINKnowledge)), any Knowledge_Sort, any Knowledge_Sort)
endproc

process IntermediateOutRoute[sl_ION, as_ION, in_ION, out_ION](IONKnowledge: Knowledge_Sort):
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    (* Node out of route receives a route request packet *)
    in_ION ? sync:Syncronization_Sort;
    in_ION ? req:RouteRequestPacket_Sort;
    i; (* It checks whether it has received the packet and whether it is the destination, records
    some information, generates others and modifies the request *)
    (* Anonymity status *)
    as_ION ! verify_anonymityIDsrc(insert_rreq(req, IONKnowledge));
    as_ION ! verify_anonymityIDdest(insert_rreq(req, IONKnowledge));
    as_ION ! knows_venue_source ! hopcounter(get_tboreq(req));
    (* Node out of route forwards the route request packet *)
    out_ION ! pck_RREQ(RREQType, get_seqnum(req), PK_ION_one, get_trapdoorpart1(req),
    get_trapdoorpart2(req), TBO_IN(N_ION, get_tboreq(req),K_IIN_one));
    (* Node out of route receives a route reply packet *)
    in_ION ? sync:Syncronization_Sort;
    in_ION ? rep:RouteReplyPacket_Sort;
    i; (* It verifies whether it is the intended receiver *)
    (* Anonymity status *)
    as_ION ! verify_anonymityIDsrc(insert_rrep(rep, insert_rreq(req, IONKnowledge)));
    as_ION ! verify_anonymityIDdest(insert_rrep(rep, insert_rreq(req, IONKnowledge)));
    as_ION ! does_not_discover_venue_destination;
    sl_ION ! discards_rrep; (* Node out of the route discards the reply packet *)
    out_ION ! synchronization;
    exit(any SecretKey_Sort, any SecretKey_Sort, any Knowledge_Sort, insert_rrep(rep,
    insert_rreq(req, IONKnowledge)), any Knowledge_Sort)
endproc

process External[sl_EN, as_EN, in_EN, out_EN](ENKnowledge: Knowledge_Sort): exit(SecretKey_Sort,
SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
```

```
    (* External node receives a route request packet *)
    in_EN ? req:RouteRequestPacket_Sort;
    (* Anonymity status *)
    as_EN ! verify_anonymityIDsrc(insert_rreq(req, ENKnowledge));
    as_EN ! verify_anonymityIDdest(insert_rreq(req, ENKnowledge));
    as_EN ! does_not_discover_venue_source;
    sl_EN ! discards_rreq; (* External node discards the request packet *)
    out_EN ! synchronization;
    (* External node receives a route reply packet *)
    in_EN ? rep:RouteReplyPacket_Sort;
    (* Anonymity status *)
    as_EN ! verify_anonymityIDsrc(insert_rrep(rep, insert_rreq(req, ENKnowledge)));
    as_EN ! verify_anonymityIDdest(insert_rrep(rep, insert_rreq(req, ENKnowledge)));
    as_EN ! does_not_discover_venue_destination;
    sl_EN ! discards_rrep; (* External node discards the reply packet *)
    out_EN ! synchronization;
    exit(any SecretKey_Sort, any SecretKey_Sort, any Knowledge_Sort, any Knowledge_Sort, insert_rrep(rep,
insert_rreq(req, ENKnowledge)))
endproc

process Destination[sl_DN, rt_DN, in_DN, out_DN](K_SD: SecretKey_Sort): exit(SecretKey_Sort, SecretKey_Sort,
Knowledge_Sort, Knowledge_Sort, Knowledge_Sort):=
    (* Destination receives a route request packet *)
    in_DN ? req1:RouteRequestPacket_Sort;
    sl_DN ! receives_rreq;
    i; (* It checks seqnum and whether is the destination, generates the reply packet *)
    out_DN ! synchronization;
    (* Destination receives another route request packet - case of duplicated packet *)
    in_DN ? req2:RouteRequestPacket_Sort;
    sl_DN ! receives_rreq;
    i; (* It checks seqnum *)
    sl_DN ! duplicated_packet;
    sl_DN ! discards_rreq; (* Destination node discards the duplicated packet *)
    (* Destination sends a route reply packet *)
    sl_DN ! sends_rrep;
    out_DN ! pck_RREP(RREPType, asy_encrypt(append3(K_IIN_DN_seed), get_pubkey(req1)),
    sym_encrypt(append4(get_commitmentkey(sym_decrypt(get_trapdoorpart1(req1),K_SD)),
    get_tboreq(req1)), K_IIN_DN_seed));
    exit(any SecretKey_Sort, K_IIN_DN_seed, any Knowledge_Sort, any Knowledge_Sort,
    any Knowledge_Sort)
endproc

process Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]:
exit(SecretKey_Sort, SecretKey_Sort, Knowledge_Sort, Knowledge_Sort, Knowledge_Sort) :=
    out_SN ? req:RouteRequestPacket_Sort;
    in_EN ! req;
    out_EN ? sync:Syncronization_Sort;
    in_IIN ! sync;
    in_IIN ! req;
    out_IIN ? req:RouteRequestPacket_Sort;
    in_DN ! req;
    out_DN ? sync:Syncronization_Sort;
    in_ION ! sync;
    in_ION ! req;
    out_ION ? req:RouteRequestPacket_Sort;
    in_DN ! req;
    out_DN ? rep:RouteReplyPacket_Sort;
    in_EN ! rep;
    out_EN ? sync:Syncronization_Sort;
    in_ION ! sync;
    in_ION ! rep;
    out_ION ? sync:Syncronization_Sort;
    in_IIN ! sync;
    in_IIN ! rep;
    out_IIN ? rep:RouteReplyPacket_Sort;
    in_SN ! rep;
    exit(any SecretKey_Sort, any SecretKey_Sort, any Knowledge_Sort, any Knowledge_Sort,
    any Knowledge_Sort)
endproc endproc

(* Data forwarding process is finished successfully *)
```

```
process DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN,
rt_DN](kseed_SN_IIN, kseed_IIN_DN: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge:
Knowledge_Sort): exit :=
hide in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN in
(
    Source[sl_SN, rt_SN, in_SN, out_SN](kseed_SN_IIN)
    |||
    IntermediateInRoute[sl_IIN, rt_IIN, as_IIN, in_IIN, out_IIN](kseed_SN_IIN, kseed_IIN_DN,
    IINKnowledge)
    |||
    IntermediateOutRoute[sl_ION, as_ION, in_ION, out_ION](IONKnowledge)
    |||
    External[sl_EN, as_EN, in_EN, out_EN](ENKnowledge)
    |||
    Destination[sl_DN, rt_DN, in_DN, out_DN](kseed_IIN_DN)
)
|[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]|
Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]

where

process Source[sl_SN, rt_SN, in_SN, out_SN](kseed: SecretKey_Sort): exit :=
    (* Data transmission is initiated *)
    sl_SN ! start_transmission;
    i; (* It generates a data packet *)
    (* Source sends the data packet *)
    sl_SN ! sends_data;
    out_SN ! pck_DATA(DATAType, n1_SN_IIN, sym_encrypt(data1, kseed));
    in_SN ? ack:Ack_Sort; (* It receives local acknowledgment of the packet *)
    out_SN ! synchronization;
    exit
endproc

process IntermediateInRoute[sl_IIN, rt_IIN, as_IIN, in_IIN, out_IIN](kseed_prev, kseed_next: SecretKey_Sort,
IINKnowledge: Knowledge_Sort): exit :=
    (* Node in the route receives a data packet *)
    in_IIN ? sync:Syncronization_Sort;
    in_IIN ? data:DATAPacket_Sort;
    i; (* It verifies whether it is the intended receiver, modifies the packet *)
    (* Anonymity status *)
    as_IIN ! verify_anonymityIDsrc(insert_dataIIN(data, kseed_prev, IINKnowledge));
    as_IIN ! knows_IDdestination;
    as_IIN ! knows_venue_source;
    as_IIN ! does_not_discover_venue_destination;
    out_IIN ! local_ack; (* It sends local acknowledgment of the packet *)
    (* Node in the route forwards the data packet *)
    in_IIN ? sync:Syncronization_Sort;
    out_IIN ! pck_DATA(DATAType, n1_IIN_DN, sym_encrypt(sym_decrypt(get_payload(data),
    kseed_prev), kseed_next));
    in_IIN ? ack:Ack_Sort; (* It receives local acknowledgment of the packet *)
    exit
endproc

process IntermediateOutRoute[sl_ION, as_ION, in_ION, out_ION](IONKnowledge: Knowledge_Sort): exit :=
    (* Node out of route receives a data packet *)
    in_ION ? sync:Syncronization_Sort;
    in_ION ? data:DATAPacket_Sort;
    i; (* it verifies whether it is the intended receiver *)
    (* Anonymity status *)
    as_ION ! verify_anonymityIDsrc(insert_pckdata(data, IONKnowledge));
    as_ION ! verify_anonymityIDdest(insert_pckdata(data, IONKnowledge));
    as_ION ! does_not_discover_venue_source;
    as_ION ! does_not_discover_venue_destination;
    sl_ION ! discards_data; (* Node out of the route discards the reply packet *)
    out_ION ! synchronization;
    in_ION ? sync:Syncronization_Sort;
    in_ION ? ack:Ack_Sort;
    out_ION ! synchronization;
    exit
endproc
```

```
process External[sl_EN, as_EN, in_EN, out_EN](ENKnowledge: Knowledge_Sort): exit :=
    (* External node receives a data packet *)
    in_EN ? data:DATAPacket_Sort;
    (* Anonymity status *)
    as_EN ! verify_anonymityIDsrc(insert_pckdata(data, ENKnowledge));
    as_EN ! verify_anonymityIDdest(insert_pckdata(data, ENKnowledge));
    as_EN ! does_not_discover_venue_source;
    as_EN ! does_not_discover_venue_destination;
    sl_EN ! discards_data; (* External node discards the data packet *)
    out_EN ! synchronization;
    in_EN ? sync:Syncronization_Sort;
    in_EN ? ack:Ack_Sort;
    out_EN ! synchronization;
    exit
endproc

process Destination[sl_DN, rt_DN, in_DN, out_DN](kseed: SecretKey_Sort): exit:=
    (* Destination receives a data packet *)
    in_DN ? data:DATAPacket_Sort;
    i; (* It verifies whether it is the destination *)
    sl_DN ! receives_data;
    out_DN ! local_ack; (* Destination sends local acknowledgment of the packet *)
    sl_DN ! end_transmission; (* It has received all the packets *)
    exit
endproc

process Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]: exit :=
    out_SN ? data:DATAPacket_Sort;
    in_EN ! data;
    out_EN ? sync:Syncronization_Sort;
    in_ION ! sync;
    in_ION ! data;
    out_ION ? sync:Syncronization_Sort;
    in_IIN ! sync;
    in_IIN ! data;
    out_IIN ? ack:Ack_Sort;
    in_SN ! ack;
    out_SN ? sync:Syncronization_Sort;
    in_EN ! sync;
    in_EN ! ack;
    out_EN ? sync:Syncronization_Sort;
    in_ION ! sync;
    in_ION ! ack;
    out_ION ? sync:Syncronization_Sort;
    in_IIN ! sync;
    out_IIN ? data:DATAPacket_Sort;
    in_DN ! data;
    out_DN ? ack:Ack_Sort;
    in_IIN ! ack;
    exit
endproc endproc

(* Data forwarding process is initiated, but there is a broken link detection *)
process MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN](kseed_SN_IIN,
kseed_IIN_DN: SecretKey_Sort, IINKnowledge, IONKnowledge, ENKnowledge: Knowledge_Sort): exit :=
hide in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN in
(
    Source[sl_SN, rt_SN, in_SN, out_SN](kseed_SN_IIN)
    |||
    IntermediateInRoute[sl_IIN, rt_IIN, in_IIN, out_IIN](kseed_SN_IIN, kseed_IIN_DN,
    IINKnowledge)
    |||
    IntermediateOutRoute[sl_ION, in_ION, out_ION](IONKnowledge)
    |||
    External[sl_EN, in_EN, out_EN](ENKnowledge)
    |||
    Destination[sl_DN, rt_DN, in_DN, out_DN](kseed_IIN_DN)
)
|[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]|
Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]
```

```
where

process Source[sl_SN, rt_SN, in_SN, out_SN](kseed: SecretKey_Sort): exit :=
    (* Data transmission is initiated *)
    sl_SN ! start_transmission;
    i; (* It generates the data packet *)
    (* Source sends a data packet *)
    sl_SN ! sends_data;
    out_SN ! pck_DATA(DATAType, n1_SN_IIN, sym_encrypt(data1, kseed));
    in_SN ? ack:Ack_Sort; (* It receives local acknowledgment of the packet *)
    (* Source receives an error packet *)
    in_SN ? sync:Syncronization_Sort;
    in_SN ? error:ErrorPacket_Sort;
    sl_SN ! receives_error; (* Source receives an error packet *)
    exit
endproc

process IntermediateInRoute[sl_IIN, rt_IIN, in_IIN, out_IIN](kseed_prev, kseed_next: SecretKey_Sort,
IINKnowledge: Knowledge_Sort): exit :=
    (* Node in the route receives a data packet *)
    in_IIN ? sync:Syncronization_Sort;
    in_IIN ? data:DATAPacket_Sort;
    i; (* It verifies whether it is the intended receiver, modifies the packet *)
    out_IIN ! local_ack; (* It sends local acknowledgment of the packet *)
    (* Node in the route forwards the data packet *)
    out_IIN ! pck_DATA(DATAType, n1_IIN_DN, sym_encrypt(sym_decrypt(get_payload(data), kseed_prev),
    kseed_next));
    in_IIN ! timeout; (* It does not receive local acknowledgment of the packet *)
    sl_IIN ! broken_link;
    out_IIN ! pck_ERROR(ERRORType, n1_SN_IIN);
    exit
endproc

process IntermediateOutRoute[sl_ION, in_ION, out_ION](IONKnowledge: Knowledge_Sort): exit :=
    (* Node out of route receives a data packet *)
    in_ION ? data:DATAPacket_Sort;
    i; (* It verifies whether it is the intended receiver *)
    sl_ION ! discards_data; (* Node out of the route discards the data packet *)
    out_ION ! synchronization;
    (* Node out of route receives an error packet *)
    in_ION ? sync:Syncronization_Sort;
    in_ION ? error:ErrorPacket_Sort;
    sl_ION ! discards_error; (* Node out of the route discards the data packet *)
    out_ION ! synchronization;
    exit
endproc

process External[sl_EN, in_EN, out_EN](ENKnowledge: Knowledge_Sort): exit :=
    (* External node receives a data packet *)
    in_EN ? data:DATAPacket_Sort;
    sl_EN ! discards_data; (* External node discards the data packet *)
    out_EN ! synchronization;
    (* External node receives an error packet *)
    in_EN ? error:ErrorPacket_Sort;
    sl_EN ! discards_error; (* External node discards the error packet *)
    out_EN ! synchronization;
    exit
endproc

process Destination[sl_DN, rt_DN, in_DN, out_DN](kseed: SecretKey_Sort): exit :=
    exit (* Destination does not receives the data packet *)
endproc

process Channel[in_SN, out_SN, in_IIN, out_IIN, in_ION, out_ION, in_EN, out_EN, in_DN, out_DN]: exit :=
    out_SN ? data:DATAPacket_Sort;
    in_EN ! data;
    out_EN ? sync:Syncronization_Sort;
    in_IIN ! sync;
    in_IIN ! data;
    out_IIN ? ack:Ack_Sort;
    in_SN ! ack;
    out_IIN ? data:DATAPacket_Sort;
```

```
    in_ION ! data;
    out_ION ? sync:Syncronization_Sort;
    in_IIN ! timeout;
    out_IIN ? error:ErrorPacket_Sort;
    in_EN ! error;
    out_EN ? sync:Syncronization_Sort;
    in_ION ! sync;
    in_ION ! error;
    out_ION ? sync:Syncronization_Sort;
    in_SN ! sync;
    in_SN ! error;
    exit
endproc
endproc
endproc
endspec
```

Apêndice C

Especificação do Serviço - anodr_serv.lotos

Neste apêndice, apresenta-se a especificação em LOTOS do serviço esperado.

```
specification anodr_serv[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN, rt_DN]: noexit

library ANODRLIB endlib

behaviour

    Service[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN, rt_DN]

    where

    process Service[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN,
    rt_DN]: noexit :=
    (* Source node knows the route to the destination and the data forwarding process is finished successfully *)
    (
        RouteDiscoveryProcess1[sl_SN, rt_SN]
        >> DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
    )
    []
    (* Source node knows the route to the destination and the data forwarding process is initiated, but there is a
    broken link detection *)
    (
        RouteDiscoveryProcess1[sl_SN, rt_SN]
        >> MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN]
    )
    []
    (* Source node does not know the route to the destination and the data forwarding process is finished
    successfully *)
    (
        RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
        >> DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
    )
    []
    (* Source node does not know the route to the destination and the data forwarding process is initiated, but
    there is a broken link detection *)
    (
        RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN,
        as_EN, sl_DN, rt_DN]
        >> MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN]
    )
    >> Service[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION, sl_EN, as_EN, sl_DN, rt_DN]

        where

        (* The source knows the route to the destination *)
        process RouteDiscoveryProcess1[sl_SN, rt_SN]: exit :=
            sl_SN ! route_to_destination; (* Superior layer requests a route *)
            rt_SN ! checks_table; (* Routing table is verified *)
            rt_SN ! knows_route; (* Result: route is known *)
            sl_SN ! knows_route; (* Network layer is ready to transmit data *)
            exit
        endproc
```

```
(* The source does not know the route to the destination *)
process RouteDiscoveryProcess2[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION,
sl_EN, as_EN, sl_DN, rt_DN]: exit :=
    sl_SN ! route_to_destination; (* Superior layer requests a route *)
    rt_SN ! checks_table; (* Routing table is verified *)
    rt_SN ! does_not_know_route; (* Result: route is not known *)
    sl_SN ! does_not_know_route; (* Network layer is not ready to transmit data *)
    sl_SN ! start_route_discovery; (* Source initiates the route discovery process *)
    sl_SN ! sends_rreq; (* Source sends a route request packet *)
    (* Anonymity in relation to external node after receiving a RREQ *)
    as_EN ! does_not_discover_IDsource;
    as_EN ! does_not_discover_IDdestination;
    as_EN ! does_not_discover_venue_source;
    sl_EN ! discards_rreq; (* External node should discard the request packet *)
    (* Anonymity in relation to internal node in route after receiving a RREQ *)
    as_IIN ! does_not_discover_IDsource;
    as_IIN ! does_not_discover_IDdestination;
    as_IIN ! does_not_discover_venue_source;
    sl_DN ! receives_rreq; (* Destination receives a route request packet *)
    (* Anonymity in relation to internal node out of route after receiving a RREQ *)
    as_ION ! does_not_discover_IDsource;
    as_ION ! does_not_discover_IDdestination;
    as_ION ! does_not_discover_venue_source;
    sl_DN ! receives_rreq; (* Detection of duplicated packet *)
    sl_DN ! duplicated_packet;
    sl_DN ! discards_rreq; (* Destination node discards the duplicated packet *)
    sl_DN ! sends_rrep; (* Destination sends a route reply packet *)
    (* Anonymity in relation to external node after receiving a RREP *)
    as_EN ! does_not_discover_IDsource;
    as_EN ! does_not_discover_IDdestination;
    as_EN ! does_not_discover_venue_destination;
    sl_EN ! discards_rrep; (* A external node should discard the reply packet *)
    (* Anonymity in relation to internal node out of route after receiving a RREP *)
    as_ION ! does_not_discover_IDsource;
    as_ION ! does_not_discover_IDdestination;
    as_ION ! does_not_discover_venue_destination;
    sl_ION ! discards_rrep; (* A node out of the route should discard the reply packet *)
    (* Anonymity in relation to internal node in the route after receiving a RREP *)
    as_IIN ! does_not_discover_IDsource;
    as_IIN ! does_not_discover_IDdestination;
    as_IIN ! does_not_discover_venue_destination;
    sl_SN ! receives_rrep; (* Source receives a route reply packet *)
    sl_SN ! end_route_discovery; (* Route is established *)
    exit
endproc

(* Data forwarding process is finished successfully *)
process DataForwardingProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, as_IIN, sl_ION, as_ION,
sl_EN, as_EN, sl_DN, rt_DN]: exit :=
    sl_SN ! start_transmission; (* Data transmission is initiated *)
    sl_SN ! sends_data; (* Source sends a data packet *)
    (* Anonymity in relation to external node after receiving a DATA *)
    as_EN ! does_not_discover_IDsource;
    as_EN ! does_not_discover_IDdestination;
    as_EN ! does_not_discover_venue_source;
    as_EN ! does_not_discover_venue_destination;
    sl_EN ! discards_data; (* A external node should discard the data packet *)
    (* Anonymity in relation to internal node out of the route after receiving a DATA *)
    as_ION ! does_not_discover_IDsource;
    as_ION ! does_not_discover_IDdestination;
    as_ION ! does_not_discover_venue_source;
    as_ION ! does_not_discover_venue_destination;
    sl_ION ! discards_data; (* A node out of the route should discard the data packet *)
    (* Anonymity in relation to internal node in the route after receiving a DATA *)
    as_IIN ! does_not_discover_IDsource;
    as_IIN ! does_not_discover_IDdestination;
    as_IIN ! does_not_discover_venue_source;
    as_IIN ! does_not_discover_venue_destination;
    sl_DN ! receives_data; (* Destination receives the data packet *)
    sl_DN ! end_transmission; (* Transmission is finished successful *)
```

```
            exit
        endproc

        (* Data forwarding process is initiated, but there is a broken link detection *)
        process MaintenanceProcess[sl_SN, rt_SN, sl_IIN, rt_IIN, sl_ION, sl_EN, sl_DN, rt_DN]:
        exit :=
            sl_SN ! start_transmission; (* Data transmission is initiated *)
            sl_SN ! sends_data; (* Source sends a data packet *)
            sl_EN ! discards_data; (* A external node should discard the data packet *)
            sl_ION ! discards_data; (* A node out of the route should discard the data packet *)
            sl_IIN ! broken_link; (* A node detects a broken link *)
            sl_EN ! discards_error; (* An external node discards the error packet *)
            sl_ION ! discards_error; (* A node out of the route should discard the error packet *)
            sl_SN ! receives_error; (* Source receives an error packet *)
            exit
        endproc

    endproc

endspec
```