Especificação Formal de um Protocolo de Roteamento Anônimo para Redes Ad-Hoc

Andréa Martins Araujo[1*], **Aloysio de Castro P. Pedroza**[1]

[1]Grupo de Teleinformática e Automação
PEE/COPPE - DEL/POLI
Universidade Federal do Rio de Janeiro
Caixa Postal 68504 – CEP 21945-970
Rio de Janeiro, RJ

andrea@gta.ufrj.br, aloysio@gta.ufrj.br

Abstract. *An ad-hoc wireless network consists of a group of mobile nodes that communicate with each other without help of a fixed infraestructure. Some nodes might be malicious threatening the security or confidentiality of exchanged data. Using anonymous paths for communication provides security and privacy against traffic analysis. In this paper, we propose an anonymous routing protocol that prevents against traffic analysis hiding the source and destination of packets. An formal description technique (LOTOS) is used to design specifications and validate the protocol.*

Resumo. *Uma rede sem fio ad-hoc consiste num grupo de nós móveis que se comunicam entre si sem ajuda de uma infraestrutura fixa. Alguns nós podem ser maliciosos ameaçando a segurança e confidenciabilidade dos dados trocados. Usando caminhos anônimos para comunicação fornecemos segurança e privacidade contra a análise de tráfego. Nesse artigo, propomos um protocolo de roteamento anônimo que previne contra a análise de tráfego escondendo a fonte e o destino dos pacotes. Uma técnica de descrição formal (LOTOS) é usada para projetar especificações e validar o protocolo.*

1. Introdução

Uma rede sem fio ad-hoc consiste num grupo de nós móveis que se comunicam entre si através de *links* de rádio freqüência (RF) sem ajuda de uma infraestrutura fixa. Cada nó móvel atua como um roteador, tornando o mecanismo de roteamento mais vulnerável a falhas e ataques.

Aplicações como operações de resgate, incursão em terreno hostil e operações militares necessitam trocar informações de maneira segura. Enquanto mecanismos de segurança fim-a-fim protegem os dados, informações como as identidades dos usuário podem ser reveladas através da análise do tráfego. Daí a importância do estudo de métodos que proporcionem confidenciabilidade e privacidade à comunicação.

*Este trabalho é financiado com recursos da FAPERJ.

Para a Internet, foram desenvolvidos vários trabalhos, *Web-MIXes*[Berthold et al., 2000], *Stop-and-Go-MIXes*[Kesdogan et al., 1998], *Onion Routing*[Reed et al., 1998]entre outros, que se basearam no método proposto por Chaum [Chaum, 1981] para prover anonimato em correio eletrônico. Nessas redes, o nó fonte pode enviar mensagens secretas que só poderão ser decrifradas pelo receptor.

Em redes ad-hoc alguns estudos estão sendo realizados para prover anonimato como o protocolo de roteamento anônimo ANODR [Kong and Hong, 2003] que usa o conceito de *pseudonymity* [Pfitzmann and Köhntopp, 2000] e *trapdoor information*, o *Mix Method* [Jiang et al., 2001], onde são determinados os *mix nodes* que encaminharão pacotes, e um algoritmo de roteamento seguro [El-Khatib et al., 2003] que provê roteamento anônimo, usando somente criptografia e assinatura digital, sem utilização do conceito de nós misturadores.

Nesse artigo, nós propomos um protocolo de roteamento anônimo cujo funcionamento é baseado no protocolo DSR (*Dynamic Source Routing*) [Johnson et al., 2003]. Os dados trocados são protegidos por criptografia e o encaminhamento de pacotes é realizado somente por *mix nodes*, criando então, um caminho anônimo entre fonte e destino. Ao contrário do protocolo projetado por Jiang *et al.* [Jiang et al., 2001], responsável pela busca por nós misturadores, o protocolo proposto insere a busca por misturadores no próprio algoritmo de roteamento. Na verdade, o protocolo é uma versão extendida do DSR.

O protocolo proposto é especificado utilizando a linguagem para descrição formal LOTOS (*Language of Temporal Ordering Specification*)[ISO/IFC, 1988], acompanhada de verificação com o software de análise CADP (Pacote de Desenvolvimento CAESAR/ ALDEBARAN)[Fernandez et al., 1996]. Essa abordagem oferece uma especificação livre de ambigüidades e erros, além de facilitar a expansão e alteração do modelo. Existe uma variedade de trabalhos que utilizam LOTOS para desenvolvimento de protocolos como os realizados por Bagatelli *et al.*[Bagatelli et al., 2002] e Leduc *et al.* [Leduc and Germeau, 2000].

Este artigo está organizado da seguinte forma. Na seção 2 faremos uma breve revisão do funcionamento do protocolo de roteamento DSR. Na seção 3, descreveremos o protocolo proposto. Na seção 4, será apresentada a metodologia de análise. Na seção 5, serão analisados os resultados e por fim, na seção 6, descrevem-se algumas consiferações finais sobre o trabalho, bem como propostas de pesquisas futuras.

2. O Protocolo de Roteamento DSR

Atualmente, o DSR é o protocolo mais antigo do IETF para roteamento em redes ad-hoc e o mais difundo e testado entre os existentes. Por isso, foi selecionado para servir de base para o desenvolvimento do nosso protocolo.

O DSR opera sob demanda, ou seja, sua tabela de rotas é montada somente se houver necessidade de envio de mensagens. É composto por dois mecanismos principais: *route discovery* (descoberta de rota) e *route maintenance* (manutenção de rota).

A descoberta de rota é iniciada quando um nó recebe um pacote a ser enviado e nao possui rota armazenada para o destino. Então, o nó, chamado fonte, envia um *route*

request (requisição de rota) por difusão. Cada nó que recebe esse pacote verifica se ele já foi recebido anteriormente. Caso afirmativo, o pacote é descartado. Caso contrário, o nó, que não é o nó destino, coloca seu endereço como próximo nó intermediário e retransmite o pacote, por *flooding* (inundação) e envia um *ack* (confirmação de recebimento) ao nó anterior. O próximo nó que receber o *route request* fará o mesmo até que o pacote alcance o destino. Este nó, ao receber o *route request*, envia um *route reply* para a fonte através do caminho reverso.

A manutenção das rotas é feita quando cada nó que transmite um pacote é responsável por armazenar o *ack* recebido do nó seguinte, sem que haja sua retransmissão até o nó fonte. Caso ocorra quebra de um *link*, o nó que encaminha um pacote e ao não receber o *ack*, procura uma nova rota para o destino em sua tabela. Caso não tenha esta rota, o nó envia um *route error* ao seu nó antecessor que age da mesma forma até que o *route error* alcance o nó fonte. Quando o nó fonte receber esse pacote, verifica se existe outra rota para o destino. Caso negativo, deve-se iniciar nova descoberta de rota.

3. O Protocolo Proposto

O protocolo apresentado neste trabalho baseia-se no funcionamento do protocolo DSR e tem como objetivo o estabelecimento de um caminho anônimo entre a fonte e o destino da comunicação. A idéia é esconder a identidade da fonte e do destino através de criptografia oferecendo, então, privacidade à comunicação. Para tal, usamos o conceito de nós misturadores desenvolvido por Chaum [Chaum, 1981] e estabelecemos que o encaminhamento de pacotes só seria realizado por estes nós, pois eles são confiáveis e responsáveis pelo anonimato da comunicação.

Neste trabalho foram considerados somente ataques do tipo passivo, mais especificamente análise de tráfego. Todos os nós da rede possuem um par de chaves assimétricas, uma chave pública e outra privada. Os nós misturadores formam um grupo que divide o mesmo par de chaves assimétricas. Somente os nós misturadores conhecem todas as chaves públicas da rede.

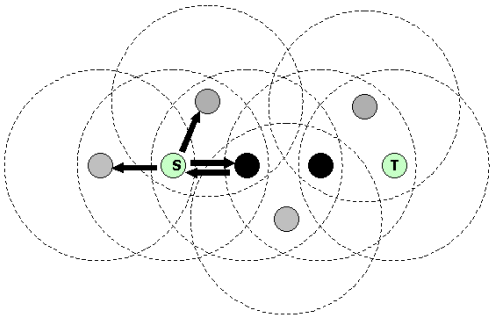
3.1. Notação

As notações são definidas a seguir:

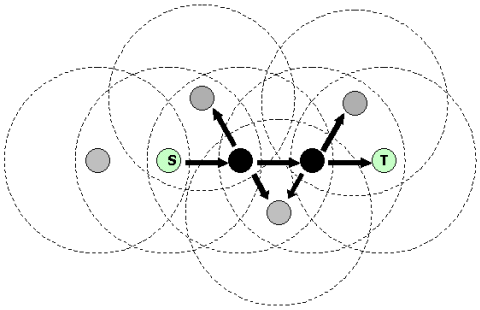
- Ki: Chave pública do nó i;
- Si: Chave privada do nó i;
- Km : Chave pública dos misturadores;
- Sm : Chave privada dos misturadores.

3.2. O Protocolo

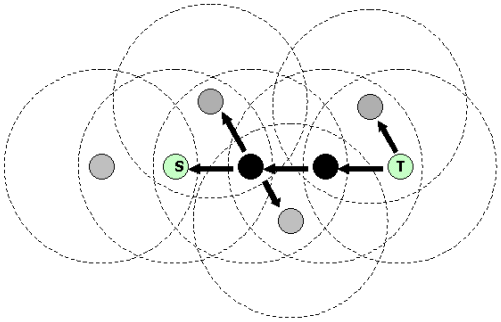
O funcionamento básico do nosso protocolo está descrito a seguir:



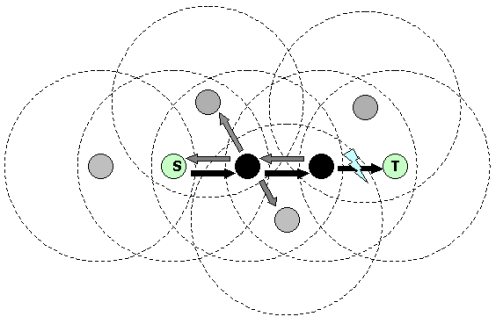
(a) Envio do Mix Request e recebimento do Mix Replay



(b) Envio do Route Request



(c) Envio do Route Replay



(d) Envio do Route Error

Figura 1: Sequência de operação do Protocolo

- Passo 1 - O nó *source* S deseja estabelecer comunicação anônima com o nó *target* T, porém não tem caminho para T. Então, S deve iniciar a descoberta de rota com encaminhamento de pacotes através de nós misturadores. S não sabe quais nós são misturadores, então, envia um *mix request* em difusão.

- Passo 2 - Os nós misturadores que receberem um *mix request* respondem com um *mix reply* contendo Km, vide Figura 1(a)(os nós pretos são misturadores)

- Passo 3 - Ao receber um *mix reply*, S monta um *route request* que contém: o endereço de origem (endereço de S) criptografado por Kt, o endereço de destino final (endereço de T) criptografado por Km e as informações destinadas somente a T criptografadas por Kt. S envia o *route request* para o misturador do qual recebeu um *mix reply*, vide Figura 1(b)

- Passo 4 - Quando um misturador recebe um *route request*, ele decifra o endereço de destino final através de sua chave privada e verifica se é vizinho de T. Caso negativo, ele cifra novamente o endereço de T com Km e envia o pacote em difusão.

- Passo 5 - O próximo misturador age da mesma forma, até que o pacote seja recebido por um misturador vizinho a T. Neste caso, o misturador criptografa o endereço de T com Kt e envia por difusão. Somente T pode decifrar o endereço da mensagem, os demais nós descartam o pacote.

- Passo 6 - T monta seu *route reply* que contém: o endereço de origem (endreço de T) e o endereço de destino final (endereço de S) criptografados por Ks. T envia o *route reply* para S pelo caminho reverso, vide Figura 1(c)

- Passo 7 - Como o endereço de destino final do *route reply* está criptografado por Ks, o misturador vizinho a S não sabe para quem entregar o pacote. Então, o pacote é enviado por difusão. E somente o nó S é capaz de decifrá-lo, os demais nós descartam o pacote. Fica estabelecido, então, o caminho anônimo entre S e T.

- Passo 8 - Caso haja uma quebra de link, o nó misturador deve enviar um *route error* para seu nó antecessor (do qual recebeu um *route request*) e assim por diante até alcançar o misturador vizinho ao nó S. Como esse nó misturador desconhece a identidade da fonte, ele envia o *route error* por difusão, vide Figura 1(d). Todos os nós da vizinhança recebem esse pacote, que só terá significado para S. Caso não exista outra rota armazenada para T, S deve iniciar o processo de descoberta de rota.

4. Metodologia

Para a modelagem do protocolo foi utilizada técnicas de descrição formal (*FDT - Formal Description Techniques*), no caso a linguagem LOTOS. A vantagem de usar as FDTs se deve, sobretudo, a sua capacidade de tornar as especificações livres de ambigüidades e erros, além de atuar como um guia para a implementação. Em particular, essas técnicas são adequadas à análise dos aspectos de segurança do protocolo.

O projeto em questão, adotou uma rede ad-hoc simples, composta por cinco nós, dos quais dois são nós misturadores (*mix nodes*), vide Figura 2. O ponto de partida do trabalho foi uma rede com apenas dois nós, pode parecer desnecessário essa simulação, mas é fundamental na medida em que promove a correção rápida da especificação. E a partir desses dois nós, adicionou-se nó a nó até atingir o cenário atual.

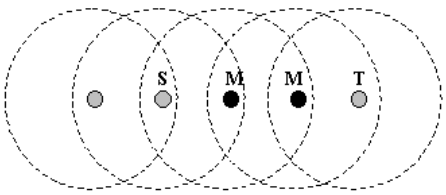


Figura 2: Rede Ad-Hoc simples

Com o objetivo de facilitar a análise, foi considerada duas situações: rede sem a presença de nós intrusos e com nós intrusos. A especificação do protocolo inclui a descrição, em LOTOS, do comportamento interno do nó e de suas interações com o meio externo e os demais nós da rede.

Utilizou-se para análise, o pacote de ferramenta para engenharia de protocolos CADP. O uso da interface gráfica TCL-TK integrante do pacote CADP, Eucalyptos, proporcionou um acesso fácil às funcionalidades das ferramentas CADP, pela simplicidade em relação a chamada das ferramentas pela linha de comando. Faz parte dessa análise com CADP, a compilação do código, simulação, verificação, validação e testes da descrição formal do protocolo.

Com a especificação escrita em LOTOS, o próximo passo consiste na sua compilação através das ferramentas *Caesar*. O resultado desta operação é um arquivo (.bcg) contendo o grafo chamado LTS (*Labelled Transition Systems*). Esse grafo contém as possíveis seqüências de execução do protocolo estudado.

As propriedades do protocolo, bem como a visualização do grafo, são exibidos a partir do emprego do conjunto de ferramentas *BCG* (integrante do pacote CADP)[Fernandez et al., 1996]. Entre todas as propriedades, duas são as mais testadas:

- Equivalência Observacional: Usada para verificar se o protocolo especificado representa os serviços discriminados inicialmente como requisitos de projeto.

- Minimização: Usada para reduzir os grafos do sistema de transições rotuladas ou LTS (*Labelled Transition Systems*), retirando, então, as redundâncias e permitindo uma análise mais simples.

4.1. A Linguagem LOTOS

LOTOS é uma linguagem de descrição formal adequada para a especificação de protocolos e sistemas distribuído e concorrentes em geral. Esta linguagem, padronizada pela ISO, é composta por duas sub-linguagens:

- LOTOS Básico: sua representação é inspirada em álgebra de processos, CCS (*Calculus of Communicating Systems*) e CSP (*Communicating Sequencial Process*), e descreve o modelo comportamental do sistema;

- LOTOS Completo: inclui a linguagem de especificação ACTONE (*Abstract Data Type Formalism*), onde os tipos de dados são descritos por seus operadores e equações, as quais são especificadas mediante o emprego de operações algébricas.

A Tabela 1 apresenta algumas operações em LOTOS e suas respectivas descrições.

OPERADOR	DESCRIÇÃO
A[f,g]	Instanciação do processo A, onde f e g são as portas por onde dados são trocados
A[f,g](S,T)	Chamada do processo A com portas f e g e parâmetros de valores S e T
hide g, h in	Portas serão escondidas na apresentação dos resultados
p!V?X:T	Interação pela porta p, envio do valor V e recepção em X de um valor do tipo T
A ||| B	Funcionamento dos processos A e B são independentes e em paralelo
A || B	Sincronismo entre os processos A e B
A |[p]| B	Sincronismo entre os processos A e B através da porta p
A [ ] B	Escolha entre os processos A e B, com igual probabilidade
A >> B	O processo A habilita o processo B
exit	Termina com sucesso
stop	Parada de processo

Tabela 1: Alguns operadores de LOTOS

5. Análise dos Resultados

De forma a ilustrar a metodologia adotada para a verificação do protocolo, toma-se como exemplo a versão 6 do protocolo, numa rede com e sem intrusos.

5.1. Rede sem Intrusos

Nessa verificação foram definidos cinco processos, que representam os cinco nós, interligados por um meio de comunicação simples para troca de dados. A representação do comportamento do processo *route discovery* é descrita no trecho de código a seguir:

```
process Route_Discovery[U0, U1, U2, U3, Ut] : exit :=

hide M0_in, M0_out, M1_in, M1_out, M2_in, M2_out, M3_in, M3_out, Mt_in, Mt_out in
(
(
Source[U0, M0_in, M0_out] (Kt, Km, Ks, R, Ss, As, At, Ab, A1)
 |||
No1_mix[U1, M1_in, M1_out] (Kt, Km, Ks, R, Sm, As, At, Ab, A1)
 |||
No2[U2, M2_in, M2_out](As, Ab)
 |||
No3_mix[U3, M3_in, M3_out] (Kt, Km, Ks, R, Sm, As, At, Ab, A1, A3)
 |||
Target[Ut, Mt_in, Mt_out](Kt, Km, Ks, R, St, As, At, Ab, A1, A3)
)
|[M0_in, M0_out, M1_in, M1_out, M2_in, M2_out, M3_in, M3_out, Mt_in, Mt_out]|
(
Meio[M0_in, M0_out, M1_in, M1_out, M2_in, M2_out, M3_in, M3_out, Mt_in, Mt_out]
(Kt, Km, Ks, R, As, At, Ab, A1, A3)
)
)
```

O processo *Route_Discovery* inclui vários processos que correspondem aos nós da rede e um processo chamado Meio que representa o canal de comunicação entre os nós. O operador | | | demonstra que os processos possuem comportamento independentes e paralelos, enquanto o operador | [M0_in, ..., Mt_out] | representa o sincronismo entre os processos através das portas de comunicação M0_in, ..., Mt_out. A especificação também contém a representação do comportamento do processo *route maintenance*, com e sem situações onde haja quebra de link.

O serviço proposto para este modelo tem por objetivo estabelecer rotas para um destino e restabelecer a transmissão de dados, caso ocorra uma quebra num link, através do mecanismo de *route maintenance*. O próximo trecho representa o modelo do serviço oferecido pelo protocolo.

```
Specification ANDSR6_SERVICE [U0, U1, U2, U3, Ut] : noexit

library ANDSR_LIB6 endlib

behaviour

SERVICE [U0, U1, U2, U3, Ut]

where

process SERVICE [U0, U1, U2, U3, Ut] : noexit :=
(
(Discovery[U0, U1, U2, U3, Ut] >> Maintenance [U0, U1, U2, U3, Ut]
>> SERVICE [U0, U1, U2, U3, Ut])
 [ ]
(Discovery[U0, U1, U2, U3, Ut] >> Maintenance [U0, U1, U2, U3, Ut]
>> Falha [U0, U1, U2, U3, Ut] >> SERVICE [U0, U1, U2, U3, Ut])
```

```
 [ ]
(Discovery[U0, U1, U2, U3, Ut] >> Falha [U0, U1, U2, U3, Ut]
>> SERVICE [U0, U1, U2, U3, Ut])
)
where

process Discovery [U0, U1, U2, U3, Ut] : exit :=
U0 ! Rota_para_Destino; U0 ! RotaEncontrada; exit
endproc

process Maintenance [U0, U1, U2, U3, Ut] : exit :=
U0 ! Iniciar_Transmissao; U0 ! msgRecebida; exit
endproc

process Falha [U0, U1, U2, U3, Ut] : exit :=
 U0 ! Iniciar_Transmissao; U0 ! Link_Error; exit
endproc
endproc
endspec
```

Nosso serviço, inicialmente, foi modelado com três comportamentos. O primeiro se refere ao mecanismo de *route discovery*, cujo fim é seguido pela transmissão de dados com manutenção de rotas, neste caso não há falhas nos links. No segundo caso, o fim do *route discovery* é seguido por uma transmissão de dados normal e por uma outra com quebra de link. E por fim, no terceiro comportamento, o *route discovery* é seguido por uma transmissão mal sucedida, onde houve quebra de link antes da informação alcançar o destino. O processo SERVICE é composto por três subprocessos representando cada mecanismo do protocolo, como pode ser visto no trecho anterior. A Figura 3 apresenta o grafo LTS desse serviço. Os círculos representam os estados, as setas representam as transições e os rótulo representam as ações executadas.

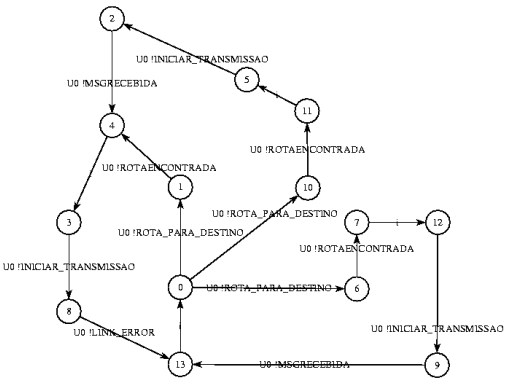


Figura 3: LTS minimizado do Serviço

A Tabela 2 apresenta o número de estados e transições alcançados pelo modelo antes e depois da minimização.

MODELO	ESTADOS	TRANSIÇÕES	MINIMIZADO
Protocolo	243	236	não
Protocolo	211	213	sim
Serviço	19	21	não
Serviço	14	16	sim

Tabela 2: Geração dos LTS

5.2. Rede com presença de Intrusos

O exemplo descrito anteriormente modela o funcionamento do protocolo numa rede sem a presença de nós intrusos. Para modelar o comportamento do nosso protocolo diante de ataques do tipo análise de tráfego, o nó dois foi considerado um intruso que analisa o tráfego em diferentes pontos da rede. O nó intruso poderia observar o tráfego entre o nó fonte e o primeiro nó misturador, entre o primeiro misturador e o segundo e entre o segundo e o nó destino. Porém, nossa análise se restringiu ao mecanismo de *route discovery*. O código a seguir apresenta o um trecho da especificação modelando um intruso entre o nó fonte e o primeiro misturador.

```
behaviour

hide M0_in, M0_out, M1_in, M1_out, Mi_in, Mi_out,M3_in, M3_out, Mt_in, Mt_out in
(
(
Source[U0, M0_in, M0_out] (Kt, Km, Ks, R, Ss, As, At, Ab, A1)
|||
Intruder[Ui, Mi_in, Mi_out](Kt, Km, Ks, R, Sm, As, At, Ab, A1, A3)
|||
No1_mix[U1, M1_in, M1_out] (Kt, Km, Ks, R, Sm, As, At, Ab, A1)
|||
No3_mix[U3, M3_in, M3_out] (Kt, Km, Ks, R, Sm, As, At, Ab, A1, A3)
|||
Target[Ut, Mt_in, Mt_out] (Kt, Km, Ks, R, St, As, At, Ab, A1, A3)
)
|[M0_in, M0_out, M1_in, M1_out, Mi_in, Mi_out, M3_in, M3_out, Mt_in, Mt_out]|
(
Meio[M0_in, M0_out, M1_in, M1_out, Mi_in, Mi_out, M3_in, M3_out, Mt_in, Mt_out]
(Kt, Km, Ks, R, As, At, Ab, A1, A3)
)
)
```

O respectivo serviço está descrito no próximo trecho de código.

```
Specification INTRUDER_SERVICE [U0, U1, Ui, U3, Ut] : noexit

library INTRUDERLIB endlib

behaviour

SERVICE [U0, U1, Ui, U3, Ut]

where

process SERVICE [U0, U1, Ui, U3, Ut] : noexit :=

U0 ! Rota_para_Destino;
Ui ! Descobri_S_quer_Rota;
Ui ! Descobri_Km;
Ui ! Nao_descobri_identidades;
U0 ! RotaEncontrada;
Service [U0, U1, Ui, U3, Ut]
endproc

endspec
```

A Figura 4 apresenta o grafo LTS do modelo de serviço para este exemplo, onde existe um intruso entre a fonte e o primeiro nó misturador. Observa-se que o intruso conseguiu perceber que S gostaria de estabelecer rota, pois S procurou por um misturador na sua vizinhança, mas não consegue comprovar essa situação concretamente. Logo, não conseguiu descobrir a identidade da fonte e do destino do tráfego analisado.

A Tabela 3 apresenta o número de estados e transições antes e depois da minimização para as diferentes posições do intruso na rede.

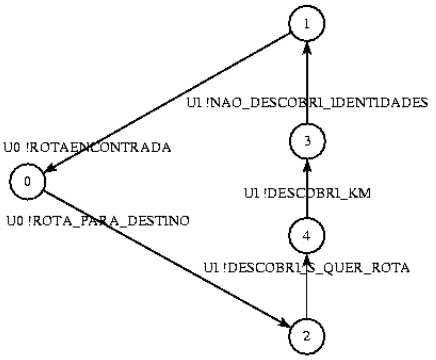


Figura 4: LTS minimizado do Serviço - rede com intruso

MODELO	INTRUSO	ESTADOS	TRANSIÇÕES	MINIMIZADO
Route Discovery	Entre a fonte e o primeiro misturador	55	55	não
Route Discovery	Entre a fonte e o primeiro misturador	54	54	sim
Serviço	Entre a fonte e o primeiro misturador	5	5	não
Serviço	Entre a fonte e o primeiro misturador	5	5	sim
Route Discovery	Entre o primeiro misturador e o segundo	46	46	não
Route Discovery	Entre o primeiro misturador e o segundo	45	45	sim
Serviço	Entre o primeiro misturador e o segundo	3	3	não
Serviço	Entre o primeiro misturador e o segundo	3	3	sim
Route Discovery	Entre o segundo misturador e o destino	46	46	não
Route Discovery	Entre o segundo misturador e o destino	45	45	sim
Serviço	Entre o segundo misturador e o destino	3	3	não
Serviço	Entre o segundo misturador e o destino	3	3	sim

Tabela 3: Geração dos LTS - rede com intruso

5.3. Verificação

O resultado mais importante é comprovar que o serviço foi oferecido, ou melhor, se o protocolo conseguiu estabelecer rota para o destino e fazer sua devida manutenção. Com o uso da ferramenta *Aldebaran* [Fernandez et al., 1996] obteve-se **TRUE** como resultado, ou seja, o modelo de protocolo e serviço descritos em LOTOS apresentam equivalência observacional, demonstrando que o protocolo especificado atende aos requisitos do serviço. Este resultado comprova a exatidão do modelo proposto. Constata-se, além da equivalência entre protocolo e serviço, algumas prorpriedades são verificadas:

- O protocolo descrito é vivo, ou seja, não há presença de *deadlocks*(bloqueios);
- É reinicializável, visto que, de qualquer estado, é possível alcançar o estado inicial;
- Também não há presença de *livelocks*.

Em relação a rede com intruso, observa-se que o modelo de serviço para estes casos são iguais, exceto para o caso em que o intruso se encontra entre a fonte e o primeiro misturador. Neste caso há mais duas ações: **Ui ! Descobri_S_quer_Rota** e **Ui ! Descobri_Km**. A primeira indica que o intruso verificou que a fonte S deseja estabelecer rota e a segunda ele obtém a chave pública de Km trocada entre a fonte e o misturador. Em relação ao algoritmo de *route discovery*, as diferenças são pequenas. Para o caso em que o intruso está no primeiro link (entre a fonte e o primeiro misturador), existe um número maior de ações e para os outros dois casos as diferenças se dão somente pela posição do intruso na rede. É verificado também que o número de estados alcançados para esses dois últimos são iguais. Por fim, para essas situações, o intruso não conseguiu sucesso, ou seja, as identidades dos nós não foi revelada.

6. Considerações Finais

Privacidade e Anonimato são aspectos de segurança cada vez mais importantes para a comunicação em redes móveis ad-hoc. O protocolo apresentado neste trabalho cria rotas anônimas com o uso de encaminhamento de pacotes por nós misturadores. O objetivo é criar um caminho anônimo entre a fonte e o destino, escondendo a identidade de ambos através de criptografia. Ataques do tipo análise de tráfego teriam menores chances de obter a identidade da origem e do destino do pacote, bem como as informações trocadas.

A vantagem do algoritmo proposto é que a procura por nós misturadores está inserida no próprio algoritmo de roteamento como um único algoritmo, ao contrário do método desenvolvido por Jiang *et al.* [Jiang et al., 2001] que propõe um protocolo para a busca de nós misturadores. Além disso, o algoritmo proposto não precisa usar o serviço de uma autoridade certificadora (CA) como o protocolo desenvovido por El-Khatib *et al.* [El-Khatib et al., 2003]. O protocolo modelado pode ser considerado uma extensão do protocolo DSR, devido a inclusão do estabelecimento de rotas anônimas. Adicionalmente, no modelo, a fonte não precisa ter um conhecimento global da topologia da rede. Em relação aos aspectos de segurança, o algoritmo não está preparado para enfrentar ataques do tipo *Denial-of-Service (DoS)*. E para o caso em que os nós misturadores sejam corrompidos, só é possível descobrir a identidade do nó destino. Nossa abordagem utiliza a técnicas de descrição formal LOTOS, com seu rigoroso processo de validação que gerou uma especificação livre de ambigüidades e erros, além de facilitar a expansão e alteração do modelo.

Este trabalho constitui o nosso primeiro passo para prover uma comunicação anônima em redes ad-hoc. Nosso objetivo é expandir o modelo incluindo mais nós, na tentativa de representar o maior número de comportamentos que caracterizam o protocolo. Utilizaremos outras análise disponíveis no pacote de ferramentas que ainda não foram exploradas, a fim de aperfeiçoar nossa análise. Desta forma, teremos um modelo completo do protocolo. Em futuros trabalhos, planejamos também avaliar o desempenho do protocolo.

Referências

Bagatelli, R., Moura, D. F. C., and de Castro P. Pedroza, A. (2002). Especificação Formal de uma Arquitetura de Suporte à Descoberta de Serviços em Redes Móveis Ad Hoc. In *V Workshop de Métodos Formais, WMF'2002*, Gramado, RS, Brasil.

Berthold, O., Federrath, H., and Köhntopp, M. (2000). Project "Anonymity and Unobservability in the Internet". In *In Computers Freedom and Privacy Conference, CFP'2000, Workshop on Freedom and Privacy by Design*.

Chaum, D. L. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90.

El-Khatib, K., Korba, L., Song, R., and Yee, G. (2003). Secure Dynamic Distributed Routing Algorithm for Wireless Ad Hoc Networks. In *Proceedings in International Conference on Parallel Processing Workshops*.

Fernandez, J. C., Garavel, H., Kerbrat, A., Mounier, L., Mateescu, R., and Sighireanu, M. (1996). CADP: A Protocol Validation and Verification Toolbox. volume 1102, pages 437–440, New Brunswick, NJ, USA. Springer-Verlag.

ISO/IFC (1988). *Is 8807: Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Techniques Based on The Temporal Ordering of Observacional Behavior*.

Jiang, S., Vaidya, N., and Zhao, W. (2001). A Dynamic Mix Method for Wireless Ad-Hoc Networks. *Military Communications Conference, MILCOM 2001)*.

Johnson, D. B., Maltz, D. A., and Hu, Y.-C. (2003). The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). In *Work in progress, draft-ietf-manet-dsr*. IEEE.

Kesdogan, D., Egner, J., and Buschkes, R. (1998). Stop-and-Go-Mixes Providing Probabilistic Security in an Open System. *Information Hiding: Second International Workshop*, 1525:83–98.

Kong, J. and Hong, X. (2003). ANODR: ANonymous on Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. In *Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking Computing*. ACM Press.

Leduc, G. and Germeau, F. (2000). Verification of Security Protocols using LOTOS-Method and Application. *Computer Communications*, 23:1089–1103.

Pfitzmann, A. and Köhntopp, M. (2000). Anonymity, Unobservability and Pseudonymity - A Proposal of Terminology. volume 2009 of *Lecture Notes in Computer Science*, Berkeley, CA, USA. Springer-Verlag, Berlin, Germany.

Reed, M. G., Syverson, P. F., and Goldschlag, D. M. (1998). Anonymous Connections and Onion Routing. *IEEE Journal on Special Areas in Communications*, 16(4):482–494.