Implementação de um esquema de gerenciamento de chaves auto-organizado para redes ad hoc móveis

Eduardo da Silva, Aldri Luiz dos Santos[1]

[1] Universidade Federal do Paraná – Departamento de Informática
Caixa Postal 19081 – CEP 81531-900 – Curitiba – PR

eduardos,aldri@inf.ufpr.br

Abstract. *The mobile ad hoc networks security is a challange and objective of many studies at last years. Several papers are published approaching this issue. Amongst the essential security mechanisms for a network, is key authentication and management. In this paper is done the implementation and tests of self-organized key management model fo mobile ad hoc networks presented in [Capkun et al. 2003a]. This model allows users to generate themselves theirs private-public keys, to issue certicates each others and to perform authentication without any central service. This model is very interesting to mobile ad hoc networks, since it does not require any trusted authority, even in the system initialization phase. The implementation and tests will be performed using NS-2 [NS-2 2007].*

Resumo. *A segurança para as redes ad hoc móveis (MANETs) é um desafio e objeto de muitos estudos nos últimos anos. Diversos artigos são publicados abordando esse assunto. Dentre os mecanismos de segurança essenciais para uma rede está a autenticação e gerenciamento de chaves. Nesse artigo é realizada a implementação e testes do modelo de gerenciamento de chaves auto-organizado para redes ad hoc móveis apresentado em [Capkun et al. 2003a]. Esse modelo permite que os usuários gerem seus pares de chaves públicas e privadas, emitam certificados uns aos outros e realizem autenticação sem qualquer serviço centralizado. Esse modelo é bastante interessante para redes ad hoc móveis, uma vez que não necessita de qualquer autoridade confiável, nem mesmo na fase de inicialização do sistema. A implementação e testes serão realizados com a ferramenta NS-2 [NS-2 2007].*

1. Introdução

Uma rede *ad hoc* móvel (MANET) é um grupo de computadores móveis sem fio, denominados nós, que cooperam entre si realizando o encaminhamento de pacotes, de forma a possibilitar a comunicação entre os nós, mesmo que a distância entre os nós comunicantes ultrapasse ao limite de transmissão do sinal direto sem fio [Hu et al. 2005]. Caso a distância entre os dois nós pares comunicantes ultrapasse ao limite de transmissão de sinal direto entre eles, os outro nós pertencentes à rede *ad hoc* realizam o roteamento e encaminhamento dos pacotes pela rede.

Uma característica das redes *ad hoc*, é a ausência de infra-estrutura, ou seja, os nós pertencem a uma rede onde não existe o papel de uma unidade central, como por exemplo, uma estação base. Diversas são as dificuldades encontradas para prover segurança

nas MANETs, devido a diversos fatores. Alguns desses fatores foram discutidos em [Buttyan and Hubaux 2003] e são apresentados a seguir:

- o canal de comunicação sem fio é vulnerável, assim, as mensagens transmitidas na rede podem ser capturadas e também mensagens falsas podem ser injetadas na rede;
- os nós também são vulneráveis, principalmente por não estarem localizados em uma sala fisicamente protegida;
- a ausência de infraestrutura faz com que soluções clássicas de segurança, baseadas em autoridades certificadoras, não seja aplicáveis a essas redes;
- a topologia dinâmica dessas redes dificulta o processo de roteamento dos pacotes na rede, e ainda, informações incorretas de rotas podem ser criadas por intrusos ou por nós comprometidos. O desafio dos algoritmos de roteamento está em distinguir entre as mudanças de rotas causadas por mudança na topologia, das mudanças que foram causadas de forma maliciosa.

Segundo [Capkun et al. 2003a] existem duas maneiras de se adicionar segurança em uma rede *ad hoc*: a primeira, por meio de um domínio de autoridade, onde os certificados e/ou chaves são emitidos por uma autoridade única, geralmente na fase de configuração ou inicialização do sistema; a segunda, por meio de uma completa auto-organização do sistema, onde a solução não confia ou depende de qualquer autoridade confiável ou um servidor fixo, nem mesmo na fase de inicialização.

Nesse artigo, é assumida a segunda abordagem, e é apresentado um sistema de gerenciamento de chaves-públicas auto-organizável que permite aos usuários criarem, armazenarem, distribuirem e revogarem seus chaves públicas sem a ajuda de nenhuma autoridade confiável ou servidor fixo. Esse sistema foi proposto por [Capkun et al. 2003a], e é parte de um projeto de pesquisa intitulado *Terminodes*[1].

[Capkun et al. 2003a] propõe um esquema de gerenciamento de chaves usando os princípios do PGP (*Pretty Good Privacy*). O funcionamento e as características do PGP são discutidas em [Zimmermann 1995] e, dentre as principais características do PGP, está o gerenciamento de chaves: no PGP o acordo das chaves é realizado mediante a confiança entre os pares comunicantes. Logo, as redes que utilizam uma comunicação baseada em PGP, possuem um fenômeno chamado do *Small World* [Capkun et al. 2002].

O artigo está organizado da seguinte forma: na seção 2 são apresentadas as características do gerenciamento de chaves em redes *ad hoc*; também é discutido os problemas da utilização de uma autoridade certificadora central em uma rede *ad hoc*; ainda nessa seção são apresentadas as operações básicas para o funcionamento do modelo de gerência de chaves proposto em [Capkun et al. 2003a], assim como é apresentado o esquema de criação e troca dos certificados de chaves públicas; a seção 3 apresenta, de forma resumida, o funcionamento do algoritmo de construção dos repositórios de certificados. A implementação desse algoritmo é realizada utilizando o simulador NS-2 (*Network Simulador -2*) [NS-2 2007] e são apresentados os resultados e as conseqüências de sua utilização, como utilização de recursos e vulnerabilidades do modelo.

[1]Projeto Terminodes - http://www.terminodes.org

2. O gerenciamento de chaves em redes *ad hoc*

Muitos objetivos da segurança podem ser alcançados utilizando mecanismos criptográficos. Por outro lado, os mecanismos criptográficos desenvolvidos para redes *ad hoc*, bem como para as redes tradicionais, confiam que o gerenciamento das chaves criptográficas é realizado de forma apropriada.

Em redes *ad hoc* o gerenciamento de chaves é um grande desafio. Os mecanismos tradicionais de gerenciamento de chaves não são adequados para as redes *ad hoc*, uma vez que necessitam de uma entidade confiável central, conhecida da como Autoridade Certificadora (AC). O principal problema de qualquer sistema de segurança baseado em chaves-públicas é fazer com que a chave pública de cada usuário da rede seja disponibilizada para os demais usuários de forma que sua autenticidade seja verificada. Esse problema é ainda maior nas MANETs, visto que não existe o papel de uma autoridade central na rede. Além disso, elas podem ser particionadas devido o dinamismo de sua topologia.

Uma abordagem amplamente utilizada são os certificados de chaves-públicas. Um certificado de chave-pública é uma estrutura de dados na qual a chave-pública é associada a uma identidade por meio da assinatura digital do emissor do certificado. Na maioria dos esquemas propostos para a utilização de certificados de chaves-pública, existe uma terceira entidade, conhecida como Autoridade Certificadora (AC).

No entanto, é uma questão problemática utilizar uma única AC em uma MANET [Zhou and Haas 1999], uma vez que a AC é responsável pela segurança da rede inteira, sendo um ponto vulnerável da rede. Caso a AC fique indisponível, os nós não podem obter as chaves públicas atuais dos outros nós, e como conseqüência, não podem estabelecer uma comunicação segura com os demais nós. Também se uma AC fica comprometida, um atacante pode assinar certificados falsos usando uma chave privada obtida da AC comprometida e, personificar qualquer outro nó, ou ainda, revogar um certificado válido. Assim, a proposta de [Capkun et al. 2003a] é que os próprio nós emitam os certificados uns aos outros, não precisando assim de qualquer autoridade certificadora central.

2.1. Operações básica da solução

Esta seção apresenta as operações básicas do modelo de gerenciamento de chaves proposto por [Capkun et al. 2003a]. Nesse modelo, as chaves públicas e os certificados do sistema são representados por um grafo direcionado $G(V, A)$, onde V representa o conjunto dos vértices e A o conjunto das arestas. Os vértices representam as chaves públicas e as arestas representam os certificados. Resumidamente, caso exista uma aresta direta conectando dois vértices (K_u à K_v), então existe um certificado assinado com a chave pública de u que associa K_v ao uma identidade.

Conforme explicado em [Capkun et al. 2003b], esse modelo de gerenciamento de chaves-públicas auto-organizável é similar ao PGP, no sentido que os usuário emitem certificados uns aos outros, baseados em seus relacionamentos pessoais.

Em [Buttyan and Hubaux 2003], os autores definem, que diferente do PGP, os certificados não são armazenados em um repositório central. Pelo contrário, os repositórios de chaves são distribuídos pelos nós da rede. Cada usuário mantém um repositório local de certificados. Quando um usuário u deseja verificar a autenticidade da chave-pública

K_v de um usuário v, o usuário u tenta encontrar uma caminho direto do vértice K_u até K_v. Ele realiza essa operação por meio de uma fusão dos repositório locais de certificados de K_u e K_v, conforme mostrado na Figura 1.

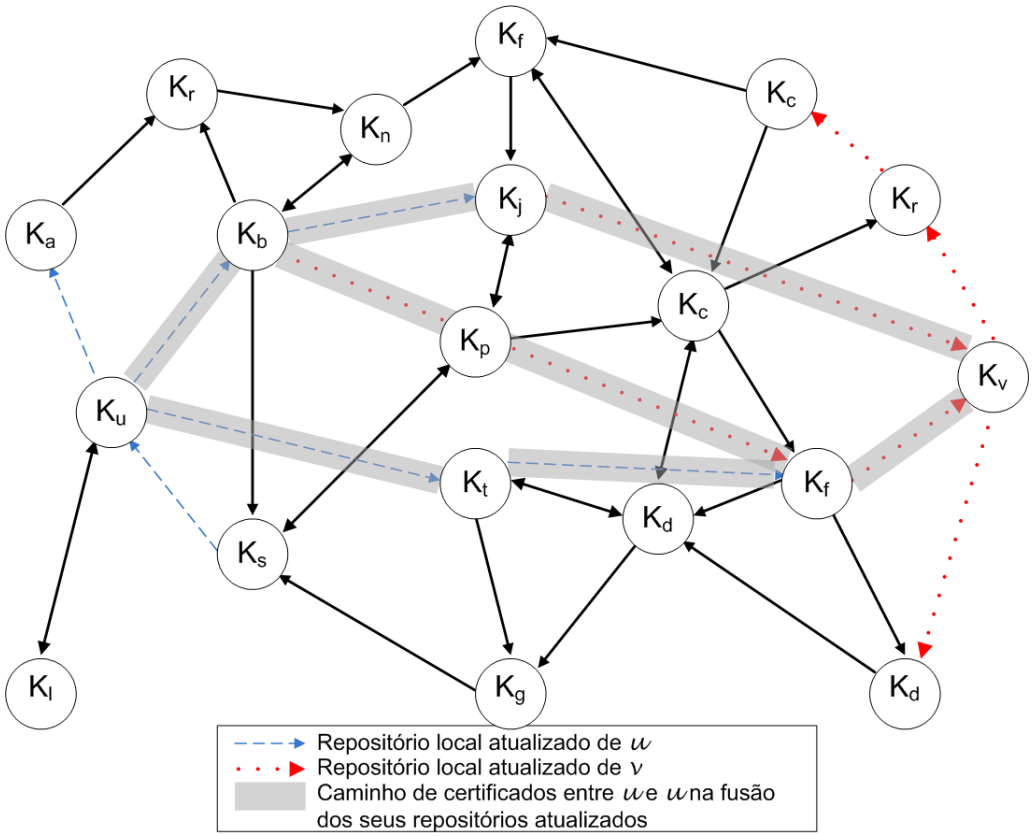

Figura 1. Grafo de certificados e os caminhos de certificados

A Figura 1 apresenta uma visão geral dos certificados válidos de toda a rede. As arestas azuis representam o sub-grafo G_u e as arestas vermelhas o sub-grafo G_v. Quando u tenta autenticar a chave pública K_v de v, ele faz a fusão dos repositórios de certificados atualizados de u e v e encontra um ou mais caminhos válidos para a autenticação. Isso é representado pelas arestas sombreadas do grafo.

Esses repositórios locais descritos anteriormente, são chamados repositórios atualizados, e para o nó u são denotados por G_u, ou grafo de certificados atualizado de u. Cada nó também mantém um repositório local de certificados não atualizados, que são denotados por G_u^N, ou grafo de certificados não atualizados de u.

Quando o nó u tenta verificar a autenticidade de uma chave pública K_v de um usuário v, e não encontra um caminho de certificados na fusão dos repositórios locais de certificados de u e de v, ele procura um caminho de certificados na fusão do seu repositório local de certificados atualizados com o seu repositório local de certificados não atualizados ($G_u \cup G_u^N$).

2.2. A criação e troca dos certificados de chaves públicas

No modelo proposto em [Capkun et al. 2003a] a chave pública de um usuário e sua chave privada correspondente é criada pelo próprio usuário. Similarmente ao PGP, os certifi-

cados de chaves públicas são emitidos pelos usuários. Assim, se o usuário u acredita que uma chave pública K_v pertence a um usuário v, ele então emite um certificado de chave-pública, associando K_v ao usuário v por meio da assinatura de u.

Os certificados são emitidos por um tempo limitado, chamado T_v. Quando os certificados expiram e o seu emissor acredita que a associação do usuário com a chave pública seja ainda válida, ele pode atualizar o mesmo certificado, agora com um novo tempo de expiração.

Em [Capkun et al. 2002] os autores mostram uma análise do uso de grafos de certificados tipo o PGP. O artigo mostra, que devido ao relacionamento social existente entre os usuários da rede, a utilização de grafos de certificados baseados nas características do PGP, no sentido de que o próprios usuários emitem os certificados de chave-pública uns aos outros, possuem um fenômeno definido como mundo pequeno[2].

Cada nó realiza periodicamente uma troca de seus certificados com todos os seus nós vizinhos, como apresentado na Figura 2. Ele envia para os seus vizinhos o seu repositório local de certificados atualizados e não-atualizados. Assim, após uma fase inicial de convergência, chamada de T_{CE}, todos os certificados da rede serão armazenados por todos os nós da rede.

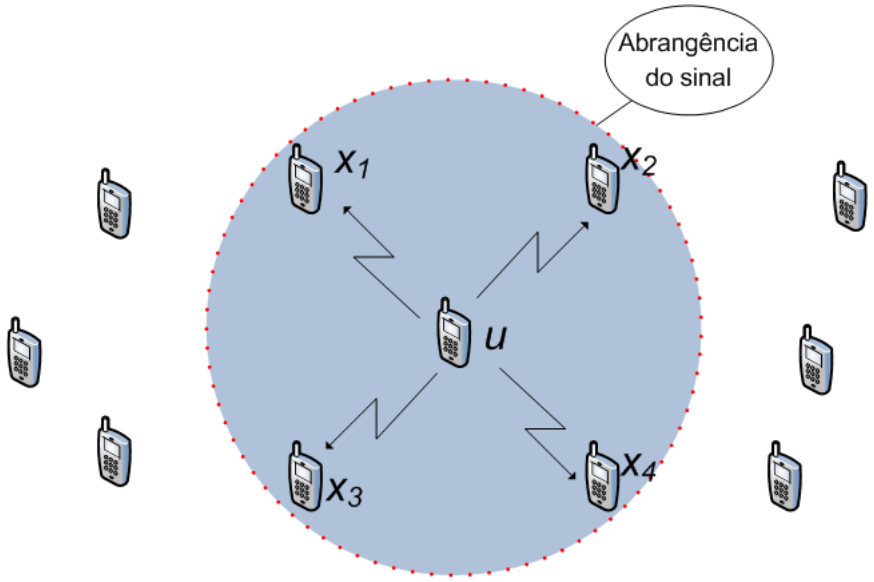

Figura 2. Processo de troca de certificados entre os nós vizinhos

Nessa mensagem de troca, o nó u por exemplo, não envia todos os seus certificados atuais, mas somente uma lista de identificadores únicos. Os nós vizinhos de u que recebem a mensagem, respondem com os valores $hash$ dos certificados que estão em seus repositórios atualizados e não atualizados. O nó u então realiza uma verificação dos valores recebidos e envia para o vizinho somente os certificados que ele ainda não possui ou revalida os certificados que não estão atualizados, ou seja, estão expirados.

[2]Tradução do fenômeno Small World

2.3. Revogação dos certificados

Os certificados podem ser revogados de forma explícita ou de forma implícita. Na revogação explícita, o nó emissor emite um testamento de revogação explícita para todos os nós que solicitaram dele o certificado de chave pública agora revogado. O nó emissor possui uma lista de todos os nós para os quais ele enviou um certificado por ele emitido, o que facilita a revogação dos certificados.

Na revogação implícita, a revogação é baseada no tempo de expiração dos certificados. Caso um certificado expire, e o nó emissor não envie uma revalidação do certificado, então o certificado é revogado implicitamente, e transferido do repositório local de certificados atualizados dos nós para o repositório local de certificados não atualizados dos mesmos nós.

3. O algoritmo PGP-Like e seu funcionamento

O algoritmo foi implementando de forma que todo o mecanismo de gerenciamento de chaves estivesse associado diretamente ao nó móvel, e não a uma aplicação específica. Embora o artigo original tomado como base para a implementação deste esquema não especificasse em qual camada esse gerenciamento deve acontecer, essa decisão foi tomada para permitir uma flexibilidade do esquema e também, a sua utilização com aplicações e protocolos existentes.

Com isso, as principais implementações de código C++ foram realizadas nos arquivos ./common/mobilenode.h e ./common/mobilenode.cc. Esses arquivos implementam o funcionamento da classe mobileNode, responsável pelo gerenciamento dos nós móveis no ambiente NS-2. Também foram necessárias algumas alterações nos arquivos de interface de comunicação entre a linguagem C++ e a linguagem OTcl, entre eles os arquivos ./tcl/lib/ns-lib.tcl e /tcl/lib/ns-node.tcl. Algumas partes do códigos serão apresentadas nessa seção.

A Figura 3 mostra como deve ser instanciado um nó móvel. Deve ser informado, além de outras configurações clássicas, o valor da chave pública do nó móvel. Também é necessário que, para cada nó sejam realizadas: (i) a execução do método start-pgplike (linha 3), onde são inicializadas as estruturas de controle do algoritmo PGP-Like, e; (ii) a associação do nó móvel ao objeto $god_ (linha 4), que pertence a uma classe auxiliar do NS-2 para gerenciamento dos nós móveis.

```
1  for {set i 0} {$i < $val(nn) } {incr i} {
2          set node_($i) [$ns_ node]
3          $node_($i) start-pgplike
4          $god_ new_node $node_($i)
5  }

7  $node_(0) set X_ 5.0
8  $node_(0) set Y_ 2.0
9  $node_(0) set Z_ 0.0
10 $node_(0) set publicKey_ 100
```

Figura 3. Código oTCL de inicialização dos nós no NS-2

Dentre os métodos desenvolvidos para a classe mobileNode, um importante para o funcionamento do modelo é o emissão de certificados. Quando um nó acredita que uma dada chave K_v pertence ao usuário v, ele emite um certificado público para esse usuário. Esse método é apresentado na Figura 4.

```
1  void
2  MobileNode::genCertificate(MobileNode *nodeDst, TclObject*
       nodeParam)
3  {
4    int publicKeyM;
5    int publicCertificateM;
6    double validateTimeM;
7    int publicKeyIssuer =  getPublicKey(this);
8    repositoryNode* rootNode = certificate_repository_->
         repositoryNodes;
9    lVisited* listVisited = new lVisited;
10   if ( (findNodeInRepository(rootNode,nodeDst->address_,
         listVisited)) == NULL ) {
11     fprintf(stderr,"Gerando chave para no %d\n",nodeDst->address_)
           ;
12     publicKeyM = getPublicKey(nodeDst);
13     publicCertificateM = publicKeyM + 10;
14     double now = Scheduler::instance().clock();
15     validateTimeM = now + VALIDITY_PERIOD;
16     addCertificate(nodeDst, this, publicCertificateM,
           validateTimeM, publicKeyIssuer, address_);
17     nodeDst->addCertificate(nodeDst, this, publicCertificateM,
           validateTimeM, publicKeyIssuer, address_);
18     sendCertificate(nodeDst, publicCertificateM, validateTimeM,
           nodeParam);
19   } else {
20     fprintf(stderr, "Certificador jÃ¡ emitido\n");
21   }
22 }
```

Figura 4. Código C++ para a emissão de certificados na classe mobileNode

No código apresentado, são necessários 2 parâmetros: (i) um ponteiro para o nó confiável do que está sendo emitido o certificado, para ser armazenado no repositório de certificados atualizados, e; (ii) um ponteiro para o objeto Tcl associado ao nó confiável, para que o certificado seja enviado ao nó. Antes de emitir o certificado, é necessário saber se esse nó confiável não está na repositório de certificados atualizados do nó. Para isso, é utilizado o método findNodeInRepository (linha 10), que retorna o valor NULL, caso o nó ainda não esteja no repositório.

Depois de realizadas as verificações, é adicionado o certificado gerado ao repositório de certificados atualizados, com o método addCerticate (linha 16) e em seguida, enviado o certificado para o nó confiável, com o método sendCertificate (linha 18).

Outro método importante para o funcionamento do PGP-Like é a troca de certificados entre os nós vizinhos em uma rede. Periodicamente os nós mandam uma mensagem para os seus vizinhos com o seu repositório de certificados atualizados. O nó destino, ao receber essa mensagem contendo o repositório de certificados de seu vizinho, mescla

(*merge*) o seu repositório de certificados com o repositório recebido. Nesse caso, como os repositórios de certificados são representados por um grafo direcionado conexo, essa mescla dos repositório é uma operação de união dos vértices e arestas dos dois repositório. A união do repositório do nó u com o repositório do nó v é representado por $G_u \cup G_v$, que consiste na união dos vértices e arestas dos dois nós. Essa operação de mescla de repositórios é realizada pelo método mergeRepository, onde é informado o repositório que será mesclado com o atual repositório de certificados do nó.

O processo de troca dos certificados deve ser periódico. Atualmente foi desenvolvida uma interface entre o oTCL e o C++, onde dentro do código oTCL pode ser chamado método exchangeCertificates de um nó. Esse método, procura por todos os vizinhos do nó, utilizando a classe auxiliar god e solicita o repositório de certificados desse nó vizinho, como é mostrado na Figura 5

```
1  void
2  MobileNode::exchangeCertificates()
3  {
4    int numNodes = God::instance()->nodes();
5    for (int i=0; i < numNodes; i++) {
6      if (i != address_) {
7        if (God::instance()->IsNeighbor(address_,i) == true) {
8          MobileNode* neighborNode = (MobileNode*)
               get_node_by_address(i);
9          fprintf(stderr, "Node %d exchanging certificate with node
               %d\n", address_, neighborNode->address_);
10         repositoryGraph* repository_neighbor = neighborNode->
               certificate_repository_;
11         repositoryNode* neighborRootNode = repository_neighbor->
               repositoryNodes;
12         lVisited* visited = new lVisited;
13         mergeRepository(neighborRootNode, visited);
14       }
15     }
16   }
17 }
```

Figura 5. Código C++ da interface de troca de certificados de um nó

Outro método importante desenvolvido no PGP-Like é o caminho no repositório de certificados. Esse caminho é utilizado para mostrar todos os certificados emitidos por um nó e também o certificados que foram emitidos por ele. Além disso, depois das trocas de certificados, apresenta também os certificados que foram emitidos por outros nós que, de forma transitiva, possuem alguma relação com o nó móvel. Para apresentar todos os certificados de um repositório de certificados, foi utilizado o método showCertificates, apresentado na Figura 6.

3.1. O algoritmo de grau máximo

Nessa seção é descrito o algoritmo de de construção Grau Máximo[3] para a construção dos repositórios atualizados. Uma descrição mais detalhada do algoritmo pode se encontrada em [Capkun et al. 2003a]

[3]Traduzido de *Maximum Degree*

```
1  void
2  MobileNode::showCertificates(repositoryNode* subRootNode, lVisited
       * visited)
3  {
4    visited->queue(subRootNode->nodeid);
5    repositoryArc* currentArc = subRootNode->listArc;
6    if (currentArc == NULL) {
7      return;
8    }
9    while (currentArc != NULL) {
10     repositoryNode* visitedNode;
11     fprintf(stderr,
12         "Lista de no %d --> No = %d; Certificado = %d; No Emissor:
               %d\n",
13         address_,currentArc->dstNode->nodeid,currentArc->
               publicCertificate,currentArc->issuerId);
14     if (currentArc->dstNode != subRootNode) {
15       visitedNode = currentArc->dstNode;
16       if (visited->listOf(visitedNode->nodeid) == -1) {
17         showCertificates(visitedNode, visited);
18       }
19     }
20     currentArc = currentArc->nextArc;
21   }
22   return;
23 }
```

Figura 6. Código C++ para mostrar todos os certificados de um repositório

O algoritmo Grau Máximo seleciona um sub-grafo de duas partes logicamente distintas: um sub-grafo de entradas e um sub-grafo de saída, ou seja, um sub-grafo que possui os vértices que estão chegando a um nó e outro com os vértices que estão saindo de um nó. Quando inicia a partir do vértice K_u, o algoritmo gera o caminho dos vértices de saída $e_{out} = min(deg_{out}, c)$ e o caminho dos vértices de entrada $e_{in} = min(deg_{in}, c)$, onde:

- deg_{out} corresponde o número de arestas saída de K_u;
- deg_{in} corresponde o número de arestas entrando em K_u, e;
- c é uma constante que representa o número desejável de caminhos a ser construído.

Os tamanhos dos caminhos de entrada e saída, representados por l_{in} e l_{out} respectivamente, são calculados pela função $s/2e_{in}$ e $s/2e_{out}$ respectivamente, sendo que s é uma entrada do algoritmo que representa o número necessário de vértices do sub-grafo resultante.

O algoritmo executa em duas rodadas. Na primeira, inicia do vértice K_u, ou seja, a chave-pública do usuário que está construindo o sub-grafo, e inclui nesse sub-grafo de saída e_{out} as arestas de saída que originaram de K_u, juntamente com seus respectivos vértices associados. Esse conjunto de vértices destino é chamado D_{out}. Na prática, os vértices de saída de um vértice K_u corresponde a todos os certificados emitidos por u. Nos passos seguintes, as aresta e_{out} e seus respectivos vértices de terminação são selecionadas e o processo se repete. Na prática, significa que o nó u pergunta aos nós pertencentes ao

vértices em D_{out} pela lista de suas arestas de saída. A construção do sub-grafo de entrada de u é similar.

4. Considerações finais e trabalhos futuros

O modelo proposto é bastante interessante para as redes *ad hoc* móveis, uma vez que os próprios usuário podem emitir o seus certificados. Diversas etapas do artigo original proposto por [Capkun et al. 2003b] foram implementadas, porém existe ainda um caminho a ser percorrido na definição de métricas, processo de revogação e autenticação dos nós comunicantes.

A definição das métricas e também a implementação de um cenário maior para os testes do algoritmo PGP-Like, serão os próximos passos a serem percorridos, com o objetivo de ter um esquema de gerenciamento de chaves eficiente para MANETs.

Depois de implementado totalmente, diversos estudos e propostas podem ser realizadas baseadas no PGP-Like. Porém alguns pontos podem ser consideravelmente negativos. Para a autenticação de chaves públicas é utilizada uma cadeia de certificados baseados no repositório de certificados atualizados. Esse grafo de certificados, usado para modelar o relacionamento de confiança da rede, pode não ser fortemente conectado [Wu et al. 2005], o que pode ser uma grande problema na comunicação, principalmente em redes com grande índice de mobilidade.

[Buttyan 2006] apresenta duas desvantagens do modelo apresentado em [Capkun et al. 2003a]: primeiro, cada usuário precisa construir o repositório de certificados locais antes de poder utilizar o sistema; segundo, como utiliza cadeia de certificados, assume uma confiança transitiva. Portanto, métrica de reputação e autenticação podem ser estudadas para melhorias no modelo.

Referências

[Buttyan 2006] Buttyan, L. (2006). Mobility helps peer-to-peer security. *IEEE Transactions on Mobile Computing*, 5(1):43–51. Member-Srdjan Capkun and Senior Member-Jean-Pierre Hubaux.

[Buttyan and Hubaux 2003] Buttyan, L. and Hubaux, J.-P. (2003). Report on a working session on security in wireless ad hoc networks. *SIGMOBILE Mobile Computing and Communications Review*, 7(1):74–94.

[Capkun et al. 2002] Capkun, S., Buttyan, L., and Hubaux, J.-P. (2002). Small worlds in security systems: an analysis of the pgp certificate graph. In *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, pages 28–35, New York, NY, USA. ACM Press.

[Capkun et al. 2003a] Capkun, S., Buttyan, L., and Hubaux, J.-P. (2003a). Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64.

[Capkun et al. 2003b] Capkun, S., Hubaux, J.-P., and Buttyan, L. (2003b). Mobility helps security in ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*.

[Hu et al. 2005] Hu, Y.-C., Perrig, A., and Johnson, D. B. (2005). Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38.

[NS-2 2007] NS-2 (2007). The network simulator - ns-2. [Online; acessado em 07-maio-2007].

[Wu et al. 2005] Wu, B., Wu, J., Fernandez, E. B., and Magliveras, S. (2005). Secure and efficient key management in mobile ad hoc networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 17*, page 288.1, Washington, DC, USA. IEEE Computer Society.

[Zhou and Haas 1999] Zhou, L. and Haas, Z. J. (1999). Securing ad hoc networks. *IEEE Network*, 13(6):24–30.

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