Interconectando Clusters com Transparência em Rede de Sensores Sem Fio Segura

Alexandre Gava Menezes[1], Carlos Becker Westphall[2]

[1] Núcleo de Processamento de Dados – Universidade Federal de Santa Catarina (UFSC)
NPD – Campus Universitário – 88040-900 – Florianópolis – SC – Brasil

[2] Laboratório de Redes e Gerência (LRG) – Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Universidade Federal de Santa Catarina (UFSC)
INE – Campus Universitário – 88040-900 – Florianópolis – SC – Brasil

[1]shakal@npd.ufsc.br, [2]westphal@lrg.ufsc.br

Abstract. *A wireless sensor network is a collection of devices limited in low-powered batteries, processing, communication bandwidth capabilities and low memory availability. Due to these constraints, most of security approaches used in wired networks cannot be applied directly in this environment. In this work, we present a hybrid protocol that treats the group key management scheme and the transparent cluster interconnection. It also treats the node-tampering problem, offering a simple solution to control the group key. The feasibility of this protocol was verified by simulation and we concluded that increasing the number of nodes in the network does not change the performance of the interconnection between any two clusters.*

Resumo. *Uma rede de sensores sem fio é uma coleção de dispositivos limitados em poder de processamento e alcance de transmissão, com baixa disponibilidade de memória e de energia. Devido a estas limitações, a maioria das soluções de segurança de redes cabeadas não se aplicam diretamente neste tipo de ambiente. Este artigo apresenta um protocolo híbrido que trata do esquema de gerenciamento de chaves e da interconexão transparente de clusters. Também é tratado o problema de captura de nós, oferecendo uma solução para a proteção da chave de grupo. Simulações mostram que aumentando o número de sensores na rede, o desempenho da comunicação entre dois clusters quaisquer permanece o mesmo.*

1. Introdução

Uma rede de sensores sem fio se caracteriza por ser formada por uma coleção de nós que não utilizam infraestrutura de rede. O baixo poder de processamento, a baixa disponibilidade de memória e o consumo de energia baseado em baterias também são fatores limitantes que devem ser levados em consideração. Neste ambiente de redes de sensores, devido à limitação do alcance da transmissão, para o estabelecimento de comunicação entre dois nós A e B, faz-se necessária a adoção de políticas de múltiplos saltos através dos nós intermediários entre eles. Neste sentido, os nós atuam não só como servidores mas também como roteadores, recebendo e repassando mensagens destinadas a outros nós.

Dentro desta característica, é imprescindível que a comunicação entre os nós seja feita de forma segura, onde alguns requisitos tais como confidencialidade, autenticidade, integridade e não repúdio precisam ser atendidos [Zhou e Haas 1999].

Confidencialidade significa que somente o nó a quem a mensagem está sendo destinada pode ter acesso a informação. Autenticidade diz respeito em poder identificar quem enviou a mensagem. Integridade é a garantia de que a mensagem não tenha sido modificada durante o trajeto. Não repúdio é a característica onde o emissor não pode negar que determinada mensagem tenha sido enviada por ele.

O uso de chaves públicas e suas derivações, utilizadas juntamente com funções hash, garantem as quatro características desejadas na comunicação segura. A garantia de integridade se obtém com o uso de funções hash, confidencialidade com criptografia e autenticidade com assinatura digital. O não repúdio é garantido com o uso de criptografia e assinatura digital. Porém, uma vez que os nós de uma rede de sensores possuem limitações de processamento, de memória e de comunicação, a utilização de uma infraestrutura de chaves públicas (PKI) pura torna-se inviável pelo tempo de processamento e conseqüentemente pelo consumo de energia despendida para cifrar e decifrar as mensagens. Uma maneira de reduzir estes gastos na troca de mensagens é a utilização de chaves simétricas, tanto no descobrimento de rotas quanto na troca de dados.

Uma rede de sensores pode ser dividida em clusters, na tentativa de agrupar localmente os sensores pelas características dos serviços utilizados e providos, a fim de melhorar o desempenho [Bechler et al., 2004]. Neste tipo de abordagem, é bastante útil a adoção de um dispositivo com menos restrições de processamento, memória, transmissão do que os demais sensores do cluster. Este nó é chamado de cluster head (CH) e é normalmente o responsável pelo esquema de gerenciamento de chaves para os demais nós do cluster.

Devido a inviabilidade do uso de criptografia assimétrica em dispositivos limitados, serão utilizadas chaves simétricas compartilhadas por todos os membros de um determinado cluster para a troca de mensagens. Com este tipo de abordagem, os quatro requisitos de segurança citados anteriormente também são assegurados de forma indireta. O uso de funções hash garantem integridade das mensagens. Para os demais requisitos, são consideradas as relações de confiança entre os nós, ou seja, nenhum nó anômalo possui uma chave de grupo. Esta confiabilidade deve ser garantida na hora do estabelecimento da chave simétrica [Hu e Sharma, 2005]. Porém, o uso de chaves de grupo simétricas acarreta dois principais problemas. Um diz respeito a captura de nós, onde uma vez obtida a chave de grupo, todo cluster fica comprometido. O outro problema é a formação de grupos fechados de comunicação, onde somente quem possui a chave de grupo pode se comunicar.

Neste artigo, será apresentado um protocolo híbrido seguro, que garanta o estabelecimento das chaves de grupo, a descoberta de rotas e a troca segura de mensagens. Também será apresentada uma solução para a captura de nós e para a interconexão transparente entre os clusters.

A viabilidade deste protocolo será comprovada por simulação, onde foi montado um ambiente simulando número de clusters variados no Network Simulator – 2 (NS-2), e chegou-se a conclusão que aumentando significativamente o número de nós na rede, o desempenho da interconexão entre dois clusters quaisquer permanece o mesmo.

O foco de discussão na seção 2 será a apresentação dos trabalhos correlatos. Na seção 3 será apresentado o protocolo proposto, definindo o esquema de distribuição de chaves, uma alternativa para a proteção da chave de grupo e também a interconexão transparente de clusters. Na seção 4 serão apresentados o ambiente de simulação, as

simulações feitas e os resultados obtidos. As conclusões a respeito do protocolo proposto, baseadas no resultado das simulações, serão apresentadas na seção 5 do artigo.

2. Trabalhos Relacionados

Os trabalhos relacionados com o escopo deste artigo podem ser divididos em dois grupos. O primeiro trata do gerenciamento de chaves, enquanto o segundo trata do roteamento seguro em redes ad hoc. Vale ressaltar que a maioria das soluções que serão apresentadas foram propostas para redes ad hoc, podendo ser aplicadas a redes de sensores apenas parcialmente, pois nem sempre as limitações que os sensores impõem são levadas em consideração.

Dentro do primeiro grupo, que trata do esquema de gerenciamento de chaves, pode-se citar Hubaux et al. (2001), que propõem um repositório de certificados, onde cada nó possui um conjunto de certificados armazenados. Tanto o seu certificado quanto os certificados dos demais nós da rede são processados localmente. Dois problemas são visíveis. Primeiro, se o número de nós for grande, o tamanho do repositório necessário ultrapassará a capacidade de armazenamento do sensor. Segundo, se poucos certificados forem armazenados no repositório, temos um problema de desempenho que pode afetar o funcionamento da rede como um todo.

Muitos trabalhos abordam a criptografia limiar a qual permite associações de nós para que juntos sejam a fonte de confiança de certificados de chaves públicas. Zhou e Hass (1999) propõem o uso de um esquema limiar para distribuir os certificados de chaves públicas através de nós especializados. Cada nó é capaz de gerar uma parte da assinatura compartilhada. Mas somente após a combinação de n partes é que o certificado válido pode ser obtido Qualquer combinação de n-1 partes não conseguem reconstruir o certificado. O principal problema é o tempo necessário para fazer a combinação das partes para poder montar um certificado válido. Infelizmente, devido às limitações dos sensores, todos estes trabalhos envolvendo exclusivamente criptografia assimétrica não podem ser aplicados diretamente neste tipo de ambiente.

Outros trabalhos foram feitos na tentativa de apresentar uma solução viável para uma rede de sensores, considerando as limitações inerentes de um sensor. Basagni et al. (2001) apresentam uma solução baseada exclusivamente em criptografia simétrica, que provê autenticação de grupo, integridade de mensagens e confidencialidade. Para garantir o sigilo, é proposto ainda um esquema de redistribuição das chaves de tempos em tempos. A desvantagem desta abordagem é que uma vez que o nó é comprometido, o sigilo é revelado, ficando toda a rede comprometida. Os autores propõem uma solução bastante dispendiosa para evitar este tipo de problema, que é a utilização de hardware *tamper resistant*. Segundo Fokine (2002) e Hu e Sharma (2005) esta abordagem é ainda a mais aplicável a rede de sensores, porém, com o problema da captura de nós.

Na tentativa de suprir esta deficiência e ainda prover uma comunicação que não esteja focada na comunicação de grupo ou de broadcast, Eschenauer e Gligor (2002) propõem um esquema de distribuição de chaves baseada num repositório de chaves. Nesta abordagem, cada sensor recebe um subconjunto de chaves simétricas. De acordo com a distribuição dos sensores, espera-se que pelo menos dois sensores vizinhos compartilhem pelo menos uma chave em comum. Um dos principais problemas desta abordagem é que dois nós vizinhos podem não ter uma chave compartilhada e terem que se comunicar por uma rota que envolva diversos nós adjacentes.

O segundo grupo de trabalhos relacionado com este artigo trata do roteamento seguro em redes ad hoc. Perrig et al. (2001) apresentam SPINS, uma arquitetura onde uma estação base acessa os demais nós através de um roteamento baseado na origem. Dois protocolos são apresentados. O SNEP para troca de mensagens entre dois nós e o μTESLA para broadcast, sendo baseados em criptografia simétrica. Nesta arquitetura, a estação base atua como a única entidade de confiança, sendo que cada nó confia somente em si e na estação base, sendo este o fator limitante. Por ser a única entidade de confiança entre os nós, a detecção de rotas depende exclusivamente da estação base. Também a comunicação entre dois nós quaisquer depende da autenticação feita pela estação base.

Zapata e Asokan (2002) propõem uma alteração nas mensagens RREQ e RREP do protocolo Ad hoc on-demand Distance Vector (AODV) [Perkins e Belding-Royer 2003], através do uso de assinaturas digitais para garantir autenticidade e integridade dos campos não mutáveis. Para o campo hop count de ambas as mensagens, é usado uma função hash que permite cada nó que recebe uma mensagem verificar que este campo não foi decrementado por um intruso. Os autores ainda supõem que há um esquema distribuído de autoridades certificadoras(CA's) onde cada sensor teria acesso, a qualquer momento, para poder autenticar uma chave pública de outro sensor na rede ao qual ele queira se comunicar. Vale ressaltar que os próprios autores destacam que esta abordagem não se aplica diretamente a dispositivos com limitações tais como os sensores.

Depois de analisar cada tipo de abordagem, acredita-se que o uso de um protocolo híbrido seja bastante aceitável, uma vez que utiliza a agilidade na transmissão pela utilização de chaves simétricas, bem como utiliza as vantagens de flexibilidade e escalabilidade oferecidas pelo uso de chaves públicas. A utilização de chaves públicas se destaca principalmente para ser aplicada no estabelecimento de chaves de grupo, na proteção do segredo de grupo contra captura de nós e para possibilitar a interconexão transparente de clusters.

Kamal (2004) propõe o uso de chaves simétricas e clusters usando PKI, porém no seu trabalho não é apresentada uma solução consistente nem simulações que comprovem o que foi proposto. O que será apresentado neste artigo possui algumas características parecidas com as que foram abordadas por Kamal no seu trabalho.

3. O Protocolo Proposto

Neste artigo é proposto um protocolo híbrido, que aborda as vantagens de utilização de chaves simétricas para descobrimento de rotas e para a troca de mensagens entre sensores e chaves públicas para o estabelecimento destas chaves simétricas, bem como para a interligação entre clusters.

Como a utilização de uma entidade confiável centralizada para desempenhar o gerenciamento de chaves públicas não é possível num ambiente de redes ad hoc, torna-se necessário encontrar uma solução distribuída [Bechler et al., 2004]. Para viabilizar a utilização de chaves públicas em uma rede de sensores, é preferível que somente um nó em cada cluster utilize uma CA para autenticar as chaves públicas recebidas. E este nó deve ser o cluster head, devido as suas características menos restritas. Desta maneira, torna-se muito menor o número de CA's distribuídas necessárias para garantir a disponibilidade desejada para toda a rede. Para não desviar o foco deste artigo, será assumido que existe uma distribuição de CA's de modo que a qualquer momento, um

cluster head pode verificar a autenticidade da chave pública de outro nó com quem ele deseja comunicar-se. Uma outra solução para validação de chaves públicas poderia ser a proposta por Du et al. (2005), que armazena *hash* das chaves nos nós.

Para viabilizar o uso de um protocolo criptográfico, os sensores estabelecerão com o cluster head uma chave de sessão (simétrica) e a partir desta chave obtida, todas as operações de descoberta de rota e de troca de mensagens serão feitas utilizando esta chave de grupo. Nesta abordagem, o cluster head torna-se a entidade responsável pelo gerenciamento e distribuição destas chaves. Neste artigo considera-se que o problema de captura de nós não se aplica ao cluster head. Para os demais sensores, será proposto um esquema de garantir que a captura dos sensores não seja um problema crucial. A seguir, será apresentado o esquema de gerenciamento de chaves proposto e em seguida uma solução através deste protocolo para garantir a proteção da chave simétrica e depois a interconexão transparente de clusters.

3.1. O Esquema de Distribuição de Chaves

Antes de iniciar especificamente o esquema de distribuição de chaves, será apresentado o dispositivo que será adotado para representar a configuração dos sensores deste artigo. O Mica2 Mote foi desenvolvido na Universidade de Berkeley e fabricado pela Crossbow Inc. O sistema operacional é o TinyOS, desenvolvido também pela Universidade de Berkeley e é um sistema de código aberto exclusivo para sensores. Dentro das características deste dispositivo pode-se destacar o processador de 8-bits com 7.3828 MHz Atmega 128L, sua memória primária (SRAM) de 4 Kb e 128Kb de espaço para os programas (ROM). Este dispositivo ainda possui uma EPROM de 512 Kb e transmissão a 433-MHz de rádio freqüência, com taxa de transmissão de 38.4K.

O esquema de distribuição de chaves obedece algumas premissas básicas. Cada sensor ao ser inserido na rede, possui um conjunto de dados instalados previamente por uma entidade de confiança tanto do sensor quanto do cluster head. Estes valores são um identificador próprio (IDsns) e o par de chaves, uma chave privada (KRsns) e outra pública (KUsns). Ainda são pré-instalados no sensor o identificador (IDch) e a chave pública (KUch) do cluster head ao qual ele vai pertencer. Esta abordagem se justifica por dois principais motivos. Primeiro, para o sensor é muito dispendioso gerar o par de chaves. De acordo com Malan et al. (2004), o custo para gerar o par de chaves de 163 bits utilizando Criptografia de Curvas Elípticas (ECC) é de aproximadamente 34 segundos, utilizando a plataforma Mica2 Mote. Segundo que caso a chave pública do cluster head (KUch) não fosse instalada previamente no sensor, o custo para o estabelecimento de uma relação de confiança entre o sensor e o cluster head seria maior ainda. A distribuição de CA's para que todos os sensores pudessem ter validada uma chave pública recebida seria quase que inviável pelo alcance de transmissão do sensor. Mesmo que esse problema fosse superado, ainda deve ser levado em consideração o modo como a chave de grupo seria estabelecida, pois um dos métodos mais difundidos, o de Diffie-Hellman, é demasiadamente dispendioso para o sensor. Para se ter uma idéia, Malan et al. (2004) estima que o tempo de troca de chaves usando Diffie-Hellman é de aproximadamente 68 segundos. Recentemente, Blaß e Zitterbart (2005) apresentaram resultados bastante otimistas sobre a implementação eficiente dos pontos de multiplicação dos algoritmos de curvas elípticas. Todo o processo de geração de chaves e de troca de chaves por Diffie-Hellman, passou de aproximadamente 102 segundos [Malan et al. 2004] para perto de 24 segundos. Mesmo assim, na abordagem deste artigo, evita-se que o sensor já tenha este gasto prévio, tanto de tempo de

processamento quanto de energia de suas baterias. As mensagens usadas para o estabelecimento de chaves são:

Sensor → CH: (tipo, IDch, IDsns, KUsns, Assinatura) (1)

CH → Sensor: (tipo, IDsns, IDch, EKUsns(Ks), Assinatura) (2)

CH1 → CH2: (tipo, IDch2, IDch1, IDsns, KUsns, IDsns1, Assinatura sns, Assinatura ch1) (3)

CH2 → CH1: (tipo, IDch1, IDch2, IDsns, Assinatura ch2) (4)
Assinatura ch2 = Assinar KRch2(HASH(type, IDsns, IDch2, IDch1, KUch1, IDsns1))

CH1 → Sensor: (tipo, IDsns, IDch2, IDch1, KUch1, IDsns1, Assinatura ch2) (5)

Sensor → CH: (tipo, IDch, IDsns,Assinatura gr) (6)
Assinatura gr = EKs(Hash)

Sendo (1) a solicitação de ingresso no cluster visando a obtenção da chave de grupo. (2) é a resposta de ingresso no cluster, com o repasse da chave de grupo. (3) é o encaminhamento para o cluster head destino de uma solicitação de ingresso no cluster. (4) é a resposta de reconfiguração do sensor para reconhecer o novo cluster head. (5) é a mensagem de reconfiguração do sensor para pertencer a um novo cluster head. (6) é a mensagem de solicitação de uma nova chave de grupo.

Um novo nó ao ser inserido, solicita ao cluster head a inserção efetiva no cluster, ou seja, receber a chave de grupo para poder se comunicar secretamente com seus vizinhos. Para isso, ele monta uma requisição que tem o formato (1), onde a Assinatura é uma função de criptografia utilizando a chave privada do sensor, sobre o resultado da aplicação de uma função hash sobre os quatro primeiros campos da mensagem. Cada nó ao receber este tipo de mensagem, simplesmente repassa, sem descartá-la, em direção ao cluster head, pois somente o cluster head pode tratá-la. O cluster head ao receber uma requisição de ingresso no cluster, primeiro verifica se a requisição está destinada para ele ou para outro cluster head. Abaixo será tratado o caso da mensagem ser destinada para o próprio cluster head. O caso de endereçamento da requisição para outro cluster head será tratado adiante.

Após o cluster head receber uma solicitação de inclusão e verificar que a mensagem é destinada a ele, verifica a autenticidade da mensagem usando a chave pública do sensor. A integridade é verificada aplicando a mesma função hash sobre os quatro primeiros campos da mensagem e comparando com o hash que foi obtido na verificação da assinatura.

Para responder a esta requisição e enviar para o sensor a chave de grupo (Ks) para comunicação, o cluster head seleciona a chave de grupo atual – caso ainda não tenha, gera uma nova – e a envia para o sensor com o seguinte formato (2). De posse de Ks, o cluster head cifra Ks usando a chave pública do sensor (EKUsns), de modo que somente o sensor ao qual a mensagem está destinada poderá decifrar utilizando a sua chave privada. Após cifrar a chave de grupo, basta juntar os quatro campos (tipo,IDsns,IDch,EKU(Ks)), aplicar uma função hash sobre os valores e assinar o hash, para garantia de autenticidade e integridade. Antes de enviar a resposta para o sensor, o cluster head registra o ID do sensor e a chave que foi distribuída numa tabela que relaciona IDs a chaves de grupo. O funcionamento desta tabela será explicado melhor na seção 3.2.

Devido à abrangência de transmissão do cluster head, as mensagens originadas por ele não precisam ser repassadas através dos nós vizinhos, podendo ser descartadas caso o sensor verifique que o destinatário da mensagem não seja ele.

Ao receber esta mensagem, o sensor de destino verifica a assinatura utilizando a chave pública do cluster head que está armazenada em sua memória e depois compara o *hash* dos quatro primeiros campos, com o que foi obtido na verificação da assinatura. Uma vez comprovada a autenticidade e a integridade dos dados, o sensor decifra a chave simétrica utilizando a sua chave privada e assim obtém a chave de grupo, para poder começar a participar efetivamente do cluster. Este momento de inicialização é o que despende maior tempo de processamento, comunicação e energia para o sensor. Uma vez obtida a chave de grupo, toda a comunicação dentro do cluster é feita utilizando criptografia simétrica.

Quando o cluster head CH1de um determinado cluster C1 recebe uma solicitação do tipo (1) e verifica que o endereço de destino não é igual ao seu, ele encaminha uma mensagem (3) para o cluster head destino (CH2) do cluster (C2), comunicando que há um sensor, que deveria pertencer ao cluster C2, solicitando ingresso no cluster C1. Em vez de CH1 fazer a verificação da assinatura de (1), ele incorpora a assinatura em (3), juntamente com o IDsns do sensor solicitante e o novo ID que o sensor receberá (IDsns1), assina a mensagem, e envia para CH2. Ao receber a mensagem do tipo (3), CH2 primeiro verifica a autenticidade e a integridade da mensagem. Depois, ele monta a mensagem utilizando os campos ID ch2 e IDsns, juntamente com a assinatura enviada pelo sensor e o tipo correspondente a uma mensagem de ingresso no cluster (1). Feito isso, a assinatura do sensor pode ser verificada, assim como a integridade da mensagem.

Antes dessa requisição ser atendida, o sensor precisa ser reconfigurado para identificar CH1 como seu novo cluster head. Para isso, quem deve mandar uma mensagem de reconfiguração é CH2, o qual neste momento é ainda o único com quem o sensor mantém uma relação de confiança, ou seja, consegue verificar assinatura, por ter sua chave pública e IDch pré-instalados. Para gerar a mensagem de reconfiguração, o cluster head monta (4), com uma particularidade. A assinatura é feita como se os valores de IDch1 e KUch1 estivessem na mensagem. Depois, como CH1 possui sua identificação e sua chave pública, não é necessário envia-los para CH1. Esta assinatura é importante, pois será a assinatura que o sensor reconhecerá quando a mensagem de reconfiguração (5) chegar até ele. Quando CH1 for repassar a mensagem para o sensor, ele simplesmente remonta a mensagem (4), colocando o destinatário como CH2, bem como o IDch1 e KUch1, que serão os dados do novo cluster head que o sensor participará. Como a assinatura já foi feita em (4), CH1 simplesmente coloca a mesma assinatura e repassa (5) para o sensor. Desta forma, é possível verificar a autenticidade e a integridade. Depois de tratar a mensagem, o sensor terá posse de IDch1 e KUch1, podendo assim fazer uma solicitação inicial de chave de grupo(1), como já foi explicado anteriormente. No entanto, para total realocação e reconfiguração do sensor duas medidas precisam ser tomadas. Uma, quando o CH2 repassa a mensagem de reconfiguração para CH1, ele deve também informar a CA o novo ID que o sensor terá para fazer a relação de KUsns e IDsns. A outra medida a ser tomada é quando o sensor receber a mensagem de reconfiguração, ele precisa mudar o seu IDsns, pois o seu endereço no novo cluster será diferente do antigo.

Geralmente, um sensor solicita somente uma vez a inclusão no cluster. Outra solicitação de inclusão pode ocorrer somente em três casos. Primeiro se o sensor se

move para outro cluster, onde os sensores a sua volta trocam mensagens criptografadas utilizando outra chave de grupo. Esta situação será abordada a seguir. O segundo caso de solicitação de inclusão ocorre se o sensor estiver sob ataque, onde não consiga decifrar as mensagens que está recebendo, usando sua chave simétrica. O terceiro caso ocorre, se a bateria do sensor já estiver com nível baixo de energia, onde numa oscilação os dados que estavam em memória podem ser perdidos. Neste caso, porém, o cluster head pode ignorar este sensor, pois a chance de falhar é extremamente alta.

Quando um sensor que já possui chave de grupo move-se de um cluster para outro, torna-se necessário a obtenção de uma nova chave de grupo. Deste modo, o sensor vai precisar enviar uma mensagem de inclusão no cluster do tipo (6), que difere-se de (1) somente porque a assinatura é cifrada com a chave de grupo que o sensor possui naquele momento, passando a ser uma assinatura de grupo e não individual. Assume-se, porém, que pela relação de confiança estabelecida na obtenção da chave de grupo, confia-se que o sensor seja mesmo quem ele diz ser. Os passos seguintes são basicamente os mesmos dos apresentados anteriormente quando um sensor é inserido num outro cluster. A única diferença é que o cluster head designado para verificar a assinatura terá que decifrar usando a mesma chave de grupo que está na tabela que relaciona ID de sensor com chaves de grupo distribuídas. E também que a assinatura do cluster head será feita utilizando a mesma chave de grupo que o sensor em questão já possui. O uso desta tabela de referência é explicado melhor no item abaixo.

Em todos os casos anteriores, se a integridade ou a autenticidade de uma mensagem não possa ser comprovada, o nó que recebeu descarta a mensagem.

3.2. Garantindo a Proteção da Chave de Grupo

Como a chave simétrica é na verdade uma chave de sessão, estabelecida somente uma vez e sendo mudada de acordo com a política de troca de chaves adotada, os sensores tem que se preocupar somente em proteger sua memória RAM para que não se possa fazer um *dump* nela. Ao invés de utilizar um hardware *tamper resistant*, adota-se o princípio de que ao ser capturado e aberto, o sensor tenha sua fonte de energia cortada e conseqüentemente, os dados de sua memória RAM apagados, que significa uma saída para o principal problema da adoção de chaves de sessão. Somente os dados gravados em disco podem ser acessados, que são o ID do sensor, sua chave privada, o ID do cluster head e sua chave pública.

Para evitar que um estranho possa obter novamente a chave de grupo utilizando as primitivas de inclusão no cluster, pois possui a chave privada e o ID do sensor, o protocolo proposto possui um mecanismo de detecção deste tipo de problema. Toda vez que o cluster head receber uma requisição de chave de grupo, ele deve consultar uma tabela que guarda os IDs dos sensores que já receberam a chave de grupo. Nesta tabela também consta a chave de grupo que este sensor está utilizando. Caso já haja uma entrada, o sensor só poderá solicitar uma chave usando a primitiva (6), da sessão anterior. O uso de uma tabela de referência entre ID e chaves de grupo é útil também para a redistribuição de uma nova chave, que deve ser feito dentro de um tempo estipulado. O cluster head tem o controle de cada sensor que já recebeu e dos que faltam receber a nova chave de grupo.

3.3. Interconectando Clusters com Transparência

Duas das principais críticas sofridas pela abordagem que está sendo feita neste artigo diz respeito ao problema da captura de nós, que já foi explicado anteriormente como

evitar e o outro problema do uso de chaves simétricas em clusters, segundo Zapata e Asokan (2002), é que a comunicação fica restrita a um grupo fechado de participantes – os que detêm a chave simétrica.

Recentemente, a criptografia de curvas elípticas vem ganhando um crescente destaque na área de dispositivos limitados como sensores, em relação a outras técnicas de criptografia como o RSA (Rivest-Shamir-Adleman). Dois fatores justificam esta importância. Primeiro o tamanho das chaves. Lenstra e Verheul (2001) e Gura et al. (2004) afirmam que uma chave de 160 bits no ECC oferece uma segurança equivalente a uma chave de 1024 bits com o RSA. Já uma chave de 224 bits no ECC equivale a uma de 2048 no RSA. Segundo, que como suas chaves são menores, espera-se que o tempo de processamento e a energia despendida para cifrar e decifrar sejam menores que os do RSA. Não só por isso, mas também pela própria natureza da criptografia ECC que opera sobre um conjuto de pontos numa curva elíptica definida num universo finito. Sua principal operação criptográfica é a multiplicação escalar sobre um ponto ao invés da empregada no RSA que são operações exponenciais modulares de números inteiros bastante grandes.

Diversos artigos foram publicados no sentido de destacar a aplicabilidade da criptografia de curvas elípticas no ambiente de redes de sensores, dando destaque ao tempo de processamento e consumo de energia reduzidos. Porém, o sucesso do uso de chaves assimétricas num ambiente de redes de sensores vai depender do número de vezes que as chaves de sessão serão geradas pelos acordos de trocas de chaves entre os nós fins, como por exemplo, o acordo de Diffie-Hellman [Malan, 2004].

A preocupação deste artigo é evitar a sobrecarga de processamento e consumo de energia nos sensores e fazer com que somente o cluster head seja encarregado de fazer estes acordos com os cluster heads de outros clusters. Desta maneira, o tempo de processamento será muito menor se uma chave de sessão for estabelecida para a comunicação entre dois sensores de dois clusters quaisquer.

Quando um sensor deseja comunicar-se com outro dentro do seu cluster, ele simplesmente o faz usando a chave de grupo. Tanto o descobrimento de rota quanto a troca de dados propriamente dita é feita com esta chave, que possibilita operações criptográficas muito mais ágeis, se comparadas com chaves assimétricas. Quando o sensor que ele deseja comunicar-se está em outro cluster, uma abordagem diferente precisa ser utilizada. Todas as mensagens endereçadas a sensores em outros clusters são encaminhadas para o cluster head do seu cluster. Este por sua vez, decifra a mensagem utilizando a chave de grupo e após verificar o endereço do cluster que a mensagem deve ser encaminhada, procura estabelecer uma rota com o sensor destino, contatando o cluster head destino. Como resposta da rota, o cluster head destino propõe uma chave de sessão, juntamente com a mensagem de resposta de rota que ele envia, dando início ao acordo de troca de chaves, seguindo o modelo de Diffie-Hellman. De posse da chave pública do cluster head destino, o cluster head de origem efetua a multiplicação da chave pública recebida com a sua chave privada e decifra a chave de sessão que lhe foi enviada. Antes, porém, o cluster head já encaminha a resposta de rota para o sensor, usando a chave de grupo daquele cluster.

A partir deste momento, todo o percurso da mensagem será feito usando chaves de sessão. Primeiro, o sensor cifra a mensagem de dados usando a chave de grupo. O cluster head ao receber, decifra com a chave de grupo e cifra com a chave de sessão que foi estabelecida com o outro cluster head. O cluster head destino ao receber a

mensagem, decifra usando a chave de sessão e cifra usando a chave de grupo do cluster destino. A última etapa consiste em enviar a mensagem para o sensor destino e este de posse da chave de grupo do cluster, decifra a mensagem que lhe foi destinada.

A principal vantagem desta abordagem se dá quando qualquer outro sensor de um dos dois clusters tentar se comunicar com outro sensor do outro cluster, ele simplesmente precisará descobrir a rota usando a chave de grupo e em seguida enviar as mensagens de dados criptografadas com a mesma chave de grupo. Todo o percurso da mensagem será otimizado pois o processo mais dispendioso já foi feito anteriormente, ou seja, já houve o acordo de troca de chaves entre os cluster heads e uma chave de sessão já foi estabelecida. Até mesmo o estabelecimento de rotas é feito de maneira otimizada, pois de posse de chave de sessão entre os cluster heads, a assinatura pode ser feita usando esta chave, uma vez que somente os dois cluster heads a possuem. Claro que para isso supõe-se que a chave de sessão entre os cluster heads ainda não tenha expirado.

Desta maneira, tem-se uma interconexão de clusters, feita de maneira transparente entre os fins, que são os sensores. Um sensor não precisa saber em qual cluster está o destino com que ele necessita trocar mensagens. Ele simplesmente o faz, como se todos os sensores estivessem dentro do mesmo cluster. Esta abordagem será a base de nossas simulações, onde se quer demonstrar que interconectando dois ou mais clusters, o desempenho na troca de mensagens será o mesmo, tendo apenas o acréscimo do tempo de propagação de mensagem pelos nós adjacentes.

4. Simulações

As simulações tiveram seu foco principal no desempenho do protocolo proposto para a interconexão de clusters, sendo necessário a montagem do ambiente em um software de simulação. Devido a grande aceitabilidade no meio científico, por ter seu código totalmente aberto, que possibilita a modificação e extensão de certas funcionalidades o Network Simulator 2 (NS-2) foi o simulador escolhido. O NS-2 é um simulador de eventos discretos orientado a objetos, em constante aprimoramento e desenvolvimento. É escrito em C++, com um interpretador OTcl (Object Tool Command Language), unindo o alto desempenho da linguagem C++ à facilidade de alterações na configuração das simulações, permitidas pela OTcl. Outra característica é que o NS-2 permite simulações de ambientes não só cabeados mas também sem fio, que é o escopo deste trabalho.

Como a base da solução apresentada deve ser implementada na camada de roteamento, foi utilizado o protocolo AODV que já vem implementado no NS-2, introduzindo apenas as mudanças necessárias para que a implementação satisfizesse os requisitos de segurança do SAODV (Secure AODV). A seguir serão apresentadas uma visão geral do funcionamento do protocolo AODV e as principais características das extensões do SAODV. Posteriormente será detalhado as mudanças necessárias para o funcionamento do SAODV no ambiente que foi simulado. No final desta sessão, serão apresentados os resultados obtidos com as simulações, que comprovam as idéias defendidas neste artigo.

4.1. O Protocolo AODV e as Extensões do SAODV

O protocolo de roteamento AODV [Perkins e Belding-Royer 2003] é um protocolo reativo, que atua sob demanda. Ao contrário dos protocolos pró-ativos, onde o nó

origem já possui o caminho completo até o nó destino na sua tabela de roteamento, os protocolos sob demanda fazem o descobrimento de rotas somente quando há necessidade de comunicação entre dois nós.

A descoberta de rotas se dá através da inundação da rede com pedidos de rota por difusão (RREQ - Route Requests) iniciando com o nó que deseja iniciar a transmissão. Este processo é feito por todos os nós vizinhos que simplesmente repassam a mensagem recebida para os seus vizinhos, e assim por diante, até que o nó destino receba esta requisição ou um nó intermediário que tenha uma rota já estabelecida na sua tabela de roteamento. Durante o processo de repasse de RREQs, um nó intermediário armazena na sua tabela de roteamento o endereço do vizinho que lhe enviou, por difusão, o primeiro pacote RREQ recebido. Deste modo é estabelecido o caminho reverso. As demais cópias que este nó receber da mesma requisição são descartadas. Quando o RREQ alcança o nó destino ou um nó intermediário que contém a rota, este nó envia uma mensagem de resposta de rota (RREP – Route Response) para o vizinho que lhe enviou primeiro o RREQ, o qual repassa o RREP para os nós precurssores do mesmo RREQ, até que chegue ao nó de origem da solicitação de rota.

A manutenção das rotas é feita através de envios periódicos de mensagens (HELLO) por difusão para os nós vizinhos. Quando um nó se move, ele precisa reiniciar a descoberta de rota para atingir um nó destino. Por outro lado, quando um nó que faz parte de uma rota se move, seu vizinho antecessor no envio da mensagem, irá notar da quebra do link e propagar uma mensagem de erro (RERR – Route Error) para cada um dos nós anteriores que participam daquela rota. Desta maneira, o nó de origem pode iniciar um novo descobrimento de rota.

Zapata e Asokan (2002) propuseram extensões de segurança para proteger as mensagens de roteamento originais do AODV (SAODV). O SAODV usa assinaturas digitais para autenticar os campos não mutáveis e funções hash para autenticar as mensagens RREQ e RREP. Pelo fato dos dois únicos campos mutáveis destas duas mensagens serem o número de saltos e o campo HASH, os demais campos podem ser assinados e suas integridade e autenticidade podem ser verificadas fim a fim. Como os nós intermediários alteram o valor do número de saltos, este não pode ser fixo e incluído na assinatura. Também não pode ter uma assinatura somente para o campo número de saltos, necessitando ser verificada a cada salto, pois tomaria um tempo muito precioso no descobrimento de uma rota. Para contornar este problema, é utilizada uma função hash. Durante o processo de descobrimento de rota, o nó de origem seleciona um número randômico que será utilizado como semente. Inicializa o campo Max_Hop_Count com o valor do campo TimeToLive do cabeçalho IP. Inicializa o valor do campo Hash com o valor da semente gerada. Calcula o valor Top_Hash, aplicando a função hash x vezes na semente, onde x é o Max_Hop_Count. Esta abordagem é feita para que nenhum intruso decremente o valor do número de saltos da mensagem, fazendo com que ela tenha um comportamento diferente do esperado. Quando os nós intermediários recebem a mensagem, eles aplicam a função hash y vezes, onde y é determinado pela subtração dos campos Max_Hop_Count e Hop_Count. O resultado desta função é comparado com o conteúdo do campo Top_Hash e assim comprovada a integridade da mensagem. Antes de difundir um RREQ ou repassar um RREP, o nó intemediário incrementa o valor do campo Hop_Count em uma unidade e computa o novo valor do Hash, aplicando a função hash sobre o campo Hash. (i.e., hash(Hash)).

Apesar do trabalho de Zapata e Asokan ter sido proposto como uma RFC (Request for Comments) e conseqüentemente evoluído bastante desde 2002, a primeira

proposição do SAODV é a que será utilizada neste artigo, com apenas a inclusão da transmissão da chave pública do emissor da mensagem. Isto se deve pelo fato de que a solução proposta pela RFC é bastante genérica, sendo aplicada a qualquer tipo de rede ad hoc, diferente do que está sendo tratado neste artigo, que diz respeito somente a uma rede de sensores.

4.2. Adequando o Protocolo AODV e as Extensões SAODV para Clusters

Algumas adequações são necessárias ao protocolo AODV para funcionar de acordo com as proposições deste artigo. Primeiro, a inclusão de uma função criptográfica simétrica, no caso a escolhida foi o RC5, para a descoberta de rota dentro do cluster. Este protocolo AODV modificado é denominado de SAODV-I. Ao invés de utilizar a mesma solução proposta no SAODV, optou-se por utilizar as mesmas mensagens do AODV, só que antes de serem enviadas, são cifradas com a chave de grupo. Do mesmo modo, assim que são recebidas, antes de serem tratadas, as mensagens são decifradas pelo receptor. Esta abordagem foi adotada por ser a mais eficiente para descoberta de rota dentro do cluster. Segundo Ganesan et al. (2003), o custo para se aplicar uma função hash MD5 sobre 26 bytes é de 1,473 milissegundos e sobre 62 bytes é de 2,722 milissegundos. Já se for considerado o caso de uma função hash SHA-1, o custo para aplica-la sobre 56 bytes é de 3,636 milissegundos e sobre 64 bytes é de 7,777 milissegundos. Ganesan et al. (2003) também mostra que o custo para aplicar uma cifragem do RC5 sobre 16 bytes é de 0,413 milissegundos, portanto, 1,652 milissegundos se forem considerados 64 bytes. O tempo para decifrar é um pouco menor, cerca de 1,636 para 64 bytes de dados.

A segunda adequação necessária no protocolo AODV foi a inclusão das funções criptográficas necessárias para implementar as extensões propostas no SAODV. Este protocolo modificado será utilizado exclusivamente para a comunicação entre os clusters, sendo denominado de SAODV-E.

Ambos protocolos SAODV-I e SAODV-E tiveram adaptações nas suas mensagens em relação às mensagens originais propostos no AODV e nas extensões do SAODV. O SAODV-I teve a inclusão de uma nova mensagem, chamada de IREQ (Iniciador de Requisição) que possui o seguinte formato:

IREQ = (tipo, IDch, IDsns)

Isto se deve ao fato de que o protocolo AODV não funcionaria adequadamente caso o cluster head destino, ao receber um pedido de rota (RREQ) do SAODV-E, solicitasse internamente um RREQ do SAODV-I. Todos os nós receberiam a mesma mensagem, devido a sua abrangência de transmissão e isso definitivamente afetaria na construção das tabelas de roteamento dos nós, bem como na criação da mensagem de resposta de rota RREP, criada pelo nó destino. Para contornar este problema, o cluster head sinaliza para o sensor destino solicitando que ele inicie a descoberta de rota até o cluster head. Esta sinalização é o envio da mensagem IREQ, que contém o ID do cluster head e o ID do sensor destino. Desta forma, o sensor destino inicia uma descoberta de rota até o cluster head, fazendo difusão de RREQs. Ao receber o RREQ, o cluster head já sabe a qual distância (número de saltos) se encontra o sensor e sabe que existe uma rota até ele. Ao responder esta solicitação de rota (RREP), apesar da abrangência de transmissão, o cluster head direciona a mensagem para o sensor mais próximo, ou seja, aquele que lhe enviou a RREQ. Enquanto a mensagem RREP do SAODV-I é encaminhada para o sensor destino e as tabelas de roteamento são ajustadas e definidas, o cluster head dá continuidade com o estabelecimento de rota com o cluster head de

origem, fazendo a multiplicação de Diffie-Hellman e enviando uma reposta de rota (RREP) do SAODV-E. É importante ressaltar que assim como todas as mensagens do AODV-I, a mensagem IREQ também é inteiramente cifrada com a chave de grupo do cluster.

Já as mensagens que fazem parte do SAODV-E, que tiveram mudanças em relação ao SAODV são:

RREQ = (tipo, tamanho, num máx saltos, top_hash, chave pública, assinatura, hash)

RREP = (tipo, tamanho, num máx saltos, top hash, chave pública, chave cifrada, assinatura, hash)

As duas principais diferenças das extensões do SAODV-E em relação ao SAODV proposto por Zapata e Asokan (2002) é que está sendo incluída na mensagem a chave pública do originador da mensagem. No caso da RREQ, a chave pública do cluster head de origem. Para a RREP, a chave pública do cluster head destino. Outra diferença é que na mensagem de resposta RREP, já é enviada uma chave de sessão cifrada com a chave gerada pela multiplicação de Diffie-Hellman, como já foi explanado na sessão 3.3 deste artigo. Resultados preliminares das simulações acusaram um tempo excessivamente alto para a computação das funções hashs com intuito de garantir a integridade do campo número de saltos. Com base nestes resultados, foi preferido o envio das requisições e respostas de rota (RREQ e RREP) antes de verificar a integridade tanto da assinatura quanto do campo número de saltos. Caso um campo tenha sido alterado maliciosamente, o nó receptor ao verificar que a integridade da mensagem foi violada, simplesmente descarta a mensagem, sem utilizar os valores da mensagem para construção da tabela de rotas. Devido à utilização desta abordagem, um mecanismo de detecção de intrusão diferente deve ser adotado. Como não é o enfoque deste trabalho, supõe-se que há um mecanismo desenvolvido para tratar este tipo de anomalia.

Outro fator que deve ser considerado é que as mensagens das extensões do AODV com assinaturas duplas também não foram implementadas.

4.3. O Ambiente Simulado

O ambiente que se quis simular foi de uma rede de sensores dividida em diversos clusters, sendo que a cada momento, se queria estabelecer uma troca de mensagens entre dois sensores quaisquer situados em clusters diferentes, sem que houvesse qualquer rota pré-estabelecida entre eles. Os dados foram gerados por um agente CBR (Constant Bit Rate), que gera mensagens constantes de dados dentro de um intervalo de tempo. O tamanho destas mensagens foi de 512 bytes. Assumiu-se que cada cluster possuía 250 sensores e estes foram distribuídos uniformemente dentro do cluster. Neste trabalho, não foi considerado mobilidade entre os sensores. Primeiro, se utilizou dois clusters vizinhos, e foram levantados os tempos gastos para a troca de mensagens desde o estabelecimento da rota, passando pelo acordo de Diffie-Hellman entre os cluster heads, finalizando com a troca de dados propriamente dita.

Na segunda etapa, foram adicionados cluster heads intermediários para fazer o repasse das mensagens do SAODV-E e os tempos obtidos foram adicionados aos tempos obtidos para a troca de mensagens entre dois clusters vizinhos.

Devido ao modo como o NS-2 funciona, incluindo atrasos no tempo de simulação para cada operação relevante ao que se quer simular, foi necessário um embasamento em outros trabalhos anteriores que apresentam resultados de desempenho das funções criptográficas utilizadas neste artigo. Um dos trabalhos foi o de Ganesan et

al. (2003), cujo os valores relevantes ao presente artigo já foram citados no início da sessão 4.2. Outro trabalho que contribuiu para a obtenção dos resultados de simulação foi apresentado por Blaß e Zitterbart (2005). Segundo os autores, o tempo para assinar uma mensagem usando o algoritmo de assinatura ECDSA, baseado em ECC é de 6,88 segundos e de 24,17 segundos para que esta assinatura possa ser verificada. Os autores ainda demonstram que o tempo para a troca de chaves utilizando o algoritmo de Diffie-Hellman é de 17,28 segundos, e que a cifragem de dados utilizando o algoritmo de El-Gamal leva 24,07 segundos enquanto que a decifragem consome cerca de 17,87 segundos.

Vale ressaltar que estes resultados foram obtidos em simulações sobre a plataforma Mica2. Uma das características desta plataforma é a execução de um ciclo de instrução em um ciclo de relógio, levando apenas dois clocks para multiplicação [Gura et al., 2004]. Esta característica possibilita a projeção do desempenho dos mesmos algoritmos executando sobre outras freqüências. Nas simulações, foram consideradas além de 8 MHz, as freqüências de 16 MHz, 32MHz e 64 MHz. Outra característica das simulações deste trabalho é que tanto para conectar dois quanto cem clusters, o número máximo de saltos assumido foi de cem saltos. Este dado é relevante, pois determina quantas vezes a função hash deve ser aplicada ao campo número de saltos pelo emissor de uma mensagem.

4.4. Os Resultados Obtidos

As simulações comprovam o desempenho esperado do protocolo proposto. Na Figura 1, todas as quatro curvas têm um coeficiente angular pequeno, apesar de parecer que a partir de dez clusters, as curvas têm uma inclinação maior. Nos valores coletados, nota-se um crescimento gradual do tempo em função do número de clusters.

Desde a interconexão de dois até cem clusters, as operações desempenhadas pelos cluster heads fins são sempre as mais relevantes e custosas. O cluster head de origem precisa aplicar Max_Hop_Count vezes a função hash sobre uma semente gerada. Ainda precisa assinar a requisição. Os cluster heads intermediários simplesmente repassam a mensagem antes de verificar a integridade da mesma, computando apenas uma vez a função hash sobre o valor do campo hash contido na

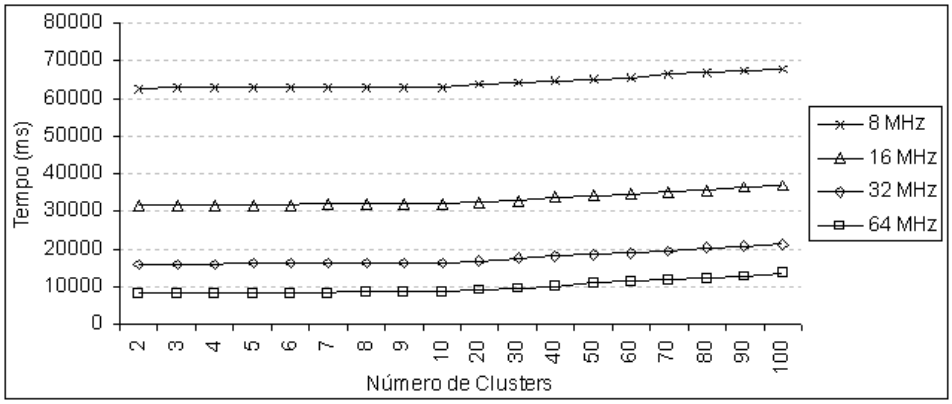

Figura 1. Tempo médio de envio de mensagens CBR (512 bytes) sem rota estabelecida por número de clusters na rede.

mensagem e incrementado o valor do campo Hop_Count. O tempo de verificação de integridade não afeta o desempenho da simulação, somente a aplicação da função hash

retarda a transmissão em 1,473 milissegundos por cluster head intermediário. O cluster head destino, precisa verificar a assinatura da mensagem, gerar uma chave pela matemática de Diffie-Hellman, gerar Max_Hop_Count vezes a função hash sobre uma nova semente gerada e assinar a resposta. Os cluster heads intermediários fazem a mesma coisa na resposta que na requisição. O cluster head destino verifica a assinatura, calcula a chave pela matemática de Diffie-Hellman e decifra a chave de sessão que lhe foi enviada.

O que determina o aumento gradual da interconexão desde dois até cem clusters é o tempo de propagação da mensagem no espaço e o tempo de aplicação da função hash por cada cluster head intermediário. Para cem clusters, o tempo total das funções hash somado chega a 147,3 milissegundos, enquanto que a propagação da mensagem leva 2722 milissegundos. Quanto maior o número de clusters, mais demorado é para uma mensagem atingir o destino, pois a distância cresce na mesma proporção que o número de clusters, e seu valor é mais relevante que o cálculo da função hash.

5. Conclusões

O presente trabalho procurou adaptar o protocolo AODV para o ambiente de redes de sensores organizadas em clusters. Foi apresentado um esquema de distribuição e proteção de chaves simétricas e proposto um mecanismo de interconectar de maneira transparente dois clusters. Verificando o resultado das simulações, notou-se um desempenho uniforme na interconexão de clusters, mesmo quando o número de clusters foi aumentado consideravelmente. Algumas desvantagens desta abordagem devem ser citadas. O modo como as mensagens trafegam, podem sobrecarregar o cluster head, tornando-o um gargalo na comunicação extra cluster. Outro ponto que pode ser encarado como desvantagem é a maneira como são tratadas as autenticações de mensagens entre cluster heads intermediários, podendo haver um retardo na identificação de uma mensagem anômala. Mesmo assim acredita-se que os benefícios apontados neste trabalho sejam maiores que as eventuais desvantagens. O principal atrativo desta abordagem é ter um desempenho constante na troca de mensagens entre dois clusters quaisquer, mesmo com o aumento do número de sensores.

Referências

Arazi, B., Elhanany, I., Arazi, O. e Qi, H. (2005) “Revisiting Public-Key Cryptography for Wireless Sensor Networks”. IEEE Computer, Vol. 38, Num. 11. Novembro.

Basagni, S., Herrín, K., Bruschi, D. e Rosti, E. (2001) “Secure Pebblenets”. In Proc. of the ACM Int. Symp. On Mobile Ad Hoc Networking and computing, ACM Press, New York. pp 156-163.

Bechler, M., Hof, H. J., Kraft, D., Pählke, F. e Wolf, L. (2004) “A cluster-based security architecture for ad hoc networks”. IEEE INFOCOM.

Blaß, E.-O. e Zitterbart, M. (2005) “Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks”. Institute of Telematics, University of Karlsruhe. http://doc.tm.uka.de/tr/TM-2005-1.pdf, Março.

Du, W., Wang, R. e Ning, P. (2005) “An Efficient Scheme for Authenticating Public Keys in Sensor Networks”. MobiHoc’05. Illinois. USA. Maio.

Eschenauer, L. e Gligor, V. D. (2002) “A key-management scheme for distributed sensor networks”. In Proc. of the 9th ACM Conf. On Computer and Communications Security, pp. 41-47. Novembro.

Fokine, K. (2002) “Key management in ad hoc networks”. Master Thesis. http://www.ep.liu.se/exjobb/isy/2002/3322/. Setembro.

Ganesan, P., Venugopalan, R. Peddabachagari, P., Dean, A., Mueller, F. e Sichitiu, M. (2003) “Analyzing and Modeling Encryption Overhead for Sensor Network Nodes”. Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pp. 151-159.

Gura, N., Patel, A., Wander, A., Eberle, H. e Chang-Shantz, S. (2004) “Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs”. CHES’2004. Workshop on Cryptographic Hardware and Embedded Systems. Springer-Verlag. Agosto.

Hu, F. e Sharma, N. K. (2005) “Security considerations in ad hoc sensor networks”. Ad Hoc Networks 3. pp. 69-89.

Hubaux, J., Buttyan, L., e Capkun, S. (2001) “The quest for security in mobile ad hoc networks”. In Proc. ACM Symp. On Mobile Ad Hoc Networking and Computing (MobiHOC).

Kamal, A. B. M. (2004) “Adaptive Secure Routing in Ad Hoc Mobile Network”. Master of Science Thesis. Royal Institute of Technology. Sweden. Novembro.

Lenstra, A. K. e Verheul, E. R. (2001) “Selecting Cryptographic Key Sizes”. Journal of Cryptology: the Journal of the International Association for Cryptologic Research, 14(4): 255-293.

Malan, D. J., Welsh, M. e Smith, M.D. (2004) “A Public-Key Infrastructure for Key Distribuition in TinyOS Based Elliptic Curve Cryptography”. First IEEE International Conference on Sensor and Ad Hoc Communications and Networks.

Okabe, N., Sakane, S., Miyazawa, K., Kamada, K., Inoue, A. e Ishiyama, M. (2005) “Security Architecture for Networks using Ipsec and KINK”. SAINT’2005. International Symposium on Applications and the Internet. IEEE. Computer Society.

Perkins, C. e Belding-Royer, E. (2003) “Ad hoc on-demand distance vector (AODV) routing”. IETF Request for Comments, RFC 3561. Julho.

Perrig, A., Szewczyk, R., Wen, V., Culler, D. E., e Tygar, J. D. (2001) “SPINS: security protocols for sensor networks”. In Proc. of the Seventh Annual Int. Conf. on Mobile Computing and Networking, pp 189-199.

Schneier, B. (1996) “Applied Cryptography: Protocols, Algorithms and Source code in C”. John Wiley & Sons, Inc. New York.

Shoup, V. (2000) "Practical threshold signatures". EUROCRYPT´00 pp. 207–220.

Stallings, W. (1999) “Cryptography and Network Security: Principles and Practice”. Second edition. Prentice Hall, Inc. New Jersey.

Zapata, M. G., e Asokan, N. (2002) “Securing ad hoc routing protocols”. Proc. ACM Workshop on Wireless Security (WiSe), ACM Press, pp. 1-10.

Zhou, L. e Haas, Z. J. (1999) “Securing ad hoc networks”. IEEE Networkd Magazine, 13(6), pp 24-30. Novembro/Dezembro.