Uma arquitetura para prover autenticidade em comunicações *Multicast*

Jean T.Garcia[1], **Lucas Vargas Dias**[1], **Tiago Antonio Rizzetti**[1]

[1]Curso Superior de Tecnologia em Redes de Computadores –
Universidade Federal de Santa Maria (UFSM)
Caixa Postal 5082 – Santa Maria – RS – Brazil

jeangarcia@redes.ufsm.br, lucas_dias@redes.ufsm.br, rizzetti@ctism.ufsm.br

Abstract. *This article describes a architecture from multicast packages authentication in network layer through of hash based message authentication code which key propagation is perform from DHT network. In addition, the architecture addresses authentication of nodes to join the DHT network. This work show fast time for key convergence as well as the speed for mount authentic packages.*

Resumo. *O trabalho traz uma arquitetura para a autenticação de pacotes multicast da camada de rede através de códigos de autenticação de mensagem baseados em hash, onde a disseminação de chaves se da através da rede DHT. Além disso, a arquitetura trata da autenticação dos nós para ingressar na rede DHT. O trabalho apresentou um tempo rápido para a convergência da chave entre os nós, bem como a velocidade para a montagem de pacotes autênticos.*

1. Introdução

Comunicações *multicast* por definição permitem que um único emissor envie uma mensagem a diversos destinatários de maneira a reduzir tráfego de rede, largura de banda e o processamento. Entretanto, qualquer dispositivo que descubra o endereço *Internet Protocol* (IP) do grupo pode participar do mesmo, bem como enviar mensagens.

Essas por sua vez, podem não serem de interesse para o grupo, no caso de aplicações nas quais enviam-se comandos ao grupo, a situação se torna ainda mais crítica. Como no exemplo mostrado em [Wang and Lu 2013] em que um dispositivo eletrônico inteligente (IED) que quando verifica anomalias em uma subestação de energia, emite comandos para disjuntores de disparo para proteger equipamentos de energia. Com isso, há necessidade de garantir que apenas mensagens de interesse para o grupo sejam enviadas pela rede e aceitas pelos dispositivos.

Sendo assim, o respectivo trabalho utilizada de códigos de autenticação de mensagem baseada em *hash* (HMAC) como forma de garantir que os pacotes recebidos são legítimos, ou seja, são provenientes de um dispositivo autenticado que possui conhecimento da chave utilizada pelo grupo. Vale ressaltar que a aplicação do HMAC é em um pacote da camada de rede, portanto, os requisitos de segurança na camada de aplicação não são abordados.

A distribuição da chave utilizada se dá por meio de *Distributed Hash Tables* (DHT). Essa por sua vez, é acoplada a uma infraestrutura de chave pública (ICP) para autenticação dos nós bem como garantir a confidencialidade na propagação das chaves.

Com isso, o restante deste artigo está organizado da seguinte forma: na seção 2 são abordados os trabalhos relacionados, em sequência, na seção 3, o trabalho proposto. Por fim, na seção 4 são trazidos os resultados obtidos junto com as discussões sobre os mesmos, e por fim, as considerações finais.

2. Trabalhos relacionados

Segundo [Baddi and El Kettani 2013], comunicações *multicast* tem como requisito que apenas participantes das mesmas possam enviar e ler mensagens relacionadas ao grupo, dessa forma, é importante garantir que os nós participantes e as mensagens são autênticas. O trabalho ainda trás alguns desafios em relação a segurança em tais comunicações, são eles:

I - Quando um nó sai do grupo, é necessário que as chaves criptográficas sejam alteradas para que o mesmo não possa interagir com os demais dispositivos e II - função de mudança de chave previne que novos membros consigam recuperar mensagens antigas do grupo.

Além disso, [Baddi and El Kettani 2013] divide arquiteturas de gerenciamento de chaves em três classes, sendo a primeira de protocolos de gerenciamento de chaves de grupo centralizado, onde uma única entidade é responsável por fazer o controle do grupo e pela distribuição de chaves. Nessa arquitetura há o problema de que se a entidade falhar, todo o sistema também falhará.

Segundo [Baddi and El Kettani 2013], a segunda classe é de arquiteturas descentralizadas onde o problema de ponto único de falha é resolvido pelo fato que os nós são divididos em subgrupos sendo cada um gerenciado por uma entidade diferente e cada uma dessas faz o compartilhamento de chaves entre si.

Já a terceira classe são de protocolos de gerenciamento de chaves distribuídos onde os mecanismos de segurança são divididos entre entidades previamente autenticadas. Nessa classe não há um centro de distribuição de chaves explícito e os membros cooperam entre si para estabelecer chaves do grupo e para a geração de chaves[Baddi and El Kettani 2013].

O trabalho de [Singh et al. 2012] se encaixa na terceira classe pelo fato de que os nós são divididos em subgrupos, onde cada um tem um líder que comunica-se com os demais. Para entrar em um subgrupo, o nó tem de autenticar-se ao membro da autoridade certificadora (CA) distribuída mais próximo.

Quando um nó desejar fazer uma comunicação *multicast*, o mesmo tem de fazer a requisição ao líder do seu subgrupo no qual irá gerar uma chave de sessão e compartilhar aos líderes dos demais subgrupos [Singh et al. 2012].

Outros desafios em relação a comunicações *multicast* são apontados em [Li and Cao 2011], onde o foco é autenticidade no âmbito de *smart grid*, o trabalho aponta que em aplicações com resposta à demanda, muitas vezes, quando verificado uma sobrecarga de consumo de energia, uma abordagem é desligar os dispositivos menos importantes via comandos de controle. Esses por sua vez, tem de ser autênticos, pois, caso contrário, um agente malicioso poderia facilmente desligar diversos dispositivos.

O trabalho de [Li and Cao 2011] ainda relata que alguns esquemas de autenticação

com chave assimétrica são computacionalmente custosos e os dispositivos envolvidos em *smart grid* geralmente são limitados quanto a isso.

3. Arquitetura proposta

A arquitetura proposta utiliza do escopo da rede DHT por fornecer uma estrutura auto-organizada para aplicações ponto-a-ponto que exigem escalabilidade, resiliência, tolerância a falhas e alta disponibilidade [MacQuire et al. 2006]. Para ingressar na rede, é necessária a autenticação inicial do nó, conforme apresentado na Figura 1.

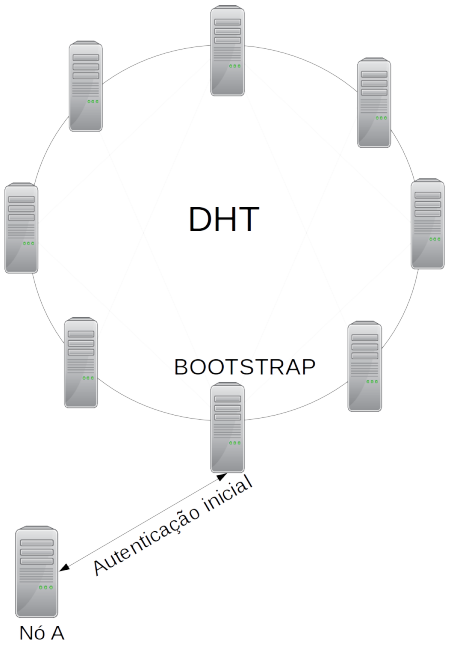

Figura 1. Rede DHT Para controle do *multicast*

Para que o nó A possa realizar tal processo, ele tem conhecimento de uma estrutura que contém todos os nós de *bootstrap* e da lista de certificados revogados (CRL) pela CA. Sendo assim, o nó A contata qualquer dos nós de *bootstrap*, então os dois estabelecem um canal de comunicação *Diffie-Hellmann* como descrito em [William Stallings 2014], em sequência trocam-se os certificados digitais, sendo realizada a verificação da validade dos mesmos.

Caso o certificado do nó A seja válido, o mesmo ingressa na rede DHT e o nó de *bootstrap* armazena, em uma estrutura de dados, o identificador do nó A e o seu certificado. Feito isso, o mesmo gera uma nova chave de sessão, a qual é encriptada com cada uma das chaves públicas dos nós que fazem parte da rede. Cada resultado é concatenado ao identificador do nó na rede, e inserido em um vetor que é publicado na rede DHT. Então, cada nó busca pelo seu identificador no vetor, decifra a chave de sessão e a salva.

O nó de *bootstrap* verifica periodicamente a validade dos certificados armazenados por ele. Quando o certificado de um nó torna-se inválido, portanto, fora do do prazo de validade ou revogado pela CA, uma nova chave de sessão é gerada e publicada na rede DHT como descrito anteriormente. Vale ressaltar que a transmissão dos dados na rede

DHT se dá através de *InfoHash*s que é definido em [Klampanos and Jose 2012] como um *hash* consistente que representa um conjunto de informações determinado por um identificador.

Basicamente, cada nó ingressado na rede DHT fica na escuta da *InfoHash* da chave. Uma vez recuperada a chave de sessão, ao realizar uma comunicação *multicast* para o grupo pré-definido, os passos realizados são apresentados na Figura 2. Vale ressaltar que os possíveis nós de *bootstrap* compartilham os certificados entre si para que caso um falhe, outro possa assumir o seu papel.

No passo 1, o emissor apenas monta o pacote a ser enviado pela rede e repassa ao módulo *multicast*, que é um *software* que faz interação com o kernel, no passo 2. Ao detectar um pacote com endereço de destino, o endereço *multicast* previamente configurado, o mesmo gera um HMAC do pacote IP, anexa o resultado ao mesmo e transmite na rede.

Cada dispositivo que recebe o pacote no passo 3, verifica a integridade do pacote, no caso de sucesso, o mesmo é repassado a aplicação final do destinatário no passo 4.

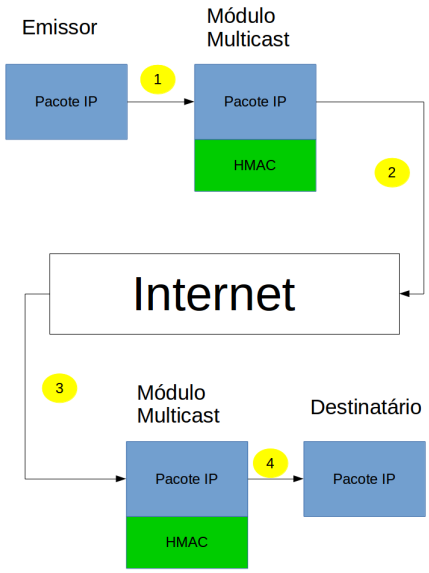

Figura 2. Processo de autenticação de pacotes IP *multicast*

Vale ressaltar que dessa forma é garantida a integridade do pacote, além disso, pressupõem-se que os dispositivos autenticados na rede DHT não irão personificar outros pertencentes a mesma. Sendo assim, podemos considerar os pacotes como autênticos, não há como um dispositivo que não tenha se autenticado na rede descobrir a chave utilizada para o HMAC.

4. Resultados e Discussões

Para colocar a rede DHT em funcionamento foi utilizado a biblioteca *OpenDHT*, as simulações foram realizados no ambiente do utilitário de emulação de redes *core emu-*

lator devido a facilidade proporcionada para os testes[Ahrenholz et al. 2008].

Primeiro, foram realizados testes sobre o tempo de convergência de chave entre diferentes nós, o resultado pode ser visto na Figura 3. É possível verificar que a diferença de tempo no intervalo de 10 a 30 nós é razoavelmente baixa. Optou-se por essa quantidade de dispositivos devido a capacidade de processamento do hospedeiro.

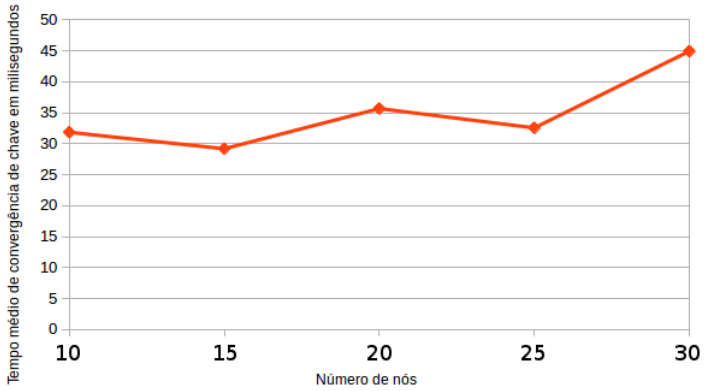

Figura 3. Resultados do tempo de sincronismo e quantidade de tráfego gerado para diferentes quantidades de registros

Com isso, é notável o impacto causado com a simulação de 30 nós. Ainda assim, o tempo de convergência de chave é baixo, em torno de 45ms. O segundo teste realizado foi o tempo que um nó leva para a montagem de pacotes autênticos, foi feita a média do tempo de 10, 15, 20, 25 e 30 pacotes originados pelo mesmo emissor. Também foram gerados a mesma quantidade de pacotes, entretanto, comprometidos, no sentido de que podem sofrer adulterações no meio do caminho ou serem originados por qualquer dispositivo que tenha apenas conhecimento do endereço IP do grupo.

O comparativo pode ser visto na Figura 4. Como era de se esperar, os pacotes autenticados iriam ter um tempo médio maior, ainda assim a diferença é de menos de 0.01ms, portanto, um custo computacional baixo devido as vantagens providas nas comunicações. Vale ressaltar que os testes foram realizados com apenas um nó de *bootstrap*.

5. Considerações Finais

Os resultados mostrados na seção anterior mostram que a arquitetura tem um tempo de convergência de chaves baixo. Isso se dá pelo fato da característica da rede DHT de cada nó que tiver posse de uma informação, mantém a mesma em cache para que nós que buscaram pela informação não precisem contatar o servidor originador.

Além disso, a utilização de certificados digitais para autenticação dos nós ao entrarem na rede garante que a chave de sessão será propagada de maneira confidencial, uma vez que a mesma é encriptada com a chave pública contida em cada certificado. Entretanto, o trabalho proposto não trata de um nó autentico agindo de maneira maliciosa, ficando a abordagem para um trabalho futuro.

Não foram utilizados certificados digitais para integridade e autenticidade dos pacotes devido a baixa escalabilidade. A medida que cresce o número de dispositivos na

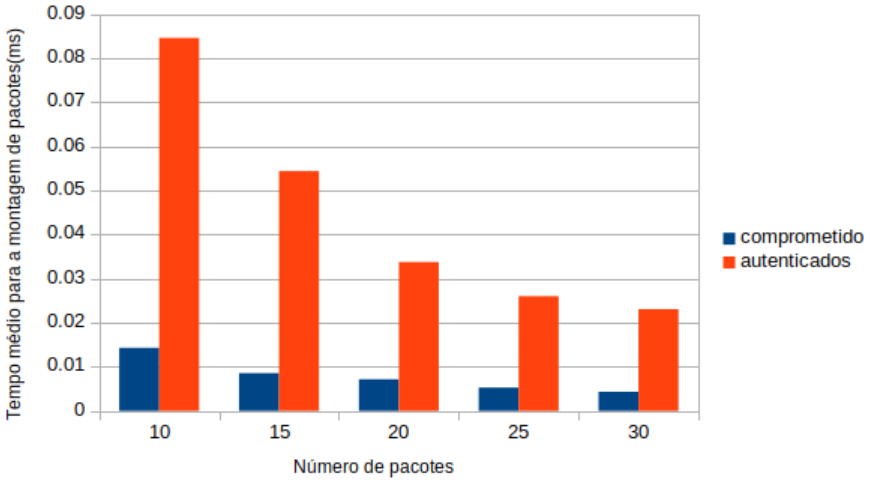

Figura 4. Resultados do tempo de sincronismo e quantidade de tráfego gerado para diferentes quantidades de registros

rede, mais certificados teriam de ser armazenados entre todos os dispositivos, além disso, cada um teria de verificar os mesmos periodicamente para tratar da revogação.

Apesar do escopo do trabalho ter limitado-se a camada de rede, a utilização do mecanismo de propagação de chaves pode ser utilizada, por exemplo, para confidencialidade em dados de aplicação. Devido o baixo custo computacional apresentado, como trabalho futuro, fica a aplicação do mesmo voltado à *smart grid*. Por fim, vale ressaltar que a detecção de falha de um nó de *bootstrap* também não é abordada.

Referências

Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. (2008). Core: A real-time network emulator. In *MILCOM 2008-2008 IEEE Military Communications Conference*, pages 1–7. IEEE.

Baddi, Y. and El Kettani, M. D. E.-C. (2013). Key management for secure multicast communication: A survey. In *2013 National Security Days (JNS3)*, pages 1–6. IEEE.

Klampanos, I. A. and Jose, J. M. (2012). Searching in peer-to-peer networks. *Computer Science Review*, 6(4):161–183.

Li, Q. and Cao, G. (2011). Multicast authentication in the smart grid with one-time signature. *IEEE Transactions on Smart Grid*, 2(4):686–696.

MacQuire, A., Brampton, A., Rai, I. A., Race, N. J. P., and Mathy, L. (2006). Authentication in stealth distributed hash tables. pages 348–355.

Singh, A. P., Potey, S. M., Barbhuiya, F. A., and Nandi, S. (2012). A scalable and secure key distribution mechanism for multicast networks. In *2012 International Conference on Advances in Computing and Communications*, pages 211–214. IEEE.

Wang, W. and Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges. *Computer networks*, 57(5):1344–1371.

William Stallings, L. B. (2014). *Segurança de Computadores: Princípios e Práticas*. Rio de Janeiro.