Inserção de Infraestrutura de Chave Pública no Projeto OpenDHT

Lucas V. Dias[1], **Tiago A. Rizzetti** [1], **Wagner S. Brignol**[2] **e Luciane N. Canha** [1]

[1]Universidade Federal de Santa Maria (UFSM)

[2]Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense (IFSul)

{lucas_dias, rizzetti}@redes.ufsm.br, wagner.brignol@gmail.com, lucianecanha@ufsm.br

Resumo. *As redes de sobreposição peer-to-peer (P2P) baseadas em Distributed Hash Table (DHT) tem ganhado atenção na literatura devido as suas características como alta disponibilidade, escalabilidade, entre outras. Uma implementação popularmente conhecida é o projeto OpenDHT, contudo, não é abordada a autenticidade dos nós, nem das mensagens. Dessa forma, ela é suscetível a diversos ataques, entre eles, ataque Sybil. O objetivo do trabalho é apresentar uma proposta para preencher essa lacuna através de uma Infraestrutura de Chave Pública (ICP). Os resultados obtidos através da ferramenta Scyther demonstram a segurança do protocolo de autenticação contra ataques bem conhecidos e os testes práticos demonstram êxito da implementação.*

1. Introdução

As redes de comunicação P2P tem sido cada vez mais atrativas e geram uma grande parte do tráfego de rede global. Muitas redes P2P são baseadas em DHT. Uma rede DHT pode ser vista como um banco de dados distribuído onde cada nó, armazena um conjunto das informações. Essas são compostas por um *hash* criptográfico e valor. Redes DHT são caracterizadas por sua tolerância a falhas, alta disponibilidade e escalabilidade [Srinivasan and Aldharrab 2019]. Dessa forma, ela é atrativa para uma diversidade de aplicações como em [Rodrigues et al. 2016], em que a rede DHT é utilizada para propagação de regras de *firewall*. Por outro lado, em [Hassanzadeh-Nazarabadi et al. 2018], a tecnologia é usada para redundância de dados. [Eisele et al. 2017] usa a rede DHT para prover um serviço de descoberta de aplicações e [Hassanzadeh-Nazarabadi et al. 2019] utiliza para implementação de uma *Blockchain* em ambientes com recursos computacionais limitados.

Ainda assim, existem diferentes tipos de rede DHT. Um protocolo para rede DHT que ganhou destaque foi o Kademlia pois o número de vizinhos de cada nó é $log\ n$ [Maymounkov and Mazieres 2002, Chowdhury and Ferdous 2017]. Porém, não há mecanismos de autenticação nativos no protocolo. Com isso, a rede é suscetível a ataques *Sybil*, *Eclipse*, entre outros. Um tipo de ataque *Sybil* que ganhou destaque é o ataque *Spartacus*, em que nós maliciosos personificam os nós legítimos da rede [Li et al. 2018, Panos et al. 2017].

Visando contribuir com ferramentas para provimento de redes DHT seguras, este trabalho propõe a utilização de uma ICP para autenticação dos nós na inserção de uma rede DHT. Adicionalmente, é fornecida uma implementação através da modificação do código fonte da biblioteca OpenDHT[1] para validação da proposta. Os testes práticos

[1]https://github.com/savoirfairelinux/opendht

demonstram êxito da solução proposta. A segurança da troca de mensagens na versão original e modificada da OpenDHT são verificadas com a ferramenta Scyther. O restante deste artigo está estruturado da seguinte forma: Seção 2 apresenta os trabalhos relacionados. Em sequência, a solução proposta é apresentada na Seção 3. Os testes e resultados são descritos na Seção 4. Por fim, na Seção 5, são dadas as considerações finais.

2. Trabalhos Relacionados

Essa Seção aborda os trabalhos relacionados que apresentam mecanismos de defesa contra ataques em redes P2P. O trabalho de [Lim et al. 2017] tem por objetivo detectar e remover nós *Sybil* em ambientes *Cloud*. Para isso, é utilizada assinatura *fail-stop*. Para detecção de nós maliciosos, cada nó seleciona algum vizinho e envia suas informações locais. O nó que recebe a mensagem, verifica se o emissor foi detectado previamente como nó *Sybil*, se as informações enviadas não são conflitantes com a identidade do nó e por fim, a assinatura da mensagem. Em sequência, o nó vizinho envia suas informações ao requisitante que as verifica da mesma forma. Adicionalmente, uma terceira entidade confiável é usada para fornecer os parâmetros públicos das chaves usadas pelos nós [Lim et al. 2017]. Contudo, não há autenticidade dos nós, apenas das mensagens.

Ainda, o trabalho de [Pecori and Veltri 2018] tem por objetivo prevenir ataques *Spartacus* e *Sybil* em redes DHT. Para isso, os nós usam métricas de confiança. Ela é baseada nas funções de busca de nós, alocação e recuperação de recursos. Cada nó verifica e calcula as métricas localmente, dessa forma, um nó considerado malicioso por um, pode não ser considerado malicioso para outros. Adicionalmente, autenticidade dos nós e das mensagens não são tratadas e o protocolo DHT levado em consideração é o Chord. Por outro lado, a proposta apresentada na Seção 3 leva em consideração uma rede DHT baseada no Kademlia. Os mecanismos empregados realizam autenticidade dos dispositivos e das mensagens. Para isso, uma ICP é usada e é pressuposto que ela é segura.

3. Solução Proposta

O objetivo principal da proposta é permitir a entrada de apenas nós autênticos na rede DHT, bem como remover os nós com certificado revogado em tempo de execução. Para isso, a arquitetura utiliza de uma ICP. A rede DHT funciona de uma maneira convencional, contudo, ela é gerenciada pela ICP como visto na Figura 1. Em sequência, o funcionamento da OpenDHT é detalhado, assim como a solução proposta.

Na maneira convencional, a OpenDHT permite a inserção de qualquer nó na rede. O novo dispositivo deve contatar o nó de *bootstrap* para ingressar na rede DHT. Para isso, ele gera um pacote com um *nonce*, o tipo de operação, que nesse momento é identificado como *PING*, o identificador do nó, além de uma *string* pré-definida. O nó de *bootstrap* recebe a mensagem, verifica as informações, adiciona o novo nó na sua tabela de roteamento e envia um pacote com seu identificador, o *nonce* recebido, a confirmação do identificador do novo nó e a *string* pré-definida. Após isso, o nó solicitante adiciona o *bootstrap* na sua tabela de roteamento e passa a trabalhar na rede DHT. Contudo, nenhum mecanismo de autenticidade da mensagem e do dispositivo é aplicado. Com isso, ela é suscetível ataques *Sybil*, ataques *Spartacus*, entre outros. Em contrapartida, a versão modificada adiciona o certificado digital do dispositivo junto as mensagens, o endereço *Internet Protocol* (IP) do destinatário bem como assinatura digital dela.

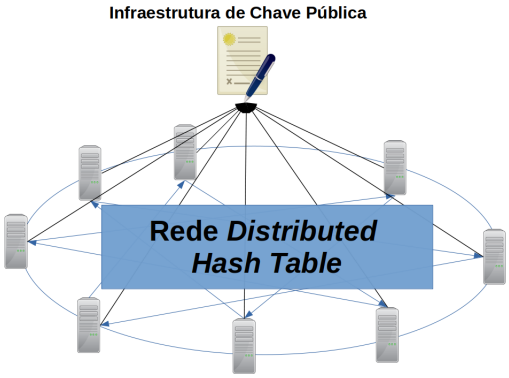


Figura 1. Arquitetura Proposta.

Ao receber os pacotes, os dispositivos verificam a validade do certificado. Se for inválido, o nó é adicionado a lista de nós revogados. Caso a assinatura seja inválida, o nó é adicionado a uma *blacklist* de nós que enviaram pacotes ou mensagens com erros. Em ambos os casos, o nó não pode alocar, nem buscar recursos na rede. Adicionalmente, o nó é adicionado a lista de revogados, quando seu certificado digital for revogado pela autoridade certificadora (CA) ou estiver fora do prazo de validade. A seguir, uma análise formal, utilizando a ferramenta Scyther e testes práticos são apresentados na Seção 4.

4. Testes e Resultados

Essa Seção apresenta os testes e resultados. Inicialmente, para validação do algoritmo proposto, foi usada a ferramenta de análise automatizada de segurança de protocolos denominada Scyther. Para verificação do protocolo são necessários definir `Roles`, que são as partes comunicantes e os eventos de troca de mensagens. Adicionalmente, para verificação de segurança, as *claim*s tem de ser definidas. Nos testes apresentados na Figura 2(a) e Figura 2(b), foram utilizadas a *claim*s `Alive`, `Weakagree`, `Niagree` e `Nisynch`, que servem para autenticidade da mensagem, dispositivos e resistência a ataques de repetição [Cremers 2008]. A versão original possui vulnerabilidades, enquanto que a solução proposta passou nos testes.

Scyther results : verify

Claim				Status		Comments	Pattern
Opendht	newnode	Opendht,newnode1	Nisynch	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,newnode2	Alive	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,newnode3	Weakagree	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,newnode4	Niagree	Fail	Falsified	Exactly 1 attack.	1 attack
	bootstrap	Opendht,bootstrap1	Nisynch	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,bootstrap2	Alive	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,bootstrap3	Weakagree	Fail	Falsified	Exactly 1 attack.	1 attack
		Opendht,bootstrap4	Niagree	Fail	Falsified	Exactly 1 attack.	1 attack

Done.

(a) Resultado do Scyther da OpenDHT original.

Scyther results : verify

Claim				Status		Commer
OpendhtModified	newnode	OpendhtModified,newnode1	Nisynch	Ok	Verified	No attacks.
		OpendhtModified,newnode2	Alive	Ok	Verified	No attacks.
		OpendhtModified,newnode3	Weakagree	Ok	Verified	No attacks.
		OpendhtModified,newnode4	Niagree	Ok	Verified	No attacks.
	bootstrap	OpendhtModified,bootstrap1	Nisynch	Ok	Verified	No attacks.
		OpendhtModified,bootstrap2	Alive	Ok	Verified	No attacks.
		OpendhtModified,bootstrap3	Weakagree	Ok	Verified	No attacks.
		OpendhtModified,bootstrap4	Niagree	Ok	Verified	No attacks.

Done.

(b) Resultado do Scyther da OpenDHT proposta.

Figura 2. Testes com a ferramenta Scyther.

O primeiro teste prático foi um ataque de repetição, em que o nó legítimo com endereço `10.0.0.39`, envia uma mensagem de inserção na rede para o nó `10.0.0.36`.

O nó malicioso (`10.0.0.38`) recupera o pacote, e reenvia ao nó `10.0.0.36`. A Figura 3 mostra o fluxo de ataque.

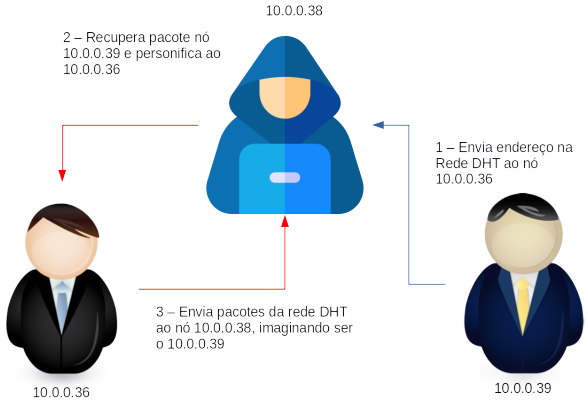


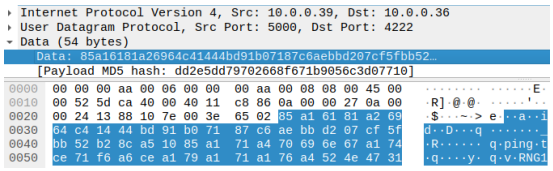
Figura 3. Fluxo de ataque.

Por outro lado, a Figura 4 mostra êxito no ataque. O nó `10.0.0.36` passa a enviar os pacotes de manutenção para o endereço do nó malicioso que passa a assumir o endereço `44bd91b07187c6aebbd207cf5fbb52b28ca51085` do nó `10.0.0.39` na rede DHT.

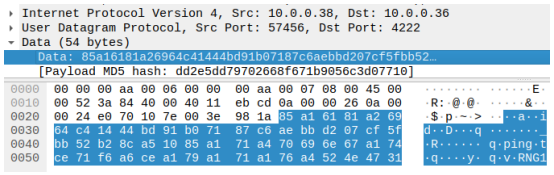
```
[003146.948053] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending find 8fdc0790116ecc83d0831ba1a901e59643c8f9fa for bucke
t maintenance
[003158.982839] [confirm nodes] initial IPv4 'get' for my id (786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636)
[003158.982876] [search 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636 IPv4] new search
[003158.982888] [search 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636 IPv4] refilled search with 1 nodes from node cache
[003158.982911] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66d9] for nei
ghborhood maintenance
[003181.936800] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66d3] for nei
ghborhood maintenance
[003195.554319] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66ec] for nei
ghborhood maintenance
[003203.284369] [search 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636] removing search
[003207.093089] [confirm nodes] initial IPv4 'get' for my id (786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636)
[003207.093119] [search 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636 IPv4] new search
[003207.093130] [search 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6636 IPv4] refilled search with 1 nodes from node cache
[003207.093163] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66ee] for nei
ghborhood maintenance
[003231.175065] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.38:55057] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6625] for ne
ighborhood maintenance
[003239.618878] Bootstraping
[003244.915159] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.39:5000] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b6620] for nei
ghborhood maintenance
[003257.723121] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.38:45934] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66ba] for ne
ighborhood maintenance
[003266.123258] Bootstraping
[003282.782156] [node 44bd91b07187c6aebbd207cf5fbb52b28ca51085 10.0.0.38:57656] sending [find 786841c91ffb0b4fd5e3f4dc3b3f4791b42b66ee] for ne
ighborhood maintenance
[003291.132702] Bootstraping
```

Figura 4. Ataque de repetição e personificação realizado.

A Figura 5(a) e Figura 5(b) mostram a solicitação de inserção do nó legítimo e malicioso, respectivamente. Os pacotes na camada de aplicação são iguais. O ataque foi executado com êxito, em acordo com o resultado obtido no Scyther.



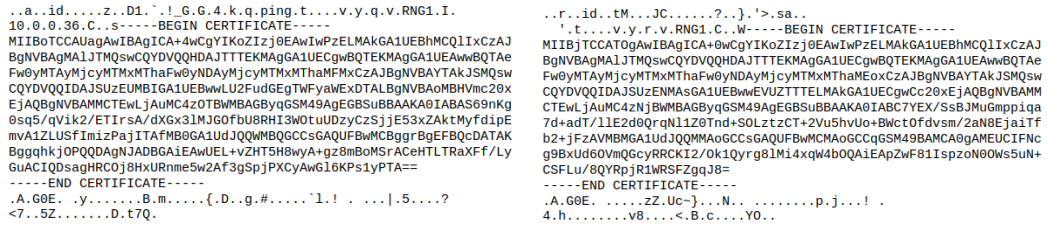
(a) Requisição do nó legítimo.



(b) Requisição do nó malicioso.

Figura 5. Fluxo de comunicação de inserção na rede DHT.

Para prevenção desse tipo de ataque, a proposta emprega assinatura digital da mensagem e certificados digitais para autenticação dos dispositivos. A implementação da proposta foi desenvolvida em C++ e utiliza a biblioteca Openssl[2] para operações criptográficas. A requisição na versão modifica da OpenDHT é mostrada na Figura 6(a). Por outro lado, a Figura 6(b) mostra a resposta recebida.

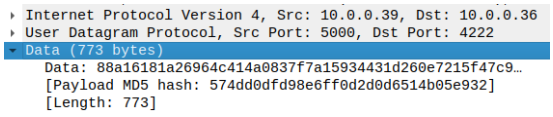


(a) Requisição com assinatura digital e certificado digital.

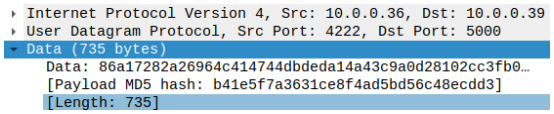
(b) Resposta da requisição com assinatura digital e certificado digital.

Figura 6. Requisição e resposta de versão modificada da OpenDHT.

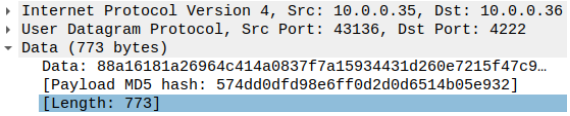
Adicionalmente, a Figura 7(a) e Figura 7(b) demonstram a mensagem enviada pelo nó que deseja ingressar na rede e o que já faz parte, respectivamente. O certificado e assinatura digital garantem autenticidade do dispositivo e da mensagem. Dessa forma, com ambas legítimas, há a inserção do novo nó na rede. Por outro lado, a Figura 7(c) mostra a tentativa de um ataque de repetição. O pacote é igual ao enviado pelo nó legítimo, contudo, o malicioso não obteve resposta devido a autenticidade do dispositivo.



(a) Requisição da solução proposta.



(b) Resposta da requisição da solução proposta.



(c) Tentativa de ataque

Figura 7. Fluxo de comunicação da solução proposta.

5. Considerações Finais

Por volta de 2021, redes P2P baseadas em DHT tem se tornado mais populares devido a alta disponibilidade, resiliência, escalabilidade, entre outras coisas. A biblioteca OpenDHT, utilizada no projeto JAMI[3], que permite comunicações de voz e vídeo em tempo-real, apresenta falha na autenticação dos nós. Isso foi demonstrado de maneira prática em conformidade com a análise automatizada com a ferramenta Scyther. Para preencher essa lacuna, foi inserido autenticação das mensagens e dos dispositivos na biblioteca. Os testes automatizados apontam segurança na definição do protocolo e os testes práticos corroboram com isso.

[2]https://www.openssl.org/

[3]https://jami.net/

Contudo, a solução apresenta ponto único de falha com uso do ICP. Portanto, a migração de ICP para uma arquitetura descentralizada, além da utilização de assinatura digital nas demais operações na rede DHT e testes de escalabilidade ficam como trabalhos futuros. Por fim, fica o agradecimento ao INCTGD, órgãos financiadores (CNPq processo n° 465640/2014-1, CAPES processo n° 23038.000776/2017-54 e FAPERGS n° 17/2551-0000517-1) e a CAPES – Brasil (CAPES/PROEX) – Código de Financiamento 001.

Referências

Chowdhury, F. and Ferdous, M. S. (2017). Performance analysis of r/kademlia, pastry and bamboo using recursive routing in mobile networks. *International Journal of Computer Networks & Communications (IJCNC)*, 9(5).

Cremers, C. J. (2008). The scyther tool: Verification, falsification, and analysis of security protocols. In *International conference on computer aided verification*, pages 414–418. Springer.

Eisele, S., Mardari, I., Dubey, A., and Karsai, G. (2017). Riaps: Resilient information architecture platform for decentralized smart systems. In *2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 125–132. IEEE.

Hassanzadeh-Nazarabadi, Y., Küpçü, A., and Özkasap, Ö. (2018). Decentralized and locality aware replication method for dht-based p2p storage systems. *Future Generation Computer Systems*, 84:32–46.

Hassanzadeh-Nazarabadi, Y., Küpçü, A., and Özkasap, Ö. (2019). Lightchain: A dht-based blockchain for resource constrained environments. *arXiv preprint arXiv:1904.00375*.

Li, J., Li, T., Ren, J., and Chao, H.-C. (2018). Enjoy the benefit of network coding: Combat pollution attacks in 5g multihop networks. *Wireless Communications and Mobile Computing*, 2018.

Lim, J. B., Yu, H. C., and Gil, J. M. (2017). Detecting sybil attacks in cloud computing environments based on fail-stop signature. *Symmetry*, 9.

Maymounkov, P. and Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer.

Panos, C., Ntantogian, C., Malliaros, S., and Xenakis, C. (2017). Analyzing, quantifying, and detecting the blackhole attack in infrastructure-less networks. *Computer Networks*, 113:94–110.

Pecori, R. and Veltri, L. (2018). A balanced trust-based method to counter sybil and spartacus attacks in chord. *Security and Communication Networks*, 2018.

Rodrigues, A. S., Rizzetti, T. A., Canha, L. N., Milbradt, R. G., Appel, S. F., and Duarte, Y. S. (2016). Implementing a distributed firewall using a dht network applied to smart grids. In *2016 51st International Universities Power Engineering Conference (UPEC)*, pages 1–5. IEEE.

Srinivasan, A. and Aldharrab, H. (2019). Xtra—extended bit-torrent protocol for authenticated covert peer communication. *Peer-to-Peer Networking and Applications*, 12(1):143–157.