DLCP: Um Protocolo para a Operação Segura de Clientes Leves em *Blockchains*

Leonardo da Costa[1], André Neto[1], Billy Pinheiro[1], Roberto Araújo[1], Antônio Abelém[1]

[1]Faculdade de Computação – Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

{lbc,billy,rsa,abelem}@ufpa.br, andre.neto@icen.ufpa.br

Abstract. *In blockchain, full nodes (FNs) are peers that store and verify the entire chain of data, while light clients (LCs) are those that request only block headers to an FN, performing simpler verifications. To cope with malicious behavior, a conventional approach to ensure the receipt of correct headers is requesting them to multiple FNs and comparing received responses. This approach, however, requires LCs to establish secure connections to each FN, which leads to greater overhead and response time. In this context, this paper proposes Distributed Lightweight Client Protocol (DLCP), that requires an LC to encrypt a headers request once to a set of FNs, which replies back to the LC with a single response. Preliminary evaluations have shown that DLCP provides lower latency than the conventional approach, while reducing overhead in LCs.*

Resumo. *Em blockchains, nodos completos (NCs) são pares que armazenam e verificam todas as transações contidas nos blocos, enquanto clientes leves (CLs) são aqueles que solicitam apenas os cabeçalhos dos blocos à um NC, realizando verificações mais simples. Para lidar com comportamentos maliciosos, a abordagem convencional para garantir o recebimento dos cabeçalhos originais é solicitá-los à múltiplos NCs e comparar as respostas recebidas. Essa abordagem, contudo, requer que um CL estabeleça conexões seguras com cada NC, o que resulta em maior sobrecarga e tempo de resposta. Nesse contexto, esse trabalho propõe o Distributed Lightweight Client Protocol (DLCP), que demanda criptografar uma requisição de cabeçalhos apenas uma vez para um conjunto de NCs, que, por sua vez, retornam uma única resposta para o CL. Avaliações preliminares mostraram que o DLCP provê menor latência que a abordagem convencional e reduz a sobrecarga nos CLs.*

1. Introdução

Blockchain é uma tecnologia promissora, vista por muitos como base para a infraestrutura da Internet em uma gama de cenários no futuro. As redes de *blockchain*, assim como as experimentais, são redes de sobreposição que rodam sobre redes IP [Croman et al. 2016]. Contudo, diferentemente de soluções convencionais de rede, as *blockchains* já foram pensadas para permitir a criação de ambientes experimentais de escala global, executando-se aplicações distribuídas em ambiente similar ao de produção, porém sem os custos deste.

Uma *blockchain* é mantida por uma rede P2P (*peer-to-peer*) com vários nodos completos (NCs), que armazenam todos os registros de transações em uma cadeia de blocos. Cada novo bloco é encaminhado à toda rede para que todos os NCs possam

validar as transações. Contudo, algumas *blockchains*, como o Bitcoin, apresentam um largo tamanho em *gigabytes*, sendo um problema para usuários que possuem dispositivos com restrições de recursos (e.g. alguns *notebooks*, *smartphones*) [Zheng et al. 2017].

Para contornar a necessidade de rodar um NC, clientes leves (CLs) foram propostos [Nakamoto 2008]. Os CLs empregam verificações de transações mais simples, como o *Simple Payment Verification* (SPV). Para isso, eles requisitam os cabeçalhos dos blocos para um NC e usam-os no SPV para verificar a existência de uma transação em um bloco. Se um NC é malicioso, este pode repassar cabeçalhos falsos à um CL. Para mitigar isso, a abordagem convencional usada por plataformas de clientes leves populares é solicitar a cadeia de cabeçalhos à vários NCs e compará-las [Bitcoinj 2018, Jaxx 2018, Infura 2018]. Requisitar os cabeçalhos de maneira segura, entretanto, requer que um CL abra um canal seguro de comunicação com cada NC, sendo necessário criptografar a mesma requisição para cada um, resultando para o CL em aumento de processamento e tempo de resposta.

Nesse contexto, esse trabalho introduz o *Distributed Lightweight Client Protocol* (DLCP), que permite à um CL criptografar a requisição dos cabeçalhos apenas uma vez e envia-la para um conjunto de NCs. Apenas uma maioria de NCs precisa tratar a requisição para responder ao CL com os cabeçalhos. O CL recebe uma resposta correta se os NCs são honestos e concordam com a mesma. Experimentos preliminares mostraram que o DLCP diminui a sobrecarga no CL, sendo mais eficiente que a abordagem convencional. O restante desse trabalho está estruturado da seguinte forma. A Seção 2 introduz as tecnologias empregadas no DLCP. A Seção 3 descreve o DLCP em detalhes. A Seção 4 avalia o DLCP. Por fim, a Seção 5 conclui esse trabalho e aponta trabalhos futuros.

2. Criptossistemas de Chave Pública e Esquemas de Assinatura Limiares

Um criptossistema de chave pública limiar [Gennaro et al. 2016] é composto por um par de chaves e um conjunto de n participantes. Enquanto a chave pública é única, a chave privada correspondente é compartilhada entre os participantes. Isto é, cada participante possui um segredo relacionado à chave privada conhecido apenas por ele. Para criptografar uma mensagem, usa-se a chave pública. Para descriptografar, um subconjunto com $t \leq n$ participantes deve empregar seus segredos no protocolo de descriptografia. Similarmente, em um esquema de assinatura limiar [Gennaro et al. 2016] é gerado um par de chaves de assinatura, onde uma chave pública única é associada à uma chave privada compartilhada entre n participantes. Para assinar uma mensagem usando a chave privada, os segredos de um subconjunto com $t \leq n$ participantes são empregados. Qualquer entidade é capaz de verificar a assinatura através da aplicação da chave pública correspondente.

3. *Distributed Lightweight Client Protocol* (DLCP)

O DLCP é baseado em criptografia e assinatura limiar e consiste em duas fases: configuração e operação. A Figura 1 ilustra uma visão geral do protocolo com os seus respectivos participantes. A seguir, as premissas e o modelo de adversário relacionados ao DLCP são apresentados e, posteriormente, as fases do protocolo são detalhadas.

Premissas. Assume-se que as autoridades certificadoras online (ACOs) são confiáveis e que os CLs contém os módulos necessários para realizar a descoberta de NCs.

Modelo de adversário. O objetivo do DLCP é impedir ataques de um NC malicioso. Por não armazenar toda a cadeia de blocos, um CL requisita apenas os cabeçalhos dos blocos

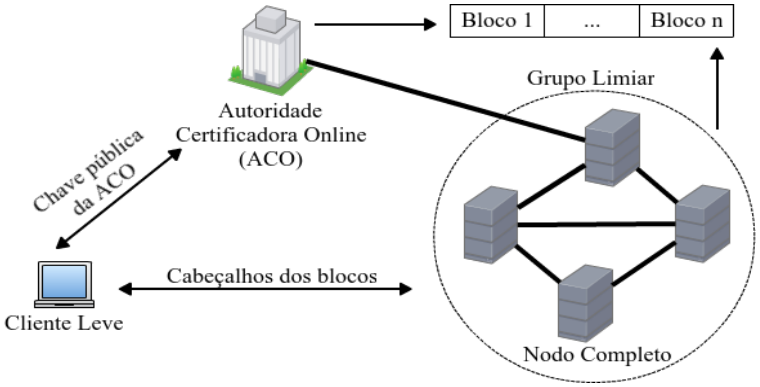


Figura 1. Visão geral do DLCP.

à um NC. Assim, se um CL confiar em um único NC, este é capaz de enganar o CL ao fornecer uma cadeia de cabeçalhos falsa.

3.1. Fase de Configuração

Essa fase tem como propósito inicializar NCs e CLs. Nela, NCs cooperam para definir vários grupos limiares que servirão CLs. Um grupo limiar consiste em NCs que compartilham pares de chaves de um criptossistema de chave pública limiar e de um esquema de assinatura limiar. Por sua vez, CLs requisitam participação na *blockchain*, recebendo as chaves públicas de um grupo limiar, que, mais adiante, é usado para receber os cabeçalhos.

Os NCs usam a *blockchain* para formar grupos limiares. Um NC interessado em participar de um grupo deve registrar o seu interesse na *blockchain* (juntamente com um número estimado n de membros e o valor limiar t) ou buscar por NCs que já tenham registrado tal interesse. Denomina-se o primeiro de NC registrado e o segundo de NC interessado. NCs interessados realizam um protocolo de *handshake* com NCs registrados em um grupo. Cada NC então registra na *blockchain* um comprometimento indicando a criação do grupo e especificando seus membros. O registro de n comprometimentos para o mesmo grupo viabiliza a sua criação.

Com a criação de um grupo, um de seus NCs gera os parâmetros de criptografia e de assinatura limiar e os envia para os outros membros. Os parâmetros são então usados para a geração cooperativa de dois pares de chaves pelo grupo, um par $(\mathrm{PK}^E, \mathrm{SK}^E)$ de um criptossistema de chave pública limiar e outro par $(\mathrm{PK}^S, \mathrm{SK}^S)$ de um esquema de assinatura limiar, onde PK^E e PK^S são chaves públicas e SK^E e SK^S são chaves privadas. As chaves privadas são compartilhadas entre os NCs do grupo, onde cada NC com identificador i possui uma parte de chave SK^E_i e uma parte de chave SK^S_i. As chaves PK^E e PK^S são postadas na *blockchain* pelos NCs. Considera-se uma infraestrutura de chaves públicas com um conjunto de ACOs que também são NCs. As ACOs monitoram a *blockchain* a fim de detectar a postagem de novas chaves. Assim que detectado, uma ACO recupera-as, verifica se os comprometimentos gerados pelo grupo são de NCs legítimos e assina cada chave, postando as chaves assinadas na *blockchain*.

Assim que vários grupos limiares são estabelecidos, segue-se a inicialização de CLs. Um CL inicializa ao requisitar as chaves públicas de um grupo limiar e da ACO. As chaves e informações sobre seu grupo correspondente podem estar listados em *websites* por um *gateway* ou pelos NCs. Dessa maneira, um usuário de um CL pode selecionar um

grupo de acordo com seus interesses. Ao receber as chaves públicas, o CL deve checar se elas foram corretamente geradas, verificando a assinatura da ACO contida nas chaves.

3.2. Fase de Operação

Após a configuração, segue-se a fase de operação. Nela, um CL requisita os cabeçalhos dos blocos para que, posteriormente, ele possa validar transações. O CL confia em um grupo limiar para obter os cabeçalhos. Ele é capaz de validá-los através da assinatura gerada por uma maioria de NCs do grupo, onde sua validade indica que os NCs concordaram sobre os cabeçalhos originais da *blockchain*. Se essa maioria de NCs é honesta, o CL recebe os cabeçalhos originais. A fase de operação é executada conforme a seguir:

1. O CL cria a requisição D dos cabeçalhos e a criptografa com a chave K usando um criptossistema simétrico, o que resulta em um texto criptografado CT_D.
2. O CL criptografa K usando a chave pública PK^E do grupo limiar, resultando em um texto criptografado CT_K. Ele então envia CT_D e CT_K para o grupo limiar. As criptografias iniciais impedem ataques do homem do meio.
3. Ao receber CT_D e CT_K, cada NC com identificador i usa o mesmo criptossistema de chave pública limiar para gerar uma parte de descriptografia de CT_K usando sua parte de chave privada SK_i^E. Ele então a envia para os outros membros do grupo. Ao receber um número limiar t de partes de descriptografia, qualquer NC do grupo é capaz de realizar um cálculo final para obter acesso à K.
4. Cada NC usa a chave simétrica K para descriptografar CT_D, obtendo a requisição D e executando-a ao construir a resposta H com os cabeçalhos.
5. Cada NC com identificador i aplica sua parte de chave privada SK_i^S no mesmo esquema de assinatura limiar para gerar uma parte de assinatura de H, que é enviada ao grupo. Ao receber um número limiar t de partes de assinatura, qualquer NC do grupo é capaz de obter acesso à assinatura completa σ de H.
6. Um NC é encarregado de criptografar H e σ usando K, resultando em um texto criptografado CT_R. Ele envia CT_R para o CL. O CL pode, na requisição, indicar o NC responsável por essa etapa baseado na reputação dos NCs na rede (e.g. NCs honestos poderiam informar ao CL sobre ações maliciosas de um dado NC).
7. Ao receber CT_R, o CL descriptografa-o usando K, obtendo acesso à H e σ.
8. O CL então verifica a assinatura σ usando a chave pública PK^S do grupo. O resultado da verificação indica se o CL deve ou não confiar na resposta recebida.

4. Avaliação

O Ethereum [Ethereum 2018] foi utilizado para criar uma *blockchain* na qual a abordagem convencional (usando TLS entre CL e NCs) e o DLCP foram testados. Como o Ethereum contém uma vasta API em Javascript, essa foi a linguagem usada para implementar o DLCP. Utilizou-se o criptossistema simétrico AES, o criptossistema de chave pública El Gamal limiar (chaves de 1024 bits) [Pedersen 1991] e um esquema de assinatura limiar baseado em criptografia de curvas elípticas (chaves de 128 bits) [Pei and Ma 2008].

Executou-se os experimentos em 4 máquinas virtuais (MVs) com processador Intel Core i5 e 2GB de RAM, alocando NCs, e 1 MV com processador Inter Core i5 e 8GB de RAM (para fins de testes de memória), alocando um CL. Como no Bitcoin o número padrão de NCs ao qual um CL conecta-se é 4 [Gervais et al. 2014], os experimentos foram

inicialmente executados com grupos contendo 4 NCs. Depois, o número de NCs foi sequencialmente aumentado em 4 até totalizar 16. O tamanho total dos cabeçalhos também foi variado entre 100KB e 100MB para abranger diferentes tamanhos de *blockchain*.

As métricas avaliadas foram: latência (tempo entre a requisição e o recebimento dos cabeçalhos) e uso de memória pelo CL. Cada experimento foi executado 50 vezes. A Figura 2 ilustra os resultados referentes à latência. A abordagem convencional apresentou latências médias que variam entre 770 e 2493 milissegundos, enquanto que o DLCP apresentou latências médias entre 533 e 2377 milissegundos. Em todos os cenários, o DLCP foi mais eficiente que a abordagem convencional. O DLCP pode reduzir a latência do processo para obtenção dos cabeçalhos da *blockchain* em até 30,7%, com 4 NCs e cabeçalhos com 100KB. O percentual de redução, contudo, diminui significativamente com 12 e 16 NCs, diminuindo a latência, nesses casos, em menos que 13%. Note-se que de acordo com o aumento do número de NCs em um grupo limiar, a latência do DLCP vai se aproximando da latência da abordagem convencional. Isso acontece devido à complexidade comunicacional dos sistemas limiares. Um alto número de NCs demanda várias comunicações P2P simultâneas, diminuindo, assim, o grau de eficiência do DLCP.

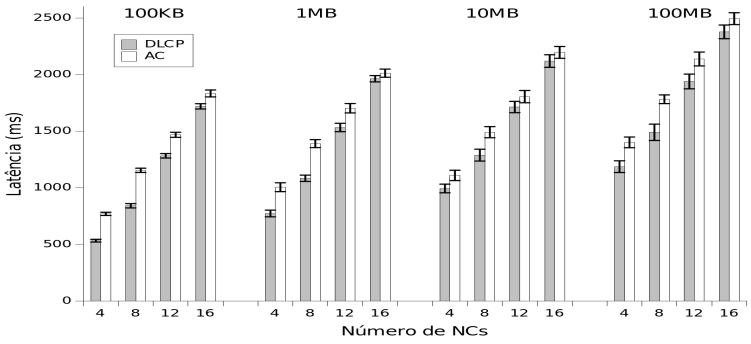
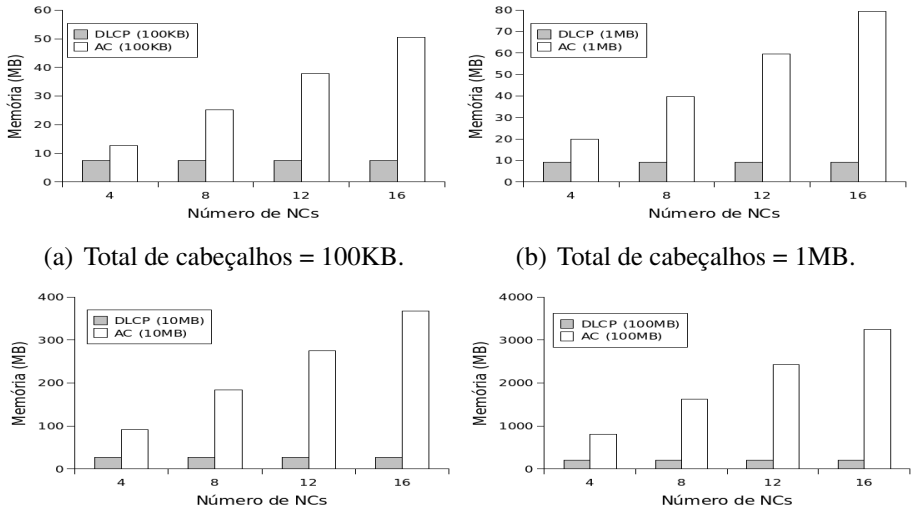


Figura 2. DLCP vs. abordagem convencional (AC) - Latência

A Figura 3 ilustra o uso de memória pelo CL. Tanto no DLCP quanto na abordagem convencional, o uso de memória é diretamente proporcional à aproximadamente o dobro do tamanho total dos cabeçalhos. Contudo, na abordagem convencional, o CL precisa tratar o recebimento dos cabeçalhos de acordo com o número de NCs. Por exemplo, quando os cabeçalhos apresentam 100MB e o CL conecta-se à 8 NCs, o uso de memória é de 1623MB ($\approx$ 8 x 200MB). Já no DLCP, o CL apenas precisa tratar o recebimento dos cabeçalhos uma vez, independentemente do número de NCs empregados. Para 8 NCs e cabeçalhos com 100MB, usa-se somente 207MB. Os resultados mostram que o DLCP diminui significativamente a sobrecarga no CL comparado à abordagem convencional.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou o DLCP, o qual assegura a operação de CLs em *blockchains*. Diferentemente da abordagem convencional, no DLCP, um CL criptografa uma requisição de cabeçalhos uma vez para um grupo de NCs de qualquer tamanho. Concluiu-se que o DLCP reduz significativamente o uso de memória nos CLs. Além disso, ele pode ser até 30.7% mais eficiente em latência que a abordagem convencional. Como trabalhos futuros, pretende-se executar testes de escalabilidade, comparar o DLCP ao estado da arte usando outras *blockchains* e definir um mecanismo eficiente para regeração de chaves de grupos.



(a) Total de cabeçalhos = 100KB.

(b) Total de cabeçalhos = 1MB.

(c) Total de cabeçalhos = 10MB.

(d) Total de cabeçalhos = 100MB.

Figura 3. DLCP vs. abordagem convencional (AC) - Uso de memória pelo CL

Agradecimentos

Essa pesquisa teve suporte parcial da chamada conjunta RNP-NSF para pesquisa e desenvolvimento em cibersegurança (INSaNE - *Improving Network Security at the Network Edge*), financiada pela *National Science Foundation* e pelo Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTIC) através da RNP e do CTIC.

Referências

Bitcoinj (2018). https://bitcoinj.github.io/security-model. [Online; acessado 18-03-2018].

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A. E., Miller, A., Saxena, P., Shi, E., Sirer, E. G., Song, D., and Wattenhofer, R. (2016). On scaling decentralized blockchains. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC*, pages 106–125.

Ethereum (2018). https://www.ethereum.org/. [Online; acessado 18-03-2018].

Gennaro, R., Goldfeder, S., and Narayanan, A. (2016). Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS 2016*, pages 156–174.

Gervais, A., Capkun, S., Karame, G. O., and Gruber, D. (2014). On the privacy provisions of bloom filters in lightweight bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014*, pages 326–335.

Infura (2018). https://www.infura.io/. [Online; acessado 18-03-2018].

Jaxx (2018). https://www.jaxx.io/. [Online; acessado 18-03-2018].

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Disponível: https://bitcoin.org/bitcoin.pdf. [Online; acessado 18-03-2018].

Pedersen, T. P. (1991). A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques*, pages 522–526.

Pei, Q. and Ma, J. (2008). Ecc-based threshold digital signature scheme without a trusted party. In *2008 Conference on Computational Intelligence and Security*, pages 288–292.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data*, pages 557–564.