An Analysis of Consensus Algorithms for Blockchain aiming at the Implementation of Transparent Distributed Information Systems

Uma Análise de Algoritmos de Consenso para *Blockchain* visando à Implementação de Sistemas de Informação Distribuídos Transparentes

Carlo Kleber da Silva Rodrigues[1], Paulo Caetano da Silva[2]

[1]Centro de Matemática, Computação e Cognição (CMCC) – Universidade Federal do ABC (UFABC) – Santo André – SP – Brazil

[2]Programa de Pós-Graduação em Sistemas e Computação – Universidade Salvador (UNIFACS) – Salvador – BA – Brazil.

carlo.kleber@ufabc.edu.br, paulo.caetano@unifacs.br

Abstract. *Transparency in distributed-database information systems is only of real value when integration, sharing, consistency and integrity requirements are all firstly guaranteed. By its turn, the Blockchain technology is considered to be the most promising implementation solution to meet these requirements simultaneously. Within this context, this paper analyzes three important consensus algorithms for the Blockchain technology. This analysis includes a theoretical study and a comparative evaluation as well. The results allow us to better understand the most positive points and vulnerabilities of each analyzed algorithm. Finally, general conclusions and future work conclude this article.*

Keywords*: Blockchain. Transparency. Systems. Consensus. Algorithms. Information.*

Resumo. *A transparência em sistemas de informação com bases de dados distribuídas somente tem real valor se antes estiverem garantidos os requisitos de integração, compartilhamento, consistência e integridade. Por sua vez, a tecnologia Blockchain é considerada a mais promissora solução de implementação para atender a esses requisitos conjuntamente. Neste contexto, este artigo analisa três importantes algoritmos de consenso para a tecnologia Blockchain. Esta análise contempla um estudo teórico e uma avaliação comparativa. Os resultados advindos permitem melhor compreender os pontos mais positivos e as vulnerabilidades de cada algoritmo analisado. Por fim, conclusões gerais e trabalhos futuros encerram este artigo.*

Palavras-chave: *Blockchain. Transparência. Sistemas. Consenso. Algoritmos. Informações.*

1. Introdução

Um Sistema de Informação Distribuído (SID) pode ser caracterizado como um conjunto de processos concorrentes acessando recursos distribuídos, os quais podem ser compartilhados ou replicados, através de troca de mensagens em um ambiente de rede.

Esses sistemas são os mais utilizados atualmente, principalmente em ambientes que necessitam ter escalabilidade, alto desempenho, tolerância a falhas e heterogeneidade. A principal motivação para a construção de um sistema distribuído é o compartilhamento de recursos, e.g. impressoras, arquivos, páginas web, acesso a banco de dados distribuídos. Destarte, uma das características mais relevantes de um SID é possuir uma única base de dados lógica constituída por partições físicas geograficamente espalhadas e interconectadas por meio de redes de comunicação. Em contraste com o modelo centralizado de base física única, a dispersão geográfica se justifica fulcralmente por poder prover maior escalabilidade, eficiência e segurança ao sistema [Silberschatz; Korth; e Sudarshan 2010; Elmasri e Navathe 2015].

Ante a significativa profusão de emprego e implementação de SIDs para atender aos mais variados tipos de negócios da sociedade moderna, cinco importantes requisitos têm sido cada vez mais solicitados, passando a ser indispensáveis em alguns casos. Os dois primeiros requisitos referem-se aos conceitos de *integração* e *compartilhamento*, sendo explicados a seguir [Date 2003; Elmasri e Navathe 2015; Zhang e Jacobsen 2018].

Por *integração*, entenda-se a necessidade de eliminar qualquer redundância parcial ou total entre as informações. Por *compartilhamento*, tem-se que as informações podem ser compartilhadas entre diferentes usuários, no sentido de que diferentes usuários podem ter acesso às mesmas informações, com acesso concorrente e sem depender da localização geográfica. Esses dois requisitos relacionam-se ao desempenho do sistema, impactando na eficiência da realização de operações (e.g., inserção e remoção), bem como na otimização do espaço de memória física [Date 2003; Elmasri e Navathe 2015].

Os dois requisitos seguintes são a *consistência* e a *integridade* das informações armazenadas. Por *consistência*, entenda-se que, havendo alguma redundância justificada, deverá então haver uma garantia de que as informações redundantes não possuirão valores associados diferentes entre si. Já a *integridade* diz respeito a assegurar que as informações estejam realmente corretas. Esses dois requisitos se vinculam ao aspecto da *qualidade* da informação. Isto é, a preocupação é que a informação armazenada esteja correta e não que uma dada operação seja executada rapidamente [Date 2003; Elmasri e Navathe 2015; Zhang e Jacobsen 2018].

O quinto e último requisito diz respeito ao conceito de *transparência*. Um Sistema de Informação Distribuído Transparente (SIDT) deve, de maneira geral, permitir o acesso às informações armazenadas. Esse requisito foi originado pela Lei nº 12.527/2011, a qual regulamentou o direito de acesso às informações públicas. À época foram criados mecanismos formais que possibilitam, a qualquer pessoa, física ou jurídica, o recebimento de informações públicas dos órgãos e entidades. Esta Lei vale para os três Poderes da União, Estados, Distrito Federal e Municípios, inclusive para os Tribunais de Conta e Ministério Público. Entidades privadas sem fins lucrativos também são obrigadas a dar publicidade a informações referentes ao recebimento e à destinação dos recursos públicos recebidos [Weiland et al. 2018; Biffi; Nunes; e Maciel 2017].

Para a implementação das bases de dados dos SIDTs, as Tecnologias de Registros Distribuídos (do inglês, *Distributed Ledger Technologies* – DLTs) constituem a principal alternativa de solução, podendo atender plenamente aos requisitos de

integração, *compartilhamento*, *consistência*, *integridade* e *transparência* [Ioini e Pahl 2018]. As DLTs possuem, como alicerce fulcral, o peremptório emprego de algoritmos de consenso, os quais permitem que partes desconhecidas entre si, e até mesmo de interesses concorrentes, cheguem a um acordo sobre o estado atual e passado das informações armazenadas na base de dados distribuída [Das et al. 2018; Wang et al. 2018; Feng et al. 2019; Zhang e Jacobsen 2018].

Este contexto é a motivação para a realização deste artigo, cujo objetivo principal é analisar três importantes algoritmos de consenso encontrados na literatura para a tecnologia *Blockchain* [Nakamoto 2008], visando à implementação de SIDTs. A tecnologia *Blockchain* é reconhecida como uma das mais promissoras DLTs devido à maturidade tecnológica já alcançada, além de sua vertiginosa popularização na academia e indústria [Nguyen e Kim 2018; Ferrag et al. 2018]. Esta análise contempla um estudo teórico geral e, ainda, uma avaliação comparativa sobre escalabilidade e segurança, considerando trabalhos da literatura e experimentos baseados em modelagem analítica e simulação.

Ante o objetivo anunciado, este artigo tem então, como principal contribuição, a disponibilização de um único texto que provê um entendimento teórico de três importantes algoritmos de consenso para emprego na tecnologia *Blockchain* e, ainda, oferece uma análise comparativa dos mesmos, sob os vieses científico e tecnológico. Essa contribuição se constitui valiosa por prover subsídios relevantes para o desenvolvimento de possíveis projetos de SIDTs.

A organização do restante deste artigo é descrita a seguir. A Seção 2 explica brevemente a tecnologia *Blockchain* e traz a conceituação formal de algoritmos de consenso. Os trabalhos relacionados são discorridos na Seção 3. Na Seção 4, é realizado um estudo teórico dos três algoritmos de consenso escolhidos para análise. A partir de resultados da literatura e experimentos, a Seção 5 analisa comparativamente os três algoritmos. Por fim, conclusões gerais e trabalhos futuros constituem a Seção 6.

2. Fundamentos

2.1. Tecnologia *Blockchain*

A tecnologia *Blockchain* teve sua origem em 2008, junto com a proposta do sistema *Bitcoin* [Nakamoto 2008]. Apesar de a sua concepção original ter sido direcionada para a implementação de um sistema de pagamento eletrônico baseado em criptomoedas, essa tecnologia foi rapidamente identificada como de aplicabilidade mais abrangente [Dorri; Kanhere; e Jurdak 2017; Huckle et al. 2016]. Como exemplos, citam-se aplicações de controle de votos, de recursos em gestão orçamentária, de direitos de propriedades, de gestão de saúde, e no cenário de Internet das Coisas (do inglês, *Internet of Things* – IoT) [Dorri; Kanhere; e Jurdak 2017; Li e Liao 2017].

A base de dados distribuída implementada pela tecnologia *Blockchain* é estruturada como uma lista encadeada de blocos de transações ou, equivalentemente, uma cadeia de blocos de transações (do inglês, *blockchain*). As transações são realizadas pelos clientes do sistema. Os registros dessas transações e os correspondentes resultados advindos promovem a atualização da base de dados, que é controlada por uma rede de nós processadores independentes e interligados sob uma arquitetura *peer-*

to-peer (P2P). A *blockchain* (i.e., a cadeia de blocos) está sempre disponível para acesso concorrente por quaisquer participantes da rede, ou seja, é possível verificar todas as transações registradas na base de dados e, ainda, confirmar que as mesmas estão corretas, garantindo *transparência* e *compartilhamento* da base de dados [Antonopoulos 2017; Zhang e Jacobsen 2018; Ferrag et al. 2018].

Em particular, um *peer* da rede de interligação é denominado de *nó completo* (do inglês, *full node*) quando ele possui uma cópia de toda a base de dados, ou seja, possui uma réplica da base de dados [da Costa 2018]. Dada a existência das réplicas nos *nós completos* da rede P2P, a base de dados do sistema converge quando todas as réplicas existentes refletem o mesmo estado atual e passado das informações armazenadas, conferindo *integração* e *consistência* à base de dados [Antonopoulos 2017; Zhang e Jacobsen 2018; Ferrag et al. 2018].

Assinaturas digitais, criptografia e algoritmos de consenso sustentam conjuntamente a operação lógica da tecnologia *Blockchain*, garantindo especialmente *descentralização* de gerência e *integridade* da base de dados. As assinaturas digitais, por exemplo, garantem o não repúdio das transações realizadas. Já a criptografia é utilizada para interligar os blocos de transações entre si, como explicado a seguir. A identificação de cada bloco é feita pelo *hash* criptográfico de seu cabeçalho. Como se trata de uma lista encadeada, cada bloco faz referência a apenas um bloco anterior, chamado de bloco *pai*. Essa referência é feita usando o *hash* do cabeçalho do bloco *pai*, que está contido dentro do cabeçalho de cada nó. Ou seja, cada bloco contém o *hash* do cabeçalho de seu *pai* dentro de seu próprio cabeçalho [Antonopoulos 2017; Zhang e Jacobsen 2018; Silva e Rodrigues 2016; Ferrag et al. 2018].

A sequência de *hashes* que liga cada bloco ao seu *pai* estabelece o caminho de volta até o primeiro bloco do sistema, denominado de *bloco gênese*. A mudança da identidade de um bloco qualquer da *blockchain* geraria um efeito cascata da mudança da identidade de todos os blocos subsequentes a ele. Esse efeito cascata constitui a segurança da base de dados com relação à imutabilidade das informações já registradas [Ferrag et al. 2018; Garay; Kiayias; e Leonardos 2015; Rodrigues 2017].

Por sua vez, os algoritmos de consenso, como detalhado mais adiante, permitem estabelecer uma espécie de acordo sobre quando um bloco de transações pode ser considerado válido e, então, adicionado à *blockchain*. Esse acordo tem influência direta na forma como o processo denominado de *mineração* do bloco é realizado. A *mineração* é usualmente um processo matemático que existe para fins de validação do bloco. Esse processo pode ser executado por um único *peer*, denominado de *nó minerador*, ou por um grupo de *peers*, denominado de *mining pool*. Os *mineradores* (ou *mining pools*) são, portanto, os nós processadores responsáveis por manter a *blockchain*. Após a sua *mineração*, o bloco é então enviado em *brodacast* pela rede para que os *nós completos* possam individualmente atualizar as suas respectivas réplicas da base de dados, estabelecendo a convergência da *blockchain* [Garay; Kiayias; e Leonardos 2015; Silva e Rodrigues 2016].

Blockchain é considerada como uma tecnologia disruptiva, que muito ainda impactará a forma de realização de negócios em todo o mundo. Acreditando neste potencial, grandes empresas e organizações, como IBM, JP Morgan, Facebook e DHL, têm feito significativos investimentos em pesquisa. Essa conjuntura fez com que as

bases de dados implementadas na tecnologia *Blockchain*, usualmente também referenciadas simplesmente como *plataformas Blockchain*, fossem classificadas como *públicas*, *em consórcio* ou *privadas* [Nguyen e Kim 2018; Zhang e Jacobsen 2018; Neudecker e Hartenstein 2018].

As plataformas *públicas* consideram que a base de dados não está sob controle de nenhum grupo de indivíduos ou organizações. É permitida a participação de qualquer usuário interessado na utilização do sistema, inclusive no próprio processo de *mineração* dos blocos, gerando um alto grau de *descentralização*. Nas plataformas *em consórcio*, há um controle por parte de um grupo de organizações, as quais estabelecem regras para participação no sistema. Por fim, as plataformas *privadas* admitem o controle de exclusivamente uma única organização. Nesses dois últimos tipos, a *descentralização* é parcial, pois há um controle explícito da participação dos nós [Nguyen e Kim 2018; Zhang e Jacobsen 2018; Neudecker e Hartenstein 2018].

Neste trabalho de pesquisa, o foco de discussão está restrito a plataformas *públicas*, ficando as plataformas *em consórcio* e *privadas* como objeto de estudo para trabalhos futuros.

2.2. Algoritmos de Consenso

Os algoritmos de consenso permitem que partes desconhecidas entre si, e até mesmo de interesses concorrentes, cheguem a um acordo sobre o estado atual e passado das informações armazenadas na base de dados. Esses algoritmos buscam, assim, prover uma solução para o clássico *Problema dos Generais Bizantinos* [Lamport; Shostak; e Pease 1982].

No contexto de sistemas computacionais, as partes desconhecidas são pensadas como nós processadores independentes que estão interconectados por meio de uma rede de comunicação sob arquitetura P2P. Exigir a aprovação declarada individual de todos os nós para realizar a adição de uma nova informação à base de dados não é razoável, pois isso poderia naturalmente comprometer o desempenho global do sistema [Nguyen e Kim 2018; Zhang e Jacobsen 2018].

Existem duas principais abordagens para solucionar esta questão e cada uma delas origina um dado tipo de algoritmo de consenso. A primeira abordagem consiste em decidir sobre uma forma de *provar* que um nó (ou um conjunto de nós) pertencente a rede é suficientemente confiável para adicionar um bloco de informações à base de dados, considerando que a *prova* oferecida seja aceita e, se desejável, possível de ser verificada por todos os demais nós da rede. Esta abordagem origina os algoritmos *proof-based consensus* [Nguyen e Kim 2018; Zhang e Jacobsen 2018]

O algoritmo de consenso *Proof of Work (*PoW) é do tipo *proof-based consensus*. Este algoritmo foi utilizado na tecnologia *Blockchain* quando essa tecnologia foi proposta no ano de 2008 [Nakamoto 2008]. Sob o PoW, os nós *mineradores* precisam *provar* que realizaram um significativo esforço computacional, fornecendo a solução para um desafio criptográfico. Na literatura podem ser encontrados vários outros algoritmos do tipo *proof-based consensus*, com diferentes processos de *mineração*. Como exemplos, citam-se os seguintes: *Proof of Stake* (PoS), *Proof of Burn* (PoB), *Proof of Space* (PoSp), e *Proof of Luck* (PoL) [Nguyen e Kim 2018; Aliaga et al. 2018].

A outra abordagem para solucionar a questão do consenso é considerar que o nó a adicionar o bloco de informações é aquele que detém o apoio da maioria dos outros nós da rede, expressa por meio de uma votação. Neste caso, se um nó deseja adicionar um bloco à base de dados, i.e., deseja ser um *minerador*, o seguinte requisito deve ser atendido: o nó deve ter garantido que ao menos *Q* nós estão de acordo. O parâmetro *Q* é um limiar que varia dependendo do sistema, devendo corresponder a mais que 50% de todos os nós da rede. Esta ideia é a base para os algoritmos *voting-based consensus*. Os algoritmos *Ripple*, *Stellar* e *Chain* são alguns exemplos que seguem essa abordagem [Nguyen e Kim 2018; Zhang e Jacobsen 2018].

Os algoritmos *proof-based consensus* e *voting-based consensus* podem ser indistintamente utilizados em plataformas *públicas*, *em consórcio* ou mesmo *privadas*. Todavia, devido à intrínseca concepção de projeto, os algoritmos *proof-based consensus* têm sido mais empregados em sistemas de significativo número de participantes, enquanto que aqueles do tipo *voting-based consensus* são mais usados quando há um número restrito de participantes.

Pode-se sintetizar a compreensão acima com a seguinte afirmação: *Quando há muitos participantes, é melhor ter representantes que provem ser suficientemente confiáveis para adicionar informações (i.e., proof-based consensus). Por outro lado, quando há poucos participantes, é melhor que a maioria sempre decida sobre as informações a serem adicionadas (i.e., voting-based consensus)* [Nguyen e Kim 2018; Zhang e Jacobsen 2018; Aliaga et al. 2018].

O escopo de discussão neste trabalho está restrito aos algoritmos do tipo *proof-based consensus*, ficando aqueles do tipo *voting-based consensus* para trabalhos futuros.

3. Trabalhos Relacionados

Optando-se por uma sequência de citação preferencialmente cronológica e sem a intenção de constituir uma lista exaustiva, esta seção discorre sobre alguns dos mais recentes e relevantes trabalhos da literatura que contribuem, direta ou indiretamente, para o objetivo deste artigo, buscando precipuamente ofertar ao leitor uma visão do estado da arte desta área de pesquisa.

Sompolinsky e Zohar (2015) e Karame (2016) discutem sobre a escalabilidade da tecnologia *Blockchain*, considerando o sistema de criptomoedas *Bitcoin*. Para tanto, levam-se em consideração a quantidade de informações, o tempo de verificação de transações, a otimização das estruturas de dados e dos algoritmos, o tamanho dos blocos da *blockchain* (i.e., da cadeia de blocos) e a diversidade de aplicações. Os resultados advindos permitem conjecturar sobre uma nova geração de aplicações baseadas na tecnologia *Blockchain*.

Dorri, Kanhere e Jurdak (2017) propõem uma versão mais otimizada da tecnologia *Blockchain* para o desenvolvimento de aplicações em IoT. Essa versão tenciona diminuir o custo de processamento criptográfico, otimizar a banda e diminuir as latências de transmissão. Emprega-se um paradigma de camadas funcionais, permitindo que bases de dados locais centralizadas se comuniquem com bases públicas distribuídas. Os blocos são *minerados* localmente e adicionados à *blockchain* de sua

camada, que posteriormente se torna consistente com à *blockchain* de camadas públicas superiores.

Danzi et al. (2017) descrevem arquiteturas gerais e protocolos de sincronização para nós de clientes IoT usando a tecnologia *Blockchain*. Consideram-se diferentes capacidades de transmissão da rede, níveis de segurança e características intrínsecas de ambientes *wireless*. Nesse trabalho são modelados e caracterizados analiticamente os tráfegos gerados pelos protocolos de sincronização e, usando simulações, são avaliados o consumo de energia e o custo de banda para convergência da base de dados.

Aliaga et al. (2018) discutem sobre alguns dos algoritmos de consenso mais conhecidos, e.g., PoW, PoS e PoB, e suas aplicações. Ainda que seja um estudo de caráter preliminar e limitado, o trabalho se reveste de relevância por conseguir apresentar uma visão teórica concisa sobre o assunto. Semelhante a esse trabalho, os pesquisadores Nguyen e Kim (2018) e Wang et al. (2018) também realizam uma discussão geral sobre algoritmos de consenso e suas aplicações. Porém, o detalhamento e a abrangência observados nestes dois últimos trabalhos são mais destacáveis. Esses três trabalhos em conjunto constituem importantes *surveys* sobre o assunto.

Sob uma perspectiva algorítmica comparativa, Bach, Mihaljevic e Zagar (2018) analisam alguns algoritmos de consenso bem conhecidos, e.g., PoS, *Stellar* e *Ripple*. O objetivo principal é mensurar o desempenho individual de cada proposta e, assim, conjecturar sobre a longevidade de seu emprego e eventual domínio em termos de maior aceitação. Embora existam restrições nos experimentos realizados, devido à impossibilidade de uso de cenários reais e da complexidade matemática da modelagem, os resultados obtidos trazem importantes evidências sobre o desempenho desses algoritmos, constituindo importantes subsídios para o desenvolvimento de projetos reais.

Huang, Ma e Zhang (2018) analisam o algoritmo do tipo *voting-based consensus* denominado de *Raft*, cujo foco é plataformas *em consórcio* e *privadas*. A análise é feita usando simulações e modelagem analítica. Os resultados obtidos mostram um satisfatório desempenho do algoritmo nos cenários investigados, além de evidenciarem a adequação do modelo analítico desenvolvido para fins de projetos e monitoração da operação das plataformas.

Hao et al. (2018) propõem um método para avaliar o desempenho de algoritmos de consenso nos sistemas *Ethereum* e *Hyperledger Fabric*, respectivamente. Para tanto, tem-se a realização de uma análise de latência e capacidade de execução de transações por unidade de tempo. Os resultados obtidos sugerem precipuamente uma superioridade do algoritmo *Practical Byzantine Fault Tolerance* (PBFT), que é do tipo *voting-based consensus*, sobre o algoritmo PoW, que é do tipo *proof-based consensus*.

Por fim, Xue et al. (2018) propõem um novo algoritmo do tipo *voting-based consensus*, denominado de *Proof of Contribution* (PoC). Esse algoritmo destina-se especialmente à atividade de *mineração* no sistema *Bitcoin*. A análise então realizada baseia-se em modelos analíticos e simulações. Os resultados obtidos evidenciam principalmente a otimização do consumo de energia e a satisfatória resiliência a ataques de fraudes, em comparação com o conhecido algoritmo PoW.

Ante o exposto e mesmo diante da diversidade de trabalhos já publicados na literatura, a contribuição e a diferenciação deste artigo se materializam pela realização de uma análise comparativa de três importantes algoritmos de consenso, do tipo *proof-based consensus*, para plataformas *públicas* com vistas à implementação de SIDTs. O escopo delimitado e o caráter comparativo, sob os vieses científico e tecnológico, seguramente ressaltam o relativo ineditismo deste trabalho de pesquisa.

4. Estudo Teórico

Esta seção traz um estudo teórico de três algoritmos de consenso do tipo *proof-based consensus*: *Proof of Work* (PoW), *Proof of Stake* (PoS) e *Proof of Activity* (PoA). Embora se reconheça a existência de outras propostas relevantes na literatura, a escolha destes três algoritmos se justifica peremptoriamente pela ampla aceitação dos mesmos pela indústria e/ou academia. Além disso, é ponto pacífico que estes três algoritmos constituem um arcabouço conceitual importante para as demais propostas que se seguiram ao longo tempo. Portanto, entender estes três algoritmos permite mais proficuamente poder analisar e avaliar outras propostas, além de inspirar novas ideias.

Explica-se que o estudo a ser apresentado tende a utilizar, muitas vezes, exemplos de sistemas de criptomoedas. A razão para essa exemplificação considerando esse tipo de negócio se dá por haver uma maior disponibilidade de trabalhos conhecidos na literatura. Em que pese essa relativa restrição quanto ao tipo de negócio, é certo que as conclusões advindas ainda se constituem em subsídios relevantes para a análise de sistemas genéricos, sob os vieses científico e tecnológico.

4.1. Algoritmo *Proof of Work*

Para realizar o estudo teórico do algoritmo *Proof of Work* (PoW), seguimos sua definição original apresentada quando da proposta da tecnologia *Blockchain* [Nakamoto 2008]. Essa definição considera seu emprego específico na implementação do conhecido sistema de pagamento eletrônico *Bitcoin*.

Cada bloco a ser inserido na *blockchain* (i.e., na cadeia de blocos) é constituído por um cabeçalho seguido por uma lista de transações, conforme descrição na Tabela 1. O cabeçalho tem 80 bytes e o tamanho médio de uma transação é de pelo menos 250 bytes. O número médio de transações em um bloco é 500 [Antonopoulos 2017].

Tabela 1. Estrutura do bloco

Tamanho	Campo	Descrição
4 bytes	*Tamanho do bloco*	Tamanho do bloco, em bytes, que segue este campo.
80 bytes	*Cabeçalho*	Alguns campos formam o cabeçalho. Vide Tabela 2.
1 a 9 bytes	*Contador de transações*	Número de transações existentes no bloco.
Variável	*Transações*	Transações armazenadas no bloco.

Conforme Tabela 2, o campo cabeçalho possui três grupos de metadados. O primeiro grupo se refere ao número de versão do protocolo e ao *hash* do bloco anterior (i.e., bloco *pai*). No segundo grupo, há os campos *Dificuldade*, *Instante de criação* e *nonce*. Estas informações se relacionam ao processo de *mineração*, detalhado mais adiante. Por último, o terceiro grupo é a *Raiz da árvore de Merkle*, que possui um resumo de todas as transações armazenadas no bloco.

Cada bloco é identificado pelo *hash* de seu próprio cabeçalho. O algoritmo de *hash* criptográfico utilizado é o SHA256 [Gilbert e Handschuh 2004]. Como já mencionado, cada bloco faz referência a apenas um bloco anterior, também chamado de bloco *pai*. Essa referência se implementa por meio do campo *hash* do bloco anterior, que está contido dentro do cabeçalho de cada nó. Ou seja, cada bloco contém o *hash* do cabeçalho de seu *pai* dentro de seu próprio cabeçalho [Ferrag et al. 2018; Garay; Kiayias; e Leonardos 2015; Rodrigues 2017].

Toda transação realizada entre clientes é enviada pela rede P2P em *broadcast*, sendo verificada e reencaminhada por nós intermediários até alcançar um *minerador* (ou *mining pool*). Cada *minerador* trabalha agrupando as transações que chegam até ele em blocos. Após formar um bloco, o *minerador* realiza a sua *mineração* [Silva e Rodrigues 2016]. Matematicamente, a *mineração* consiste em determinar, por tentativas sucessivas, o valor de *nonce* (i.e., o *golden nonce*) que satisfaz o desafio criptográfico sintetizado pela desigualdade: SHA256(SHA256(*data*)) $\leq$ *Dificuldade*. O parâmetro *data* é o cabeçalho do bloco de transações, e o parâmetro *Dificuldade* está definido na Tabela 2.

Tabela 2. Estrutura do cabeçalho

Tamanho	Campo	Descrição
4 bytes	*Versão*	Número de versão do protocolo
32 bytes	*Hash anterior*	*Hash* do bloco anterior (i.e., bloco *pai*) na *blockchain*.
32 bytes	*Raiz da árvore de Merkle*	Constitui um *resumo* das transações existentes no bloco, que é detalhado mais adiante.
4 bytes	*Instante de criação*	Instante de criação do bloco.
4 bytes	*Dificuldade*	Valor utilizado para estabelecer o nível de dificuldade da *mineração*. Este valor é ajustado consensualmente para garantir um tempo mínimo de *mineração* de bloco.
4 bytes	*nonce*	Valor encontrado como solução no processo de *mineração*. Após encontrado, este valor é denominado *golden nonce*.

O parâmetro *Dificuldade* tem seu valor periodicamente recalculado para assegurar que, diante das variações do sistema ao longo do tempo, a adição de blocos à *blockchain* ocorra a cada *X* unidades de tempo, independentemente do número de *mineradores* e do número de transações submetidas no sistema. No caso específico do sistema *Bitcoin*, o valor de *X* é igual a 10 minutos. Após *minerado*, o bloco é então enviado via *broadcast* pela rede P2P para que os *nós completos* atualizem as suas bases de dados locais. Todo *minerador* que consegue *minerar* um bloco recebe uma recompensa em *bitcoins*, que funciona como um estímulo para o ininterrupto trabalho de *mineração* [Rodrigues 2017].

Conforme Tabela 1, um bloco da *blockchain* armazena todas as transações nele existentes no campo *transações*. Além disso, conforme Tabela 2, todo bloco também guarda um *resumo* de todas as suas transações no campo *Raiz da árvore de Merkle*. A computação desse *resumo* é explicada a seguir.

A *árvore de Merkle*, também conhecida como árvore binária de *hash*, é construída a partir dos seus vértices *folhas* [Berman; Karpinski; e Nekrich 2007]. Cada folha da árvore guarda o *hash* de uma das transações existentes no bloco. São então computados recursivamente os *hashes* de pares de vértices da árvore, a partir dos vértices *folhas*, até que se obtenha um único *hash*, o qual corresponde ao *resumo* guardado na raiz da árvore. O algoritmo criptográfico utilizado é o SHA256, que é aplicado duas vezes, sendo também conhecido por *double*-SHA256. Para exemplificação, a Figura 1 ilustra uma árvore de *Merkle* para um bloco de quatro transações: *A*, *B*, *C*, *D*. No caso de o número de transações ser ímpar, a última transação deve ser duplicada, obtendo uma *árvore binária cheia* [Cormen; Leiserson; e Rivest 2009].

Para terminar esta subseção, menciona-se que *Bitcoin Cash*, *Bitcoin Gold*, *Litecoin*, *Primecoin* e *Ethereum* são exemplos de outros sistemas cujas bases de dados são implementadas na tecnologia *Blockchain* usando o algoritmo de consenso PoW [Bano et al. 2017; Silva e Rodrigues 2016; Nguyen e Kim 2018]. Embora o entendimento conceitual geral seja o mesmo do sistema *Bitcoin*, existem naturalmente diferenças entre esses sistemas, as quais residem principalmente em detalhes quanto aos campos constituintes dos blocos e no tipo de processamento matemático que deve ser realizado pelo *minerador* ou *mining pool*.

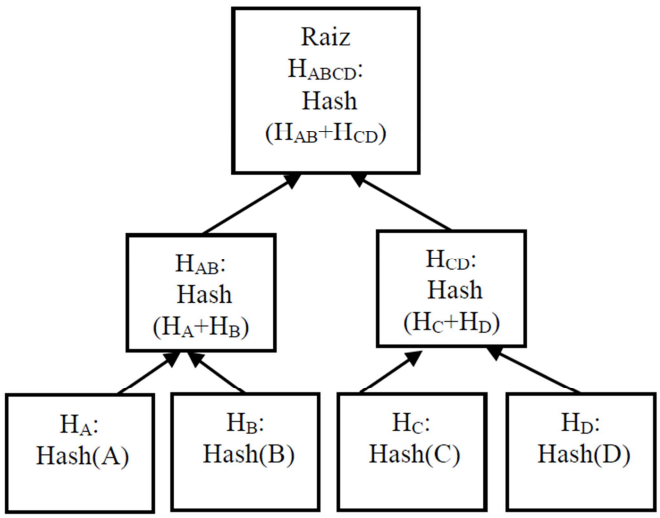

Figura 1. Árvore de Merkle para 4 transações: A, B, C e D.

4.2. Algoritmo *Proof of Stake*

Diferentemente do caminho seguido para a apresentação do algoritmo PoW na Seção 4.1, que se norteou pela implementação específica do sistema *Bitcoin*, aqui consideramos uma apresentação direcionada a aspectos conceituais funcionais do algoritmo, independentemente de um sistema específico. Para o leitor interessado em detalhes de sistemas específicos, citam-se alguns exemplos ao final.

O algoritmo *Proof of Stake* (PoS) tem sua motivação de projeto na injustiça que pode emergir pelo emprego do algoritmo PoW: se alguns poucos *mineradores* ou *mining pools* concentrarem uma grande capacidade computacional, o processo de *mineração* de blocos passa a ser controlado por eles, tornando o sistema centralizado.

A ideia base do algoritmo PoS é então usar o poder de participação (do inglês, *stake*) que o nó *minerador* detém com respeito ao sistema para determinar a probabilidade de ele poder *minerar* o próximo bloco da *blockchain* (i.e., da cadeia de blocos). Ou seja, quem é mais importante para o sistema (i.e., detém maior poder), maior probabilidade tem de *minerar* o próximo bloco da *blockchain*.

Para ilustrar essa ideia de poder de participação de forma simples, considere um sistema de pagamento eletrônico qualquer baseado em criptomoedas. Nesse caso, o poder de participação de um nó *minerador* pode ser mensurado pela quantidade de criptomoedas que este *minerador* dispõe. Por exemplo, se há um total de b criptomoedas, distribuídas entre todos os *mineradores* do sistema, e um certo *minerador M* possui a criptomoedas, então a probabilidade de o *minerador M minerar* o próximo bloco é diretamente proporcional ao valor a/b, sendo $a \leq b$.

Possuir a maior probabilidade dentre todos os *mineradores* da rede constitui, assim, a prova que o *minerador M* oferece para dizer-se suficientemente confiável para adicionar um bloco à base de dados, i.e., ser o *minerador* do próximo bloco a ser adicionado à *blockchain*. Decidido o *minerador*, ele deve verificar as transações por ele coletadas, realizar o agrupamento delas em um bloco, enviar o bloco via *broadcast* pela rede (para que os *nós completos* atualizem suas bases de dados locais) e, por fim, receber uma recompensa.

Em que pese o fato de ainda haver um processamento matemático a ser realizado para o cômputo da probabilidade de escolha do *minerador* e, ainda, da construção do bloco em si, o grau de complexidade associado é bem menor que aquele do desafio criptográfico previsto pelo PoW. Esse processamento matemático, que constitui a *mineração* do PoS, difere em acordo com a implementação do sistema real considerado. O mais importante é que, sendo um processo matemático de baixa complexidade, o poder de processamento computacional deixa de ser peremptório para a escolha do *minerador* que irá adicionar o próximo bloco à *blockchain*.

Para encerrar esta subseção, mencionamos que *Nextcoin*, *Blackcoin, Ethereum* e *Cardano* são alguns exemplos de outros sistemas cujas bases de dados são implementadas na tecnologia *Blockchain* usando o algoritmo de consenso PoS [Bano et al. 2017; Nxt Wiki 2016; Zheng et al. 2018; Nguyen e Kim 2018]. Ainda que possuindo o mesmo arcabouço básico conceitual de projeto, diferenças entre esses sistemas de fato existem, residindo especialmente nos detalhes quanto aos campos constituintes dos blocos e no próprio processamento matemático que deve ser realizado para a seleção do *minerador* e a construção do bloco a ser adicionado à *blockchain*. Essa última observação é análoga àquela feita anteriormente para o PoW.

4.3. Algoritmo *Proof of Activity*

Semelhante à forma de apresentação do algoritmo PoS na última subseção, aqui também optamos por um caminho que discorre mais sobre aspectos conceituais funcionais, independentes de sistemas específicos e realçando-se mais a concepção do projeto. Como antes, para aquele leitor mais interessado em detalhes de implementação de sistemas reais, citamos alguns exemplos ao final.

Por um lado, é verdade que o algoritmo PoW pode levar o sistema a perder sua apregoada descentralização devido a uma eventual concentração do poder computacional nas mãos de alguns poucos *mineradores* ou *mining pools*. Todavia, por outro lado, também é verdade que o algoritmo PoS pode eventualmente fazer com que os mesmos *mineradores* (ou *mining pools*) sejam sempre aqueles escolhidos, pois quem possui maior poder de participação acaba tendo mais chances de ser escolhido para *minerar* novos blocos, ganhando recompensas que aumentarão ainda mais o poder de participação e, portanto, as chances de novamente ser escolhido.

A proposta do algoritmo *Proof of Activity* (PoA) [Bentov et al. 2014] combina então os algoritmos PoW e PoS com a finalidade precípua de evitar que as duas situações supracitadas venham a ocorrer. Por essa razão, diz-se que o PoA é na verdade um *algoritmo híbrido*. Sua forma de operação é explicada a seguir.

Inicialmente, os *mineradores* competem entre si para ver quem primeiro consegue *minerar* um bloco sem transações (i.e., vazio), contendo apenas o cabeçalho, usando o algoritmo PoW. Após encontrar um *nonce* adequado, o *minerador* então envia o bloco *minerado* e o *golden nonce* correspondente para *N* outros *mineradores*, selecionados a partir do algoritmo PoS. Ou seja, os *N mineradores* escolhidos são aqueles *mineradores* que detêm maior poder de participação no sistema.

Os *N mineradores* selecionados irão então verificar se o *golden nonce* informado está correto para o bloco *minerado* que a eles foi enviado. Se todos os *N mineradores*

confirmarem a corretude do *golden nonce*, o bloco passa a ser válido e pode ter transações nele armazenadas. O último dos *N mineradores* que fez a validação é quem armazena as transações no bloco. Após ter as transações armazenadas, o bloco é então enviado em *brodacast* pela rede para que os *nós completos* possam individualmente atualizar as suas respectivas réplicas da base de dados, estabelecendo a convergência da *blockchain*. A recompensa pela *mineração* do bloco neste caso é dividida entre o *minerador* inicial e os *N mineradores* que verificaram a corretude do *golden nonce* calculado. O valor específico de *N* é tido como um parâmetro de configuração da implementação do sistema real.

Sob PoA, mesmo que um *minerador* (ou *mining pool*) no sistema detenha mais de 50% de poder de participação no sistema ou que venha eventualmente a concentrar de forma exclusiva um grande poder computacional, ele ainda assim precisará da validação de *N* outros *mineradores*. Além disso, comparativamente aos algoritmos PoW e PoS, há um estímulo diferenciado no sentido de que os *mineradores* devem permanecer ativos e trabalhar cooperativamente para receber a recompensa relativa ao bloco *minerado*.

Além do PoA, existem outras ideias de algoritmos *híbridos* na literatura [dos Santos e Swan 2018]. Por exemplo, há uma variante que consiste em um algoritmo PoW que tem a complexidade do cálculo do *nonce* sendo inversamente proporcional ao tempo que o *minerador* detém um certo poder de participação no sistema [King e Nadal 2012]. *TwinsCoins*, *PeerCoin* e *Decred* são outros exemplos de sistemas cujas bases de dados são implementadas na tecnologia *Blockchain* usando algoritmos de consenso *híbridos*. Por fim, ainda que possuindo a mesma ideia de aproveitar o melhor dos algoritmos PoW e PoS, diferenças entre esses sistemas reais naturalmente existem, residindo principalmente nos detalhes com respeito aos campos constituintes dos blocos e na própria estratégia de combinação do PoW com o PoS [King e Nadal 2012; Duong et al. 2018; Nguyen e Kim 2018; Decred 2018]. Essa última observação é análoga àquela feita para o algoritmo PoS.

4.4. Síntese Teórica Comparativa

Com base no exposto nas Seções 4.1, 4.2 e 4.3, respectivamente, esta seção apresenta três conclusões parciais, constituindo uma síntese teórica comparativa entre os três algoritmos de consenso em estudo.

Primeiro, o consumo de energia elétrica do PoW termina sendo maior que aqueles do PoA e do PoS, respectivamente. Esse maior consumo de energia elétrica do PoW se dá pela necessidade de uso ininterrupto do *hardware* em sua máxima capacidade de processamento para a resolução mais rápida possível de um complexo desafio criptográfico, pois apenas quem primeiro encontra o *golden nonce* do bloco da vez é que tem o direito de receber a recompensa por ele. Já a *mineração* do PoS não envolve disputas diretas entre os *mineradores* quanto ao poder de processamento, e o PoA é um algoritmo híbrido cujo processamento matemático para descoberta do *golden nonce* é de baixa complexidade, consumindo bem menos energia elétrica que o PoW.

Segundo, o PoW é, dentre os três algoritmos estudados, aquele que mais depende da capacidade de processamento de *hardware* para conseguir realizar a *mineração* e, por conseguinte, receber a recompensa devida. Em sintonia com o

mencionado no último parágrafo, quanto mais rápido é o *hardware* utilizado, mais chances se tem de encontrar o valor do *golden nonce*, concluindo a *mineração* do bloco. Se o *hardware* de um *minerador* não é suficientemente competitivo em relação ao *hardware* de outros *mineradores* que competem com ele, esse *minerador* nunca receberá uma recompensa e estará apenas consumindo energia elétrica. Nesse quesito, PoS leva vantagem sobre PoW e PoA por não possuir qualquer desafio criptográfico a ser vencido. Por sua vez, PoA aparece em segundo lugar por ter um desafio criptográfico de menor complexidade que aquele do PoW.

Terceiro, durante a construção da *blockchain* (i.e., da cadeia de blocos) é possível ocorrer o chamado *forking*. Isso ocorre quando dois *mineradores* (ou *mining pools*) encontram independentemente dois *golden nonces* simultaneamente, ou seja, dois blocos são *minerados* ao mesmo tempo. Isso resulta na bifurcação da *blockchain*, i.e., duas ramificações distintas e concorrentes surgem. Se isso ocorre, a convergência se dará quando uma dessas ramificações se tornar mais longa que a outra. Nesse caso, a ramificação mais curta é desconsiderada e passa a valer apenas aquela que se torna mais longa. Explica-se que a ocorrência de mais de dois blocos *minerados* em simultâneo não é considerada devido à sua insignificante probabilidade de ocorrência.

Pela própria definição do algoritmo de consenso, o *forking* tem maior probabilidade de ocorrência no PoW que nos outros dois algoritmos em estudo. Sob o PoW, os nós são independentes e competem diretamente entre si, não sendo possível garantir que apenas um dos *mineradores* (ou *mining pools*) irá encontrar o *golden nonce* do próximo bloco da *blockchain.* Além disso, ainda há a questão do tempo de propagação do bloco pela rede, o que poderia fazer com que diferentes blocos, *minerados* proximamente no tempo, fossem ambos considerados o próximo bloco da *blockchain* por distintos *nós completos* espaçados geograficamente.

Sob PoS, a escolha do *minerador* se dá em função de sua participação no sistema. Mesmo que o poder de participação fosse idêntico entre distintos *mineradores*, pode-se pensar em um sorteio aleatório para decidir quem iria *minerar* o próximo bloco. Ou seja, teoricamente, seria muito difícil ocorrer o *forking*. Por fim, sob PoA, após a descoberta do *golden nonce*, ainda se faz necessária uma confirmação por *N* distintos *mineradores*, o que torna o *forking*, apesar de também ainda possível, bastante improvável de ocorrer.

5. Avaliação: Escalabilidade e Segurança

Esta seção realiza uma avaliação sobre escalabilidade e segurança dos três algoritmos de consenso em discussão neste artigo. Para maior facilidade de entendimento dos resultados, tem-se a organização a seguir: a Seção 5.1 realiza uma avaliação com base em trabalhos da literatura; por sua vez, a Seção 5.2 discute a avaliação dos algoritmos usando modelagem analítica e simulação.

Análogo ao já mencionado na Seção 4, a avaliação a ser apresentada a seguir utiliza exemplos de sistemas de criptomoedas, isto decorre de se ter maior disponibilidade de trabalhos conhecidos na literatura para esse tipo de negócio. Todavia, é certo que as conclusões advindas ainda se constituem em subsídios relevantes para a análise de sistemas genéricos, sob os vieses científico e tecnológico.

5.1. Avaliação com Base em Resultados da Literatura

Questão 1: Escalabilidade

Esta questão se refere ao tamanho da rede e a eficiência observada conforme ocorre seu crescimento, podendo ser resumida pelas seguintes duas perguntas: Qual a capacidade da rede considerando o número de transações que a mesma pode processar por unidade de tempo? Qual o impacto do aumento do número de *mineradores* (ou *mining pools*) nessa capacidade?

No caso do PoW, o número de transações por unidade de tempo é limitado pelo valor do parâmetro *Dificuldade*, que define a complexidade de resolução do desafio criptográfico (vide Subseção 4.1, Tabela 2), em conjunto com o número de transações contidas em um mesmo bloco, i.e., tamanho do bloco (vide Subseção 4.1, Tabela 1).

Diminuir o valor do parâmetro *Dificuldade* para ter uma *mineração* mais rápida não é indicado, pois significaria o comprometimento do equilíbrio sistêmico de se ter um mesmo tempo médio de adição de blocos à *blockchain* (i.e., cadeia de blocos). Aumentar o tamanho do bloco tampouco é indicado, pois significaria o aumento no tempo de processamento do mesmo devido ao maior volume de informações, o que impactaria também no tempo médio de adição de blocos à *blockchain*, comprometendo o equilíbrio sistêmico. Por último, aumentar o número de *mineradores* não influencia o número médio de transações por unidade de tempo, pois, como já informado, o tempo médio de adição de blocos à *blockchain* deve ser mantido o mesmo. Assim sendo, PoW não é uma solução escalável [Nguyen e Kim 2018; Wang et al. 2018].

Para PoS, conforme sua definição (vide Seção 4.2), o número de transações processadas por unidade de tempo não está limitado pela complexidade de resolução de um algoritmo criptográfico. Nesse sentido, aumentar o tamanho do bloco e/ou o número de *mineradores* da rede pode aumentar o número de transações processadas por unidade de tempo. Para tanto, deve-se apenas observar quanto de poder de participação um nó processador deve demonstrar possuir para ser habilitado como *minerador*. Isso para evitar algum eventual comprometimento da estabilidade sistêmica. Assim sendo, a solução PoS é tida como escalável [Nguyen e Kim 2018; Wang et al. 2018].

Para PoA, conforme sua definição (vide Seção 4.3), o aumento de número de *mineradores* pode implicar uma maior facilidade para localizar os *N* nós processadores que vão validar o bloco vazio já *minerado*, que na sequência é adicionado à *blockchain*. Isso poderia levar a um maior número de transações por unidade de tempo. Aumentar o tamanho do bloco pode também aumentar o número de transações processadas por unidade de tempo, pois o bloco ao ser *minerado* está sempre inicialmente vazio, i.e., não há impacto no tempo de *mineração*. Essas duas alternativas devem naturalmente atentar para que não haja um comprometimento da estabilidade sistêmica. Ainda, uma terceira alternativa seria curiosamente considerar a própria diminuição do valor de *N*, o que também pode elevar o número de transações por unidade de tempo, pois menos confirmações da corretude do *golden nonce* seriam exigidas. Esta última alternativa não seria, todavia, aconselhável por poder comprometer a segurança do processo de verificação. Em todo caso, esse cenário geral demonstra uma maior escalabilidade com relação ao cenário de PoW, mas ainda inferior ao cenário de PoS devido à existência do desafio criptográfico [Nguyen e Kim 2018; Wang et al. 2018].

Para fins de exemplificação, a Tabela 3 traz alguns valores numéricos relacionados à escalabilidade de sistemas reais. São informados o algoritmo de consenso, a capacidade de transações por segundo e, ainda, o tempo médio de finalização de uma transação (i.e., desde a sua submissão até a sua aprovação final) [Bach; Mihaljevic; e Zagar 2018; Zheng et al. 2017; Schwarz 2018].

Tabela 3. Escalabilidade de sistemas

Sistema	Algoritmo de consenso	Número transações por segundo	Tempo de finalização de transação (em minutos)
Bitcoin	PoW	7	78
Bitcoin Cash	PoW	60	60
Litecoin	PoW	56	30
Cardano	PoS	7	5
Ethereum	PoS	25	6

Questão 2: Segurança

Esta questão se refere à possibilidade de controle indesejável da base de dados por parte de um *minerador* (ou *mining pool*) *desonesto*, podendo ser sintetizada pela seguinte pergunta: Quanto poder um *minerador* (ou *mining pool*) *desonesto* deve ter para conseguir fraudar as informações ou, equivalentemente, substituir blocos da *blockchain*?

O algoritmo PoW foi apresentado no sistema *Bitcoin* como sendo uma solução segura de consenso contra qualquer ataque que envolvesse menos de 51% do poder computacional total de *mineração* da rede [Nakamoto 2008]. Todavia, foi descoberto mais tarde que, por meio de uma estratégia maliciosa, um ataque com mais de 25% do poder total de computação já seria suficiente [Eyal e Sirer 2014]. Curiosamente, sabe-se ainda que, em tese, seria possível evitar ataques de até 33%, mas a solução em si ainda não foi possível de ser apresentada [Eyal e Sirer 2014].

Selfish mining é o nome da estratégia maliciosa citada acima. Essa estratégia baseia-se na ideia de produzir o *forking*, como explicado a seguir. Um *mining pool desonesto* faz a *mineração* de blocos e adiciona-os a uma ramificação *secreta* da *blockchain*. Em paralelo, os *mineradores honestos* adicionam blocos à uma ramificação *honesta*, que é de conhecimento público. O *mining pool desonesto* monitora o estado do sistema e divulga a sua ramificação *secreta* quando esta se torna mais longa que a ramificação *honesta*. Nesse caso, a ramificação *secreta* passa a ser de conhecimento público e as informações nela armazenadas passam a ser aceitas no sistema, substituindo aquelas da ramificação *honesta* [Eyal e Sirer 2014].

Note que *selfish mining* faz com que *mineradores honestos* desperdicem seus recursos computacionais resolvendo desafios criptográficos, tendo em vista que os correspondentes blocos *minerados* são posteriormente ignorados. Além disso, os *mineradores honestos* tendem a dispender mais esforço computacional que os *mineradores* que adotam o *selfish mining*, o que resulta em uma recompensa mais

favorável aos *mineradores desonestos* que aos *honestos*. Ou seja, há um incentivo tangível para que *mineradores honestos* se unam em algum momento ao *selfish mining* [Eyal e Sirer 2014; Neudecker e Hartenstein 2019].

Menciona-se ainda que dois refinamentos de modelagem surgiram após a proposta em 2008 do PoW aplicado pela tecnologia *Blockchain* [Nakamoto 2008]. Primeiro, o número de blocos *minerados* por um suposto *minerador desonesto* não segue uma distribuição de Poisson, mas sim uma distribuição binomial negativa [Grunspan e Pérez-Marco 2017]. Segundo, a chegada de blocos *minerados* à *blockchain* não segue simplesmente um processo homogêneo de Poisson [Bowden et al. 2018], como modelado originalmente [Nakamoto 2008]. Esses refinamentos subsequentes terminam reforçando o entendimento de que, em verdade, não há modelos perfeitos e que as avaliações não podem prescindir de ter-se conjuntamente análises analíticas, simulações e medições em sistemas reais.

Ante o exposto, é imediato concluir que a possibilidade de fraude de fato existe sob PoW. Para mitigar essa possibilidade, pode-se buscar garantir uma *mineração honesta* superior a 51% do poder de processamento total da rede e, ainda, uma espera suficientemente longa de tempo para a aceitação definitiva de um bloco de informações na *blockchain*. Isso certamente dificultaria o *mining pool desonesto* de conseguir ter sua ramificação *secreta* mais longa que a ramificação *honesta* [Mata e Rodrigues 2018; Neudecker e Hartenstein 2019].

Nos casos dos algoritmos de consenso PoS e PoA, respectivamente, a segurança sistêmica pode ser discutida como segue. Para PoS, a segurança reside em garantir que haja uma permutação frequente dos *mineradores* que detém o maior poder de participação no sistema, impedindo a existência de monopólios ou oligopólios (vide Seção 4.2). Para tanto, pode-se pensar na realização de um sorteio para escolher Y *mineradores* entre os G *mineradores* de maior poder de participação, devendo ser $Y << G$ [Bach; Mihaljevic; e Zagar 2018].

Para PoA, a segurança reside também na garantia de uma frequente permutação dos N nós processadores que validam as *minerações* realizadas por um outro primeiro *minerador* (vide Seção 4.2). Como sugerido para PoS, pode-se analogamente pensar na realização de um sorteio para escolher N *mineradores* entre G *mineradores* aptos a realizar a validação, devendo ser $N << G$. Em todo caso, de forma comparativa, é preciso dizer que PoA oferece menor nível de segurança que PoS pela existência do desafio criptográfico (em PoA), que pode suscitar o aparecimento de ataques usando uma estratégia inspirada no *selfish mining* [Bach; Mihaljevic; e Zagar 2018].

5.2. Experimentos

5.2.1. Modelagem Analítica

A modelagem analítica proposta a seguir tem o objetivo de estimar a probabilidade de uma transação x, recém submetida no sistema, vir a ser inserida já no próximo bloco adicionado à *blockckchain* (i.e., à cadeia de blocos). Isso implica, em tese, que a transação é finalizada no tempo médio de *mineração* do bloco.

A estimativa dessa probabilidade, ainda que de forma aproximada, permite uma discussão direta relacionada ao desempenho do sistema, estando intimamente ligada à

Questão 1 da Seção 5.1 (i.e., Escalabilidade). A diferença é que agora tem-se um enfoque quantitativo probabilístico, em vez de um enfoque conceitual argumentativo.

Para a modelagem pretendida, considere inicialmente o cenário ilustrado na Figura 2. Neste cenário, admitem-se a aplicação do algorimo PoW e a observação do sistema no instante *t* de tempo em regime estacionário. Existem *T* transações submetidas pelos clientes, *N mineradores* (de mesma capacidade) e a indicação do último bloco (com contorno pontilhado) a ser adicionado à *blockchain*. Ainda, seja *S* o tempo médio de *mineração* de bloco [Antonopoulos 2017; Ricci et al. 2016].

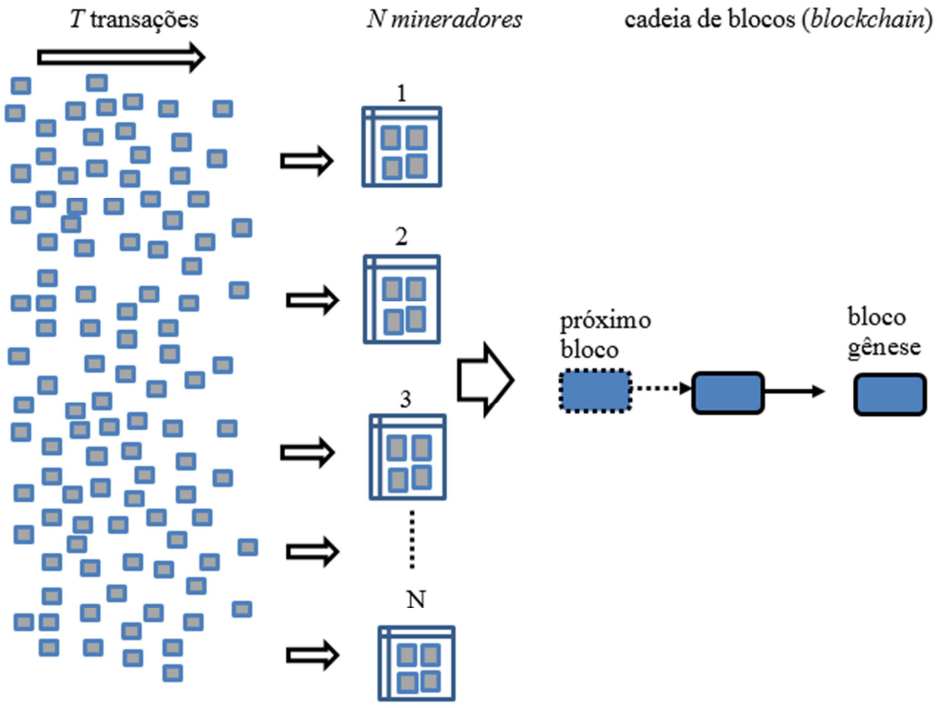

Figura 2. Cenário de análise para o algoritmo PoW

Considere agora o sistema sem memória, com interindependência de eventos, e que cada transação submetida é coletada exclusivamente por apenas um *minerador*. Segue, então, que a probabilidade a ser estimada resulta da multiplicação da probabilidade de a transação *x* estar sendo *minerada* pela probabilidade do próximo bloco adicionado ser aquele que contém a transação *x*. Essa formulação pode ser vista na Equação 1:

$$P(x \text{ está no próximo bloco}) = \left[\frac{N \times (nº\ de\ transações\ por\ bloco)}{T}\right] \times \left[\frac{1}{N}\right] \qquad (1)$$

A partir da Equação 1, pode-se concluir que o simples aumento do número de *mineradores* no sistema não tem qualquer efeito sobre a probabilidade de a transação *x* estar no próximo bloco da *blockchain*. Isso se justifica porque apenas 1 bloco é inserido por vez na *blockchain*, independentemente da quantidade de *mineradores*. Uma primeira alternativa possível para aumentar essa probabilidade seria aumentar o tamanho do bloco, ou seja, admitir um maior número de transações por bloco. Uma

segunda alternativa possível seria diminuir o número de transações submetidas pelos clientes no sistema, o que naturalmente não seria desejável.

Para fins de exemplificação, a Figura 3 fornece informações quantitativas com respeito ao número de transações de um sistema baseado no algoritmo PoW. Neste caso, os dados exemplificam a operação do sistema *Bitcoin,* lembrando que o número de transações por bloco é 500, em média (vide Seção 4).

Por sua vez, a Figura 4, concebida a partir da utilização da Equação 1, traz dados sobre a influência ocasionada pelo aumento do número de transações contidas em um bloco. Considerando os dados mostrados nas Figuras 3 e 4, é possível notar que, a partir do ano de 2012 (vide Figura 3), a solução de aumentar o tamanho do bloco não seria efetiva, pois os valores estimados de probabilidade (vide Figura 4) pouco variariam. Isso se dá pelo crescimento vertiginoso do número de transações submetidas no sistema (vide Figura 3).

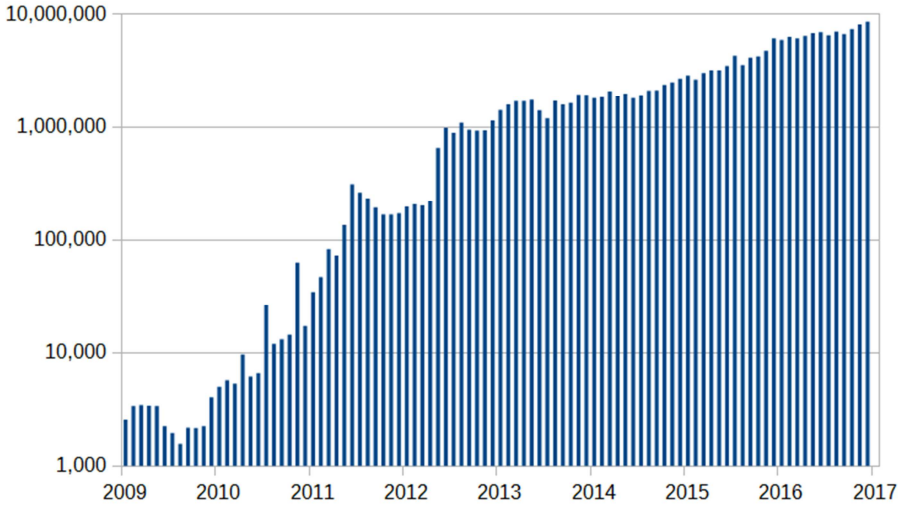

Figura 3. Número de transações por mês considerando o sistema *Bitcoin*

Fonte: de Zhitelew (2017)

Para que o aumento do tamanho do bloco surtisse algum efeito, seria necessário aumentar o tamanho do bloco na mesma ordem de grandeza do aumento do número de transações submetidas. Isso seria inexequível, pois poderia colateralmente afetar o tempo médio de *mineração S* devido ao aumento do volume de informações a ser processado, o que ocasionaria a necessidade de outros ajustes sistêmicos como, por exemplo, a alteração do valor do parâmetro *Dificuldade* (vide discussão na Seção 5.1).

Consideramos a definição de escalabilidade como a capacidade de o sistema prover a mesma eficiência (tempo médio de espera para a finalização da transação) ao usuário, independentemente do número de transações totais submetidas (este número tem aumentado significativamente ao longo do tempo, conforme mostrado na Figura 3). Portanto, aumentar o tamanho do bloco ou diminuir o grau de dificuldade da mineração não se constituem em soluções adequadas. Ante o exposto, é possível concluir que o PoW é uma solução pouco resiliente frente a um aumento significativo do número de transações submetidas no sistema, que é a realidade corrente (vide Figura 3). Ou seja, PoW não é uma solução escalável, ratificando a conclusão alcançada na Seção 5.1.

Nos casos dos algoritmos PoS e PoA, respectivamente, a Equação 1 não se aplica devido à distinta forma de operação (vide Seção 4). As equações de probabilidade de PoS e PoA, respectivamente, devem estabelecer alguma função de proporcionalidade direta com o número de *mineradores* de suficiente poder de participação no sistema e/ou tamanho do bloco. Os algoritmos PoS e PoA são soluções escaláveis, ratificando as conclusões alcançadas na Seção 5.1. Devido à maior complexidade da derivação analítica, menciona-se que as equações de probabilidade de PoS e PoA, respectivamente, são deliberadamente deixadas como objeto de estudo detalhado para trabalhos futuros.

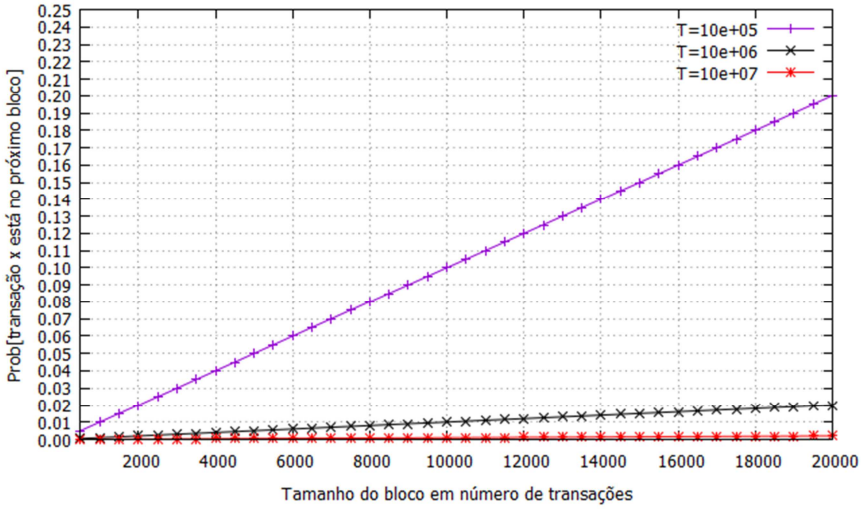

Figura 4. Probabilidade de a transação *x* estar no próximo bloco *minerado*

5.2.2. Simulação

Esta seção busca avaliar a segurança sistêmica, a qual já foi alvo da discussão teórica apresentada na Questão 2 da Seção 5.1. A diferença é que aqui é feita uma análise quantitativa por meio do emprego de um modelo de simulação. Esse modelo visa a estimar a probabilidade de ocorrer uma fraude no sistema. Essa fraude consiste na tentativa de substituição de um bloco *B*, adicionado à *blockchain* (i.e., à cadeia de blocos) por *mineradores* (ou *mining pools*) *honestos*, por um outro bloco *F*, *minerado* por *mineradores* (ou *mining pools*) *desonestos*.

Considere o algoritmo PoW aplicado na *blockchain* do sistema *Bitcoin* e, ainda, o cenário ilustrado na Figura 5. Nesta figura, ilustra-se a ocorrência de um *forking* que não é motivado por uma tentativa de fraude. Como explicado na Seção 4.4, isso ocorre quando dois *mineradores* (ou *mining pools*) *honestos* distintos enviam em simultâneo diferentes versões daquele que deveria ser o próximo bloco da *blockchain*: bloco *B* ou bloco *C*.

Os nós *completos* vão receber uma ou outra versão primeiramente. Cada nó *completo* considera a primeira versão recebida, mas vai também criar uma ramificação em sua *blockchain* local para adicionar a outra versão do bloco recebida posteriormente. Passam a existir duas ramificações: uma para a primeira versão (bloco *B*) e a outra para a segunda versão (bloco *C*), como ilustrado na Figura 5.

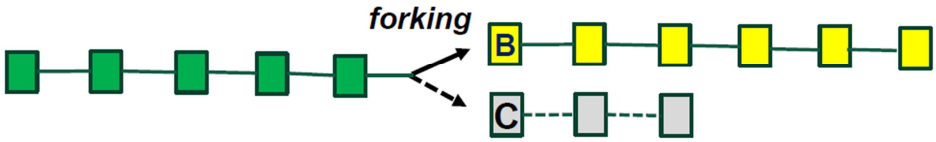

Figura 5. Cenário com ocorrência de *forking*

A seleção entre as duas ramificações ocorre quando os *golden nonces* subsequentes são encontrados, fazendo com que uma das ramificações se torne mais longa que a outra. Como já mencionado (vide Seção 4.4), uma transação para ser considerada válida deve pertencer à ramificação mais longa, sendo a ramificação mais curta então desprezada, como ilustrado na Figura 6.

Figura 6. Cenário após resolução de *forking*

No entanto, em um cenário de tentativa de fraude, a competição entre as duas ramificações pode ser entendida como uma *corrida* para *adição* de blocos à *blockchain*, como explicado a seguir. Um *minerador desonesto* deseja realizar uma fraude, substituindo um bloco *B* legítimo, adicionado por um *minerador honesto*, por um bloco fraudado *F* (vide Figura 7). Para isso, o *minerador desonesto* espera até que *z* blocos subsequentes a *B* sejam adicionados à *blockchain*. Naquele instante, o *minerador honesto* estará *minerando* o bloco *z*+1. Explica-se que essa espera de *z* blocos é o tempo que o cliente da aplicação tolera esperar para ter sua transação confirmada no sistema.

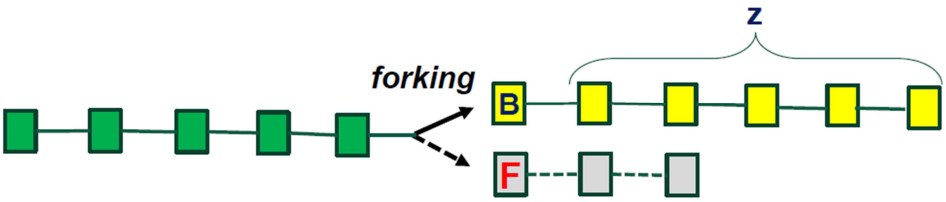

Figura 7. Cenário com ocorrência de tentativa de fraude

O *minerador desonesto* então inicia a *mineração* explícita de blocos e tenta ser mais rápido que o *minerador honesto* para compensar a desvantagem de *z* blocos, i.e., fazer a sua ramificação (aquela do bloco *F*) se tornar mais longa que a ramificação contendo o bloco legítimo *B*. Um detalhe importante é que o *minerador desonesto minera* blocos secretamente em taxa compatível com sua capacidade para que, ao iniciar a competição explícita com o *minerador honesto*, já possua blocos *minerados* para inserir no sistema.

Para analisar essa competição, admite-se o seguinte modelo parametrizado: com probabilidade p, o próximo bloco *minerado* da *blockchain* é do *minerador honesto*, e com probabilidade $q = (1 - p)$, o próximo bloco *minerado* é do *minerador desonesto*. A probabilidade p se refere à força de *mineração honesta* no sistema, e a probabilidade q é

a força da *mineração desonesta* no sistema. A chegada de blocos *minerados* à *blockchain* segue um processo de Poisson com parâmetro μ. O modelo estima então a probabilidade de fraude para um dado valor de *z*, considerando que existe um percentual de *mineradores honestos* (probabilidade *p*) e um percentual de *mineradores desonestos* (probabilidade *q*) em competição no sistema. Note que quanto maior é *z*, menor deve ser a probabilidade de fraude, porém maior é o tempo de espera para a confirmação de uma transação [Nakamoto 2008; Mata e Rodrigues 2018].

A simulação realizada emprega o Tangram-II [de Souza e Silva; Figueiredo; e Leão 2009]. Tangram-II é um ambiente de modelagem de sistemas computacionais concebido na Universidade Federal do Rio de Janeiro, com a participação da Universidade da Califórnia em Los Angeles nos EUA. Os modelos são definidos em termos de objetos que interagem entre si por mensagens, podendo ser resolvidos analiticamente ou via simulação.

Cabe destacar que os resultados de simulação apresentados a seguir têm intervalos de confiança de 95% que estão dentro do limite de 5% dos valores estimados, tendo sido consideradas 30 execuções (rodadas) com um tempo de simulação de 21.000 minutos cada. Para fins de concisão e objetividade, menciona-se ainda que apenas os resultados mais relevantes para suporte às conclusões atingidas são apresentados no que se segue.

Os resultados obtidos na simulação estão na Figura 8. São seis condições de análise. Nas três primeiras, considera-se p = 0,6 e varia-se z para determinar a probabilidade de haver fraude. Nas três seguintes, repete-se a mesma análise para p = 0,8. Os partir dos resultados dessa figura, conclui-se, em geral e como esperado, que maiores valores de p e z tendem a fornecer maior segurança ao sistema, pois a probabilidade de fraude claramente diminui.

Por exemplo, para z = 0,6 e aumentando-se z de 1 para 6, saímos de uma probabilidade de fraude de 29,49e-02 para 10,70e-02, respectivamente. Isso representa uma redução significativa de 63,71%. Todavia, é importante dizer que aumentos arbitrários de p e z sem alguma avaliação prévia podem também resultar inócuos. Por exemplo, aumentar z de 1 para 3, considerando p = 0,6, não produz qualquer otimização significativa. Enfim, como diretriz para um projeto real, deve-se admitir valores de p acima de 0,5 (i.e., força da *mineração honesta*) e valores de z que observem a tolerância do cliente em esperar pela confirmação de sua transação no sistema, lembrando que um bloco é adicionado à *blockchain* a cada 10 minutos no sistema *Bitcoin*. Note que essa discussão está em acordo com a conclusão alcançada na Questão 2 da Seção 5.1.

Para os casos dos outros dois algoritmos de consenso, i.e. PoS e PoA, respectivamente, os modelos de simulação correspondentes devem considerar a permutação frequente dos *mineradores* que detém o maior poder de participação (sob PoS) ou dos N nós processadores que validam as *minerações* realizadas por um outro primeiro *minerador* (sob PoA), conforme já discutido na Questão 2 da Seção 5.1. A expectativa é que a probabilidade de fraude seja então desprezível. Modelos de simulação específicos para esses dois algoritmos são deliberadamente deixados como objeto de estudo detalhado para trabalhos futuros.

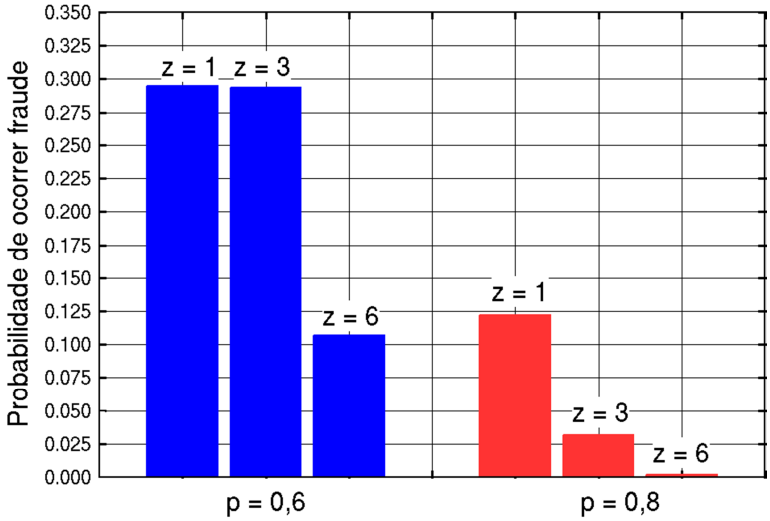

Figura 8. Probabilidade de Fraude

Ante o exposto, é possível concluir que, em termos de segurança, o algoritmo PoW apresenta-se como o menos seguro, seguido pelo algoritmo PoA e, depois, como o mais seguro dentre os três, o algoritmo PoS. A inferioridade de PoA com relação ao PoS se dá pela existência do desafio criptográfico (em PoA), que pode suscitar o aparecimento de ataques inspirados na estratégia maliciosa denominada de *selfish mining*, conforme já mencionado na Questão 2 da Seção 5.1.

6. Conclusões Gerais e Trabalhos Futuros

Este artigo realizou uma análise sobre os algoritmos de consenso *Proof of Work* (PoW), *Proof of Stake* (PoS) e *Proof of Activity* (PoA) para emprego na tecnologia *Blockchain* visando à implementação de Sistemas de Informação Distribuídos Transparentes (SIDTs). Essa análise contemplou, inicialmente, um estudo teórico e, na sequência, uma avaliação comparativa sobre escalabilidade e segurança, usando modelagem analítica e simulação.

A partir dos resultados obtidos, foi possível concluir que o algoritmo PoS é a solução mais recomendável para implementação de SIDTs. O algoritmo PoA, por sua vez, foi identificado como a segunda melhor solução e, por último, teve-se o algoritmo PoW. Chegou-se a essa conclusão porque as avaliações sobre consumo de energia elétrica, dependência de poder de processamento, possibilidade de bifurcações na *blockchain* (i.e., na cadeia de blocos), escalabilidade e segurança, respectivamente, evidenciaram a superioridade de PoS com relação ao PoA e, mais ainda, ao PoW.

Por fim, como possíveis trabalhos futuros, sugerimos os dois caminhos a seguir. Primeiro, em caráter complementar à pesquisa aqui realizada, proceder a novas análises considerando modelos analíticos e simulações mais abrangentes, incluindo algoritmos de consenso do tipo *voting-based consensus* e plataformas *em consórcio* e *privadas*, bem como realizar medições em sistemas reais [Wang et al. 2018]. Segundo, realizar análises comparativas entre Tecnologias de Registros Distribuídos diferentes da *Blockchain*, tais como: *Tangle*, *Hashgraph* e *Sidechain* [Ioini e Pahl 2018].

Referências

Aliaga, Y. E. M et al. (2018). Avaliação de mecanismos de consenso para blockchains em busca de nova estratégia mais eficiente e segura. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2018), Natal, Rio Grande do Norte, Brasil.

Antonopoulos, M. (2017). Mastering Bitcoin: Programming the Open Blockchain. 2nd Edition. Sebastopol, California: O´Reilly Media.

Bach, L. M.; Mihaljevic, B.; and Zagar, M. (2018). Comparative Analysis of Blockchain Consensus Algorithms. In: International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2018), Opatija, Croatia.

Bano, S. et al. (2017). SoK: Consensus in the Age of Blockchains. CoRR, abs/1711.03936. Available at: http://arxiv.org/abs/1711.03936 . Accessed on: Dec. 19th, 2018.

Bentov, I. et al. (2014). Proof of Activity: extending bitcoins´s proof of work via proof of stake. ACM SIGMETRICS Performance Evaluation Review, v. 42, n. 3, pp. 34-37.

Berman, P.; Karpinski, M.; and Nekrich, Y. (2007). Optimal trade-off for Merkle tree traversal. Theoretical Computer Science, v. 372, n. 1, pp. 26-36.

Biffi, G.; Nunes, E.; e Maciel, C. (2017). Método TRADIN: transparência governamental com a cooperação da economia digital. Workshop de Transparência em Sistemas (WTranS/CSBC 2017), São Paulo, SP, Brasil.

Bowden, R. et al. (2018). Block arrivals in the Bitcoin blockchain. CoRR, abs/1801.07447. Available at: http://arxiv.org/abs/1801.07447 . Accessed on: Dec. 24th, 2018.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. (2009). Introduction to Algorithms. 3rd Edition. Cambridge, Massachussets: MIT Press.

da Costa, L. et al. (2018). Securing light clients in blockchain with DLCP. International Journal of Network Management, e2055. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2055. Accessed on: Dec. 15th, 2018.

Danzi, P. et al. (2017). Analysis of the Communication Traffic for Blockchain Synchronization of IoT Devices. CoRR, abs/1711.00540. Disponível em: https://arxiv.org/abs/1711.00540v1. Acessado em: 2 de dezembro de 2019.

Das, G. et al. (2018). On the Security of Blockchain Consensus Protocols. In: (eds.) Information Systems Security. ICISS 2018. Lecture Notes in Computer Science, v. 11281, pp. 465-480. Springer, Cham.

Date, C. J. (2003). An Introduction to Database Systems, Pearson, 8th Edition.

de Souza e Silva, E.; Figueiredo, R.; and Leão, R. (2009). The TANGRAM-II integrated modeling environment for computer systems and networks. ACM SIGMETRICS Performance Evaluation Review, v. 36, n. 4, pp. 64-69.

Decred (2018). Decred Documentation. Available at: https://docs.decred.org/ . Accessed on: Dec. 20th, 2018.

Dorri, A.; Kanhere, S.; and Jurdak, R. (2017). Towards an Optimized Blockchain for IoT. In: International Conference on Internet-of-Things Design and Implementation (IoTDI´17), Pittsburgh, PA, USA.

dos Santos, R. and Swan, M. (2018). PoW, PoS, & Hybrid protocols: A Matter of Complexity? CoRR, abs/1805.08674. Available at: http://arxiv.org/abs/1805.08674 . Accessed on: Dec. 20th, 2018.

Duong, T. et al. (2018). TwinsCoin: A Cryptocurrency via Proof-of-Work and Proof-of-Stake. In: Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, Incheon, Republic of Korea.

Elmasri, R. and Navathe, S. B. (2015). Fundamentals of Database Systems, Pearson, 7th Edition.

Eyal, I. and Sirer, E. G. (2014). Majority is not enough: bitcoin mining is vulnerable. In: International conference on financial cryptography and data security, pp. 436-452. Springer, Berlin, Heidelberg.

Feng, Q. et al. (2019). A survey on privacy protection in blockchain system. Journal of Network and Computer Applications, v. 126, pp. 45-58.

Ferrag, M. A. et al. (2018). Blockchain Technologies for the Internet of Things: Research Issues and Challenges. IEEE Internet of Things Journal. Available at: https://ieeexplore.ieee.org/document/8543246. Accessed on: Dec. 14th, 2018.

Garay, J.; Kiayias, A.; and Leonardos, N. (2015). The bitcoin backbone protocol: analysis and applications. LNCS, Springer, Berlin, Heidelberg, v. 9057, pp. 281-310.

Gilbert, H. and Handschuh, H. (2004). Security analysis of SHA-256 and sisters. Lecture Notes in Computer Science (LNCS), Springer, Berlin, Heidelberg, v. 3006, pp. 175-193.

Grunspan, C. and Pérez-Marco, R. (2017). Double spend races. CoRR, abs/1702.02867. Available at: https://arxiv.org/abs/1702.02867. Accessed on: Jan. 3rd, 2019.

Hao, Y. et al. (2018). Performance Analysis of Consensus Algorithm in Private Blockchain. In: IEEE Intelligent Vehicles Symposium (IV´18), Changshu, Suzhou, China.

Huang, D.; Ma, X.; and Zhang, S. (2018). Performance Analysis of the Raft Consensus Algorithm for Private Blockchains. arXiv preprint arXiv:1808.01081. Disponível em: https://arxiv.org/abs/1808.01081. Acessado em: 11 de dezembro de 2018.

Huckle, S. et al. (2016). Internet of Things, blockchain and shared economy applications. Procedia Computer Science, v. 98, pp. 461-466.

Ioini, N. E. and Pahl, C. (2018). A Review of Distributed Ledger Technologies. In: (eds.) On the Move to Meaningful Internet Systems. OTM 2018 Conferences. OTM 2018. Lecture Notes in Computer Science, vol 11230. Springer, Cham. Available at: https://link.springer.com/chapter/10.1007%2F978-3-030-02671-4_16 . Accessed on: Dec. 29th, 2018.

Karame, G. O. (2016). On the security and scalability of bitcoin´s blockchain. In: ACM Conference on Computer and Communications Security (CCS´16), Vienna, Austria.

King, S. and Nadal, S. (2012). PPcoin: peer-to-peer crypto-currency with proof-of-stake. Available at: https://peercoin.net/assets/paper/peercoin-paper.pdf. Accessed on: Dec. 20th, 2018.

Lamport, L.; Shostak, R.; and Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programmimg Languages and Systems (TOPLAS), v. 4, n. 3, pp. 382-401.

Li, I-C and Liao, T-C. (2017). A Survey of blockchain security issues and challenges. International Journal of Network Security, v. 19, n. 5, pp.653-659, 2017.

Mata, R. Z. A. e Rodrigues, C. K. S. (2018). Uma análise competitiva entre as tecnologias Tangle e Blockchain para projetos de aplicações IoT. In: Workshop de Desempenho em Sistemas Computacionais e de Comunicação (WPerformance/CSBC 2018), Natal, RN, Brasil.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Disponível em: https://bitcoin.org/bitcoin.pdf . Acessado em: 2 de janeiro de 2019.

Neudecker, T and Hartenstein, H. (2018). Network Layer Aspects of Permissionless Blockchains. IEEE Communications Surveys & Tutorials. Available at: https://doi.org/10.1109/COMST.2018.2852480 . Accessed on: Jan. 5th, 2019.

Neudecker, T and Hartenstein, H. (2019). Short Paper: An Empirical Analysis of Blockchain Forks in Bitcoin. Available at: https://dsn.tm.kit.edu/bitcoin/forks/blockprop.pdf . Accessed on: Jan. 5th, 2019.

Nguyen, G-T and Kim, K. (2018). A Survey about Consensus Algorithms used in Blockchain. Journal of Information Processing Systems, v. 14, n. 1, pp. 101-128.

Nxt Wiki. (2016). Whitepaper: Nxt. Available at: https://nxtwiki.org/wiki/Whitepaper:Nxt#Proof_of_Stake . Accessed on: Dec. 20th, 2018.

Ricci, S. et al. (2016). Dinâmica das transações do Bitcoin: uma abordagem quantitativa. In: Workshop de Desempenho em Sistemas Computacionais e de Comunicação (WPerformance/CSBC 2016), Porto Alegre, RS, Brasil.

Rodrigues, C. K. S. (2017). Sistema Bitcoin: uma análise da segurança das transações. iSys: Revista Brasileira de Sistemas de Informação, v. 10, n. 3, pp. 5-23.

Schwarz, M. (2018). Transactions speeds: which crypto is the fastest?. Available at: https://www.abitgreedy.com/transaction-speed/. Accessed on: Dec. 12th, 2018.

Silberschatz, A.; Korth, H. F.; and Sudarshan, S. (2010). Database Systems Concepts, McGraw-Hill Science/Engineering/Math, 6th Edition.

Silva, G. A. e Rodrigues, C. K. S. (2016). Mineração individual de bitcoins e litecoins no mundo. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2016), Niterói, Rio de Janeiro, Brasil.

Sompolinsky, Y. and Zohar, A. (2015). Secure high-rate transaction processing in bitcoin. LNCS, Springer, Berlin, Heidelberg, v. 8975, pp. 507-527.

Wang, W. et al. (2018). A Survey on Consensus Mechanisms and Mining Management in Blockchain Networks. CoRR, abs/1805.02707. Available at: http://arxiv.org/abs/1805.02707. Accessed on: Jan. 2nd, 2019.

Weiland, C. et al. (2018). Integração e Análise das Despesas do Governo Federal. Workshop de Transparência em Sistemas (WTranS/CSBC 2018), Natal, RN, Brasil.

Xue, T. et al. (2018). Proof of Contribution: A Modification of Proof of Work to Increase Mining Efficiency. In: IEEE International Conference on Computer Software & Applications (COMPSAC'2018), Tokyo, Japan.

Zhang, K and Jacobsen, H-A. (2018). Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains. In: IEEE International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria.

Zheng, Z. et al. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA.

Zheng, Z. et al. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services (IJWGS), v. 14, n. 4, pp. 352-375.

Zhitelew, S. (2017). File: BTC number of transactions per month. Available at: https://commons.wikimedia.org/w/index.php?curid=40617621 . Accessed on: Dec. 26th, 2019.