Capítulo 3

Correntes de Blocos: Algoritmos de Consenso e Implementação na Plataforma Hyperledger Fabric

Gabriel Antonio F. Rebello[1], Gustavo F. Camilo[1], Leonardo G. C. Silva[1], Lucas Airam C. de Souza[1], Lucas C. B. Guimarães[1], Eduardo A. P. Alchieri[3], Fabíola Greve[2] e Otto Carlos M. B. Duarte[1]

[1] Universidade Federal do Rio de Janeiro - GTA/PEE/COPPE
[2] Universidade Federal da Bahia - GAUDI/DCC
[3] Universidade de Brasília - COMNET/CIC

Resumo

A corrente de blocos (blockchain) é uma tecnologia disruptiva que deve revolucionar o nosso modo de viver, trabalhar e negociar. A corrente de blocos é considerada a tecnologia que vai revolucionar a Internet, provendo uma camada de confiança distribuída. Assim como a Internet permite hoje a transferência de arquivos, a tecnologia de corrente de blocos permitirá a Internet de Valores, na qual é possível a transferência sem intermediários de ativos, tais como dinheiro, ações, propriedade intelectual, votos, etc. A corrente de blocos em sua essência é uma simples estrutura de dados imutável que armazena registros de transações e que é replicada em todos os participantes da rede. Os algoritmos de consenso que determinam a forma como se tomam as decisões coletivas e como são obtidas as finalizações das tarefas são a parte fundamental e mais complexa de corrente de blocos. Existem dezenas de algoritmos de consenso com diferentes características, dentre elas gasto energético, desempenho, tolerância a falha de equipamentos, robustez a ataques de conluio, escalabilidade, etc. Assim, este capítulo foca em algoritmos de consenso para ajudar o projetista de corrente de blocos a escolher o consenso apropriado para suas aplicações. Diversos algoritmos de consenso são apresentados especificando suas características, suas vantagens, suas desvantagens e finalidades. Outro foco deste minicurso é uma parte prática que mostra a construção de uma aplicação de corrente de blocos usando a plataforma Hypeledger Fabric, que é uma plataforma de corrente de blocos privada, modular e de código aberto, além de ser a mais utilizada pelas empresas. A aplicação provê garantia de análise forense em um ambiente de funções virtualizadas de rede.

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ e FAPESP (2015/24514-9, 2015/24485-9 e 2014/50937-1).

3.1. Introdução

As cidades inteligentes buscam o uso de novas tecnologias para prover facilidades que melhorem a qualidade de vida do cidadão. No entanto, há várias brechas de segurança em relação a privacidade, integridade e confidencialidade dos dados que precisam ser tratadas. Um exemplo são provedores de serviços centralizados, como Facebook e Google, que proveem pouca informação aos cidadãos sobre o uso, armazenamento e análise de dados pessoais [Xie et al. 2019]. Um arcabouço usando corrente de blocos é uma potencial solução para prover uma sistema de comunicação seguro [Biswas and Muthukkumarasamy 2016, Bano et al. 2017].

As tecnologias de corrente de blocos de primeira geração, baseadas no Bitcoin [Nakamoto 2008], e de segunda geração, baseadas nos contratos inteligentes do Ethereum [Wood 2014] já revolucionaram o mundo atual ao criar uma camada de confiança para transferências seguras de ativos e execução segura de contratos. No entanto, para a Internet do Futuro os desafios de escalabilidade são ainda maiores, pois prevê-se que em 2025 o uso de dispositivos de Internet das coisas (*Internet of Things* - IoT) gere até 3 trilhões de dólares em valor de mercado [Machina Research 2016] e atinja a marca de 75 bilhões de dispositivos conectados [Statista 2018], que atenderão a diversas aplicações, desde redes veiculares tolerantes a atraso até serviços críticos como saúde eletrônica (*e-Health*) [Zhang et al. 2018, Esposito et al. 2018], cidades inteligentes (*smart cities*) [Xie et al. 2019, Rahman et al. 2019, Mora et al. 2018] e redes elétricas inteligentes (*smart grids*) [Pieroni et al. 2018].

Este capítulo procura focar em algoritmos de consenso e se baseia em trabalhos anteriores dos autores tais como artigos publicados em eventos nacionais e internacionais e, principalmente, os minicursos apresentados no Simpósio Brasileiro de Redes de Computadores "Segurança na Internet do Futuro: Provendo Confiança Distribuída através de Correntes de Blocos na Virtualização de Funções de Rede" [Rebello et al. 2019b] e "Blockchain e a Revolução do Consenso sob Demanda" [Greve et al. 2018].

Além disso, este capítulo objetiva motivar o leitor quanto à importância da tecnologia de corrente de blocos para fornecer segurança às telecomunicações e aos ambientes distribuídos. Assim, esse capítulo dedica uma parte significativa à realização de uma atividade prática que mostra o desenvolvimento de duas correntes de blocos baseadas que executam contratos inteligentes na plataforma Hyperledger Fabric. As correntes de blocos mitigam ataques em uma arquitetura de fatiamento de redes que provê o encadeamento de funções de rede para construir serviços fim-a-fim sob demanda. Os participantes poderão criar, verificar e discutir o funcionamento de cada componente de uma corrente de blocos baseada no Hyperledger Fabric e aprender a controlar as atividades através de contratos inteligentes.

O capítulo é organizado em seis partes principais: i) os fundamentos da tecnologia de corrente de blocos; ii) o consenso em corrente de blocos iii) requisitos básicos de consenso e iv) consensos baseados em prova e tolerantes a falhas de paradas e bizantinas, v) a elaboração prática de uma aplicação que se serve de uma corrente de blocos programada no Hyperledger Fabric [2] e vi) as perspectivas futuras e problemas em aberto.

[2]Disponível em https://github.com/hyperledger/fabric

3.2. A Tecnologia de Corrente de Blocos[3]

Esta seção aborda a tecnologia de corrente de blocos, iniciando com o problema do gasto duplo resolvido em 2008 por Satoshi Nakamoto com a proposta da moeda virtual Bitcoin [Nakamoto 2008]. Aborda-se também propriedades e categorias de correntes de bloco, modelos de rede, assim como tipos de consenso e classes de protocolos de consenso.

3.2.1. As Moedas Digitais e o Problema do Gasto Duplo

Durante a maior parte do século XX, engenheiros, desenvolvedores, criptógrafos e profissionais de tecnologia da informação procuraram resolver o problema de transferência de ativos para permitir a criação de uma moeda digital descentralizada. A transferência de arquivos na Internet é realizada facilmente copiando-se o arquivo original e enviando-o para o destino. No entanto, a transferência de ativos (dinheiro, ações etc.) é bem mais complexa e requer um intermediário para prover confiança porque ao transferir um ativo da origem para o destino deve-se garantir a subtração do valor do ativo a ser transferido da origem e garantir que o valor do ativo a ser transferido será acrescentado no destino. A dificuldade reside no problema do gasto duplo (*double spending problem*), que ocorre quando um mesmo ativo é utilizado mais de uma vez em diferentes transações de um sistema, pois os ativos digitais podem ser facilmente replicados. Ainda que comumente postulado para o caso de moedas digitais, o problema estende-se a qualquer tipo de recurso digital único, como endereços IP, domínios, registros de identidade, propriedade intelectual, recursos computacionais, serviços, etc. A Figura 3.1 ilustra o problema do gasto duplo. No exemplo, suponha que Alice deseja transferir moedas para Bob. Se Alice e Bob utilizam dinheiro em espécie, Alice deve ceder suas moedas a Bob e não possuirá mais as moedas após a transação. Porém, se uma moeda digital for utilizada, é necessária uma maneira de garantir que Alice gastou as moedas, pois moedas digitais são representações em bits que podem ser facilmente duplicados. Neste caso, Alice pode enviar um arquivo digital com o valor desejado a Bob enquanto mantém uma cópia local do arquivo. Se Alice ainda mantiver uma cópia, ela pode enviá-la a uma terceira pessoa, Carol, realizando um gasto duplo.

Tradicionalmente, a prevenção do gasto duplo é realizada através da intermediação centralizada em uma terceira entidade com poderes de autoridade, que utiliza regras de negócio para autorizar as transações e verificar se as moedas envolvidas foram gastas. Essa abordagem é normalmente utilizada em bancos, companhias de crédito, plataformas de vendas em linha (*online*) e, no caso de ativos na Internet, através de organizações como a *Internet Assigned Numbers Authority* (IANA)[4]. Porém, a centralização implica que todos os participantes do sistema possuam total confiança na autoridade central, que pode cobrar taxas, limitar o volume de transações e controlar a rede de forma arbitrária. Além disso, uma falha na autoridade central pode interromper todas as transações do sistema por tempo indeterminado. O modelo centralizado, portanto, cria um ponto único de falha na rede, impactando tanto a disponibilidade quanto a confiança no sistema. No caso de ativos financeiros, os bancos cobram taxas exorbitantes e introduzem enormes

[3]Esta seção é baseada no projeto de fim de curso de Gabriel Antonio Fontes Rebello [Rebello 2019].
[4]Disponível em https://www.iana.org/

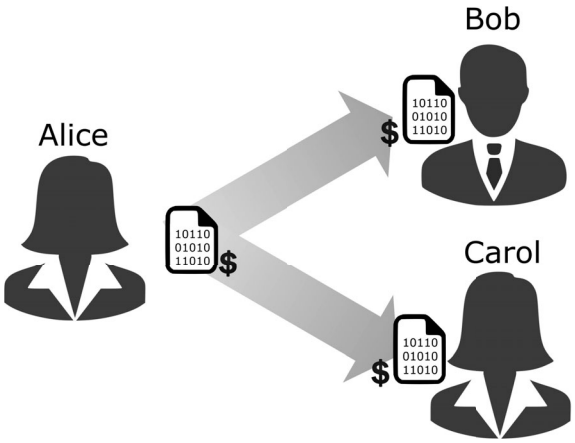


Figura 3.1: Exemplo do problema do gasto duplo no qual a mesma representação digital de uma moeda é replicada e gasta duas vezes.

atrasos para a transferência.

O conceito de moeda digital descentralizada, doravante referida como criptomoeda, é o principal propulsor de aplicações envolvendo trocas de ativos. Durante décadas, procurou-se um mecanismo para viabilizar uma criptomoeda totalmente funcional. Em 1983, David Chaumian introduziu o primeiro protocolo anônimo para criação de criptomoedas utilizando o conceito de assinatura cega (*blind signature*) [Chaum 1983]. A assinatura cega provê privacidade às transferências de moedas, mas depende fortemente de entidades centralizadas para realizar as assinaturas. Em 1998, Wei Dai propôs *b-money*, uma criptomoeda que introduz a ideia de criar valor monetário através da resolução de desafios computacionais e com consenso descentralizado [Dai 1998]. No entanto, a proposta possuía poucos detalhes sobre a implementação do consenso, dificultando sua adoção. Em 2002, Adam Back propôs formalmente *Hashcash*, um algoritmo de prova de trabalho (*Proof of Work* - PoW) baseado em funções resumo (*hash*) criptográficas[5] para prevenção de *spam* de e-mails e ataques de negação de serviço [Back 2002]. Em 2005, Hal Finney desenvolveu o conceito de provas de trabalho reutilizáveis (*Reusable Proof of Work* - RPoW) em um sistema que reúne os fundamentos do *b-money* e os desafios computacionalmente custosos do *Hashcash* [Finney 2005]. No entanto, a proposta também dependia de entidades confiáveis para ser implementada.

Satoshi Nakamoto revolucionou a área de consenso distribuído ao propor em 2008 um consenso probabilístico baseado em prova de trabalho (*Proof of Work* - PoW) [Nakamoto 2008]. O artigo "Bitcoin: A peer-to-peer electronic cash system"foi publicado em uma lista de correio eletrônicos de criptografia. No algoritmo de consenso por prova de trabalho um nó que propõe um bloco, denominado nó minerador[6], deve apresentar uma prova de que pode liderar o consenso gastando recursos computacionais para resolver independentemente um desafio matemático computacionalmente custoso[7]. O Bitcoin é a primeira criptomoeda totalmente descentralizada, combinando a prova de trabalho com

[5]Os fundamentos de funções resumo criptográficas e criptografia de chaves assimétricas são apresentados no Apêndice A.

[6]O nome minerador vem da da dificuldade e do enorme trabalho necessário para vencer o desafio.

[7]O desafio é calcular um *hash* com um certo número de zeros.

uma nova e revolucionária estrutura de dados organizada em blocos de transações: a corrente de blocos (*blockchain*). A proposta de solução do problema do gasto duplo utiliza tecnologias já conhecidas e dominadas como criptografia, consenso, redes de comunicação e incentivos.

O desafio matemático usado como prova de trabalho depende apenas do estado mais recente da corrente de blocos e, portanto, é totalmente paralelizável. A dificuldade do desafio[8] é ajustada periodicamente de acordo com a capacidade de mineração da rede para garantir uma taxa constante de mineração de blocos. Ao resolver o desafio, o minerador vencedor submete o bloco e a prova de trabalho para todos os seus vizinhos. Qualquer minerador pode tentar gerar um novo bloco a qualquer momento. O nó que vence o desafio recebe incentivos em troca do trabalho computacional realizado. O incentivo recebido em resolver o desafio deve ser maior e compensar o gasto computacional realizado para resolver o desafio. Assim, os nós maliciosos tendem a seguir a ordem instituída pelos nós honestos, pois os ganhos em agir honestamente compensam ações maliciosas [Nakamoto 2008]. A probabilidade de um minerador minerar um bloco é, portanto, proporcional à sua capacidade computacional.

O projeto Ethereum, em 2014, idealizado por Vitalik Buterin [Wood 2014, Buterin 2014], usando o consenso com prova de trabalho adota o conceito de contratos inteligentes, que permite a execução de uma máquina de Turing completa. Os contratos inteligentes expressam uma lógica de transações mais sofisticada, possibilitando a implementação de aplicações descentralizadas e autônomas, em diversos níveis e escopos. O Ethereum é considerado uma grande evolução em relação ao Bitcoin, que incorpora uma máquina de estados bem simplificada, com elementos voltados essencialmente para transações com a moeda bitcoin.

A corrente de blocos provê uma camada de confiança e, portanto, permite a transferência de ativos (valores) de uma pessoa para a outra. A Internet do Futuro está sendo chamada Internet de Valor [Truong et al. 2018], pois ativo é um valor como ações, votos, pontos de programa de fidelidade, propriedades intelectuais, músicas, descobertas científicas, entre outras. Com o sucesso da corrente de blocos começam a surgir aplicações em diversas áreas que vão revolucionar o nosso modo de fazer negócios. Na evolução de projeto da corrente de blocos *blockchain*, são então destacadas três fases [Bashir 2018]: a Blockchain 1.0 envolve o lançamento do Bitcoin em 2008, com as primeiras implementações das criptomoedas, e um ecossistema de aplicações e pagamentos com o ativo digital. A Blockchain 2.0 inicia-se com a proposta inovadora dos contratos inteligentes em 2013, e toda gama de aplicações financeiras possíveis. A Blockchain 3.0 caracteriza a adoção da tecnologia corrente de blocos no benefício de aplicações em diversas áreas, para além da financeira: governo, comércio, artes, saúde, cidades digitais, etc.

[8]O número de zeros aumenta se a capacidade de computação aumentar e o tempo para vencer o desafio diminuir. Da mesma forma, se a capacidade de processamento usada para vencer o desafio diminuir, o número de zeros diminui.

3.2.1.1. A Corrente de Blocos

A corrente de blocos, conhecida por suas aplicações em criptomoedas como o Bitcoin [Nakamoto 2008], Ethereum [Wood 2014] e outras tantas, é uma tecnologia disruptiva que deve revolucionar não somente a tecnologia da informação como também a economia e a sociedade, permitindo uma maior descentralização do mundo atual. O principal diferencial de uma corrente de blocos é ser uma estrutura de dados distribuída que garante confiabilidade sem necessitar de uma autoridade central comum a todas as entidades. Pela primeira vez, duas partes que não confiam uma na outra ou mesmo que não se conhecem podem trocar ativos sem um intermediário. A corrente de blocos substitui o modelo centralizado por um modelo colaborativo no qual a maior parte da rede deve concordar sobre todas as decisões. A corrente de blocos ao permitir decisões coletivas, eliminando a entidade centralizadora, potencialmente torna desnecessário governos, bancos, agências de crédito etc. para transferência de ativos, fornecendo poder à população. A corrente de blocos é aplicável em todo ambiente distribuído no qual os participantes não possuem confiança mútua e também não confiam ou são incapazes de entrar em acordo sobre uma autoridade centralizada que governe todos os procedimentos sensíveis da rede [Wüst and Gervais 2018]. Suas propriedades, discutidas com maior profundidade na Seção 3.2.1.3, garantem a auditabilidade, a imutabilidade e o não repúdio de todas as transações realizadas por participantes da rede, e são chave para a criação de um ambiente seguro e escalável.

A corrente de blocos é uma tecnologia de livro-razão distribuído (*Distributed Ledger Technology* - DLT) que utiliza máquinas independentes, denominadas de nós mineradores[9], para gravar, compartilhar e sincronizar transações em um sistema de controle descentralizado sobre uma rede par a par (*peer-to-peer* - P2P). Cada nó minerador possui uma cópia local da corrente de blocos na qual pode verificar qualquer transação emitida na rede desde sua criação. A adição de blocos é realizada de maneira descentralizada através de um protocolo de consenso comum aos nós mineradores. Devido à natureza descentralizada e à existência de cópias da corrente de blocos em todos os mineradores, um atacante, ou grupo de atacantes em conluio, que deseja realizar um gasto duplo precisa controlar uma parcela grande o suficiente da rede para propor um bloco que oculte uma de suas transações e seja aceito pelo protocolo de consenso [Eyal and Sirer 2018, Nakamoto 2008]. O protocolo de consenso garante que as cópias das correntes de blocos são a mesma em qualquer nó minerador e, portanto, a propriedade de não repúdio de transações é garantida. Todas as transações são assinadas por seus emissores através de criptografia de chaves assimétricas e, na maior parte das implementações, utiliza-se a chave pública como identificador do nó na rede.

A Figura 3.2 ilustra o funcionamento de uma corrente de blocos como um livro-razão distribuído através de um protocolo de consenso genérico. Um nó minerador da rede que deseja inserir um novo bloco, alterando o estado atual S da corrente de bloco, inicia uma rodada de consenso reunindo as transações válidas recebidas de usuários em um novo bloco a ser proposto. A seguir, o bloco proposto é validado localmente de acordo com políticas pré-definidas e é difundido na rede. Dependendo do protocolo de consenso

[9]Os mineradores usualmente recebem uma recompensa para executarem a mineração e, em modelos de consenso sem recompensa, são chamados de participantes do consenso ou simplesmente de nós.

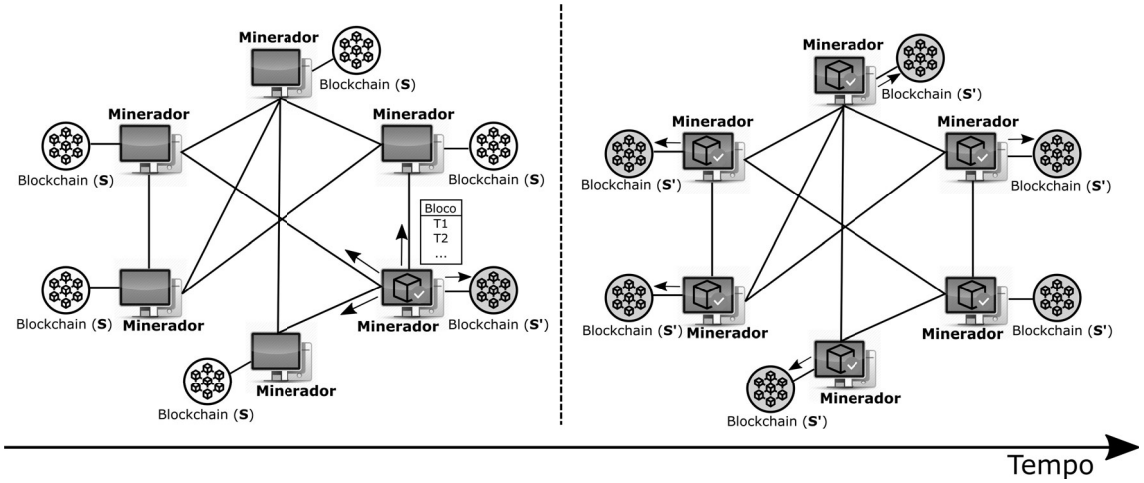


Figura 3.2: Atualização de uma corrente de blocos (*blockchain*) através de um protocolo de consenso genérico. Um nó minerador propõe um novo bloco em difusão e os demais nós mineradores verificam e adicionam independentemente o bloco proposto à corrente de blocos, incrementando o estado global S de maneira consistente.

adotado, o bloco proposto pode ser adicionado imediatamente, de forma assíncrona, à corrente de blocos do nó minerador que propôs o bloco, ou de forma síncrona em todos os nós mineradores ao fim da rodada. A figura mostra o caso de um protocolo no qual o estado S é alterado para S' no nó minerador que propôs o bloco antes da difusão. Ao

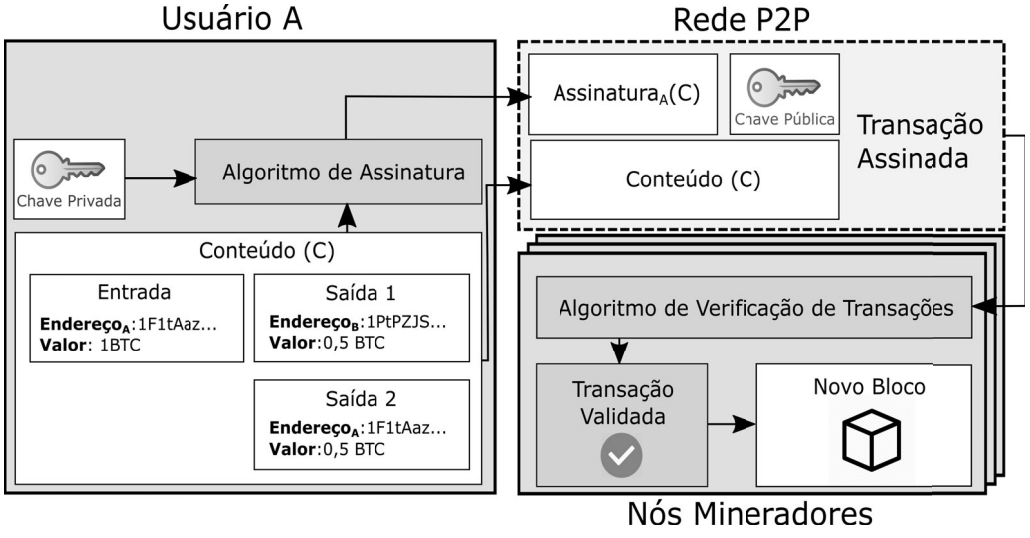


Figura 3.3: Etapas de uma transação em uma corrente de blocos. Para transferir 0,5 BTC ao usuário B, o usuário A cria uma transação contendo: i) uma entrada com seu endereço e o total de BTC que possui; ii) uma saída com o endereço de Bob e o valor que deseja transferir e iii) uma saída com seu próprio endereço e o valor que irá possuir após a transferência do valor desejado. A seguir, o usuário A utiliza um algoritmo criptográfico para assinar a transação e a envia com sua chave pública em difusão para a rede par a par (*peer-to-peer* - P2P) na qual estão os nós mineradores.

receber o bloco proposto, cada nó minerador valida o bloco independentemente e, caso aprove o bloco, o adiciona à corrente de blocos e chega ao estado S'. O algoritmo de validação de um bloco e de cada transação deve ser acordado previamente por todos os nós mineradores. Todo protocolo de consenso possui um mecanismo de atualização para obter o estado mais recente e corrigir discrepâncias de estado resultantes de blocos não aprovados, garantindo a consistência da corrente de blocos em toda a rede. Os protocolos de consenso e suas premissas são melhor discutidos na Seção 3.3.

A Figura 3.3 mostra as etapas de uma transação em um sistema de troca de ativos digitais através de uma transação simplificada de Bitcoin (BTC) [Nakamoto 2008]. Para enviar ou receber ativos, um usuário deve utilizar um par de chaves assimétricas para assinar transações. Após a difusão da transação assinada, qualquer participante do consenso pode aplicar o algoritmo de validação da rede para verificar criptograficamente a autenticidade da transação e se a transação conflita com outra já registrada na corrente de blocos. Transações inválidas são descartadas imediatamente. Caso a validação ocorra com sucesso, o participante do consenso adiciona a transação a um novo bloco que é proposto na próxima rodada do protocolo de consenso. A maior parte dos sistemas de troca de ativos provê programas que gerenciam chaves, armazenam endereços e emitem transações assinadas de forma transparente ao usuário. Estes programas são amplamente conhecidos como "carteiras".

3.2.1.2. As Estruturas de Dados da Corrente de Blocos

A estrutura de dados da corrente de blocos proposta por Nakamoto como parte da criptomoeda Bitcoin é uma lista encadeada de estruturas de dados menores, conhecidas como blocos. Cada bloco está conectado a seu antecessor através de uma função resumo (*hash*) criptográfica, criando uma corrente de blocos conforme a mostra a Figura 3.4. A corrente de blocos funciona como um livro-razão (*ledger*), ou registro permanente (*log*), de blocos ordenados no tempo. Cada bloco na corrente possui um cabeçalho contendo o valor resultante de uma função resumo (*hash*) criptográfica do bloco antecessor, e uma seção de conteúdo que armazena dados relevantes a uma aplicação. A única exceção a esta regra é o bloco inicial, o gênesis, que por definição não possui bloco anterior. A utilização de uma sequência de funções resumo criptográficas, cujas saídas diferem drasticamente após a alteração de qualquer bit na entrada[10], implica que todos os blocos incluídos a partir de uma possível alteração devem ser reconstruídos. Assim, a adição de blocos torna cada vez mais custoso alterar a corrente e, como a corrente de blocos é replicada em cada nó minerador da rede, um atacante deve invadir todos os mineradores e recriar cada corrente de blocos para modificar uma transação passada. A replicação da corrente de blocos em todos os nós mineradores e o encadeamento através de funções *hash* criptográficas garante, portanto, a imutabilidade de um bloco com muitos sucessores.

As principais características da corrente de blocos são descritas em [Greve et al. 2018, Mello et al. 2017, Chicarino et al. 2017, Mattos et al. 2018]. A literatura de corrente de blocos apresenta propostas específicas que utilizam a seção de conteúdo do bloco

[10]Os fundamentos de funções resumo criptográficas e criptografia de chaves assimétricas são apresentados no Apêndice A.

para diferentes propósitos. Nakamoto propõe originalmente registrar transações bancárias para criar uma criptomoeda [Nakamoto 2008]. Wood *et al.*, Frantz *et al.* e Androulaki *et al.* propõem utilizar correntes de blocos para armazenar e executar contratos inteligentes através de uma máquina virtual distribuída [Wood 2014, Frantz and Nowostawski 2016, Androulaki et al. 2018]. Wilkinson *et al.* propõem um sistema baseado em correntes de blocos para prover armazenamento descentralizado em nuvem com privacidade [Wilkinson et al. 2014]. Ali *et al.* propõem o uso de correntes de blocos para registrar nomes de domínio [Ali et al. 2016]. Azaria *et al.* propõem registrar informações de fichas médicas [Azaria et al. 2016]. Lee *et al.* propõem registrar votos para criar um sistema de votação descentralizado [Lee et al. 2016]. Christidis e Hun *et al.* propõem rastrear dispositivos de Internet das Coisas (*Internet of Things* - IoT) através do armazenamento de informações na corrente de blocos [Christidis and Devetsikiotis 2016, Huh et al. 2017]. Bozic *et al.* e Alvarenga *et al.* propõem registrar informações de máquinas virtuais e funções virtualizadas de rede na corrente de blocos [Bozic et al. 2017, Alvarenga et al. 2018].

O principal elemento de um sistema baseado em corrente de blocos são as transações. Uma transação representa uma ação atômica a ser armazenada na corrente de blocos que transfere um ativo entre duas entidades. A transação possui quatro componentes principais: i) um remetente, que possui um ativo que deseja transferir e que emite a transação; ii) um destinatário que receberá o ativo que se deseja transferir; iii) o ativo que será transferido e iv) uma assinatura do remetente da função resumo de toda a transação, que corresponde a uma função *hash* de todos os campos anteriores pelo remetente. Quando a imutabilidade da corrente de blocos é utilizada com transações assinadas, cria-se um sistema que funciona como um livro-razão distribuído. As transações no sistema são ordenadas dentro de cada bloco e podem ser verificadas por qualquer nó participante da rede, desde o bloco inicial da corrente de blocos, denominado bloco gênesis. Pela propriedade de chaves criptográficas assimétricas, a assinatura do *hash* da transação, usando a chave privada do remetente provê autenticidade, não repúdio e integridade à transação, fornecendo maior segurança à rede. Ainda, é possível obter pseudo-anonimidade utilizando chaves públicas ou endereços únicos, que não revelam as identidades pessoais, para identificar remetentes e destinatários. A privacidade de transações, desejada em casos onde apenas o emissor e destinatário devem poder consultar a transação [Androulaki et al. 2018], pode ser garantida criptografando a transação com a chave pública do destinatário.

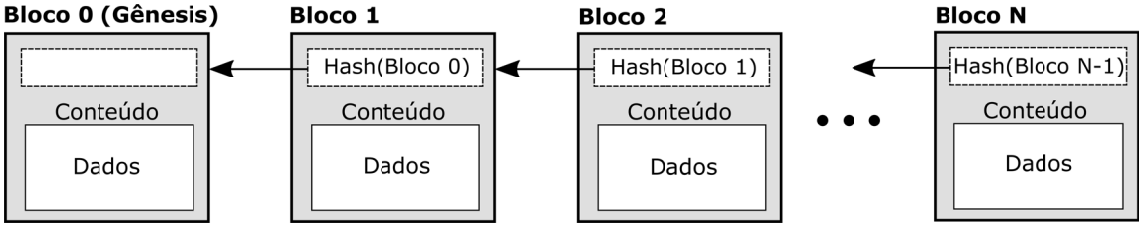


Figura 3.4: Estrutura de uma corrente de blocos na qual cada bloco é associado ao bloco seguinte e a integridade das transações de um bloco é garantida por uma uma função resumo (*hash*) criptográfica.

3.2.1.3. Propriedades de Corrente de Blocos

A corrente de blocos possui propriedades que proveem benefícios às aplicações e sistemas baseados nesta tecnologia. As principais propriedades da corrente de blocos são [Zheng et al. 2018, Xu et al. 2017, Wüst and Gervais 2018]:

- **Descentralização**. As correntes de blocos executam de maneira distribuída, através do estabelecimento de consenso entre todos os participantes da rede. Não há uma entidade centralizadora;
- **Desintermediação**. A corrente de blocos elimina a necessidade de um intermediário confiável para a troca de ativos. Os ativos podem ser trocados diretamente pela rede e a confiança é estabelecida através de consenso;
- **Imutabilidade**. Os dados armazenados em uma corrente de blocos são imutáveis. Não é possível modificar ou recriar qualquer dado incluído na corrente de blocos de forma retroativa. Toda atualização na corrente de blocos é realizada de forma incremental;
- **Irrefutabilidade**. Os dados são armazenados na corrente de blocos em forma de transações assinadas, que não podem ser alteradas devido à propriedade de imutabilidade da corrente de blocos. Portanto, o emissor de uma transação jamais pode negar sua existência;
- **Transparência**. Todos os dados armazenados na corrente de bloco são acessíveis por todos os participante da rede. Em correntes de blocos públicas, como o Bitcoin e o Ethereum, as transações são abertas para qualquer usuário com acesso à Internet;
- **Auditabilidade**. Como consequência da transparência, todo participante pode verificar, auditar e rastrear os dados inseridos na corrente de blocos para encontrar possíveis erros ou comportamentos maliciosos. No caso de correntes de blocos federadas, o processo de auditagem pode responsabilizar um malfeitor na rede;
- **Disponibilidade**. As correntes de blocos são estruturas replicadas em cada participante da rede e, portanto, a disponibilidade do sistema é garantida mesmo sob falhas, devido à redundância de informações;
- **Anonimidade**. Os usuários e nós mineradores de uma corrente de blocos são identificados por chaves públicas ou identificadores únicos que preservam suas identidades. Ainda, é possível utilizar uma chave pública em cada transação, evitando a rastreabilidade do usuário e conferindo um grau a mais de anonimidade.

Além das propriedades citadas, a corrente de blocos também pode prover privacidade, ao custo da transparência e auditabilidade. Em algumas implementações de corrente de blocos, os dados inseridos na corrente de blocos são criptografados com uma chave pública, evitando que todos os participantes possam ver o conteúdo da transação [Smolensk 2018, Androulaki et al. 2018]. Nesses casos, a transparência e a auditabilidade limitam-se a um ou mais participantes que detêm a chave privada correspondente e podem descriptografar a transação criptografada.

3.3. O Consenso em Correntes de Blocos

Um protocolo de consenso é o principal componente utilizado para manter a consistência de uma corrente de blocos. No modelo de corrente de blocos públicas usando prova de trabalho (*Proof-of-Work* – PoW), qualquer nó pode participar do protocolo de consenso e procurar acrescentar blocos, estendendo a corrente de blocos. Para isso, geralmente é necessário resolver um enigma (desafio) criptográfico, que é chamado de mineração pela dificuldade computacional do processo. O procedimento de mineração é usado para provocar um custo, que na prova de trabalho é um custo computacional que tem o grande inconveniente de consumir muita energia. Portanto, estes protocolos escalam no número de participantes na rede, mas apresentam um desempenho muito baixo em número de transações por segundo, pois múltiplas propostas de inserção de novos blocos podem ser feitas concorrentemente e gerar bifurcações (*forks*) na corrente, sendo necessário aplicar uma regra onde o ramo da bifurcação mais longo ganha. A possibilidade de bifurcações e regras para decidir qual dos ramos deve continuar são feitas de forma probabilística, resultando em um consenso probabilístico.

Diferentemente do modelo de consenso probabilístico, existe o modelo consenso determinístico que é tolerantes a falhas de parada (*crash*) ou bizantinas. No modelo de corrente de blocos privadas com tolerância a falhas, apenas nós autorizados, denominados validadores, executam um protocolo de consenso para acrescentar blocos e estender a corrente de blocos, que apresenta a mesma evolução em todos os participantes do consenso, pois não ocorrem bifurcações (*forks*). Apesar de apresentar um bom desempenho, o modelo de consenso tolerante a falhas não escala no número de participantes na rede[11].

Nota-se que a convergência e funcionamento correto de um sistema baseado em corrente de blocos depende do consenso entre todos os nós participantes, que devem adotar um protocolo comum para tal. Esta seção apresenta a fundamentação teórica necessária para a compreensão de protocolos de consenso em sistemas distribuídos, que é o principal desafio de implementação de correntes de blocos. A escolha do protocolo de consenso deve levar em consideração características arquiteturais da rede de comunicação que interconecta os participantes, requisitos de desempenho como a vazão em transações por segundo, a escalabilidade em número de nós, a tolerância a possíveis falhas e ataques, dentre outros. Esta seção procura fazer uma análise destas características.

3.3.1. O Sistema de Comunicação da Corrente de Blocos

Um sistema distribuído envolve um conjunto de diferentes processos[12], por exemplo computadores participantes, que trocam mensagens uns com os outros e se coordenam para obter um objetivo comum. Portanto, todo sistema distribuído se baseia em um sistema de comunicação que interconeta os participantes e que é usado pelos protocolos de consenso. Os protocolos de consenso assumem diversas hipóteses sobre o sistema de comunicação. Praticamente todos os protocolos de consenso assumem um sistema

[11]Deve ser ressaltado que o modelo de consenso por prova de trabalho (PoW) escala, mas não apresenta uma alta vazão de transações. Por outro lado, o consenso tolerante a falhas apresenta alta vazão, mas não escala, existem propostas de modelos híbridos, usando o PoW para definir comitês de participantes que executam o consenso tolerante a falhas bizantinas tradicional.

[12]Este capítulo usa os termos nós (*node*), par (*peer*), participante do consenso, computador ou componente como sinônimos de processo (*process*).

de comunicação par-a-par (*peer-to-peer*) completamente conectado, em que cada par de participantes consegue se comunicar diretamente através de canais bidirecionais. Além disso, várias outras suposições são realizadas para definir o nível de sincronia fornecido pelo sistema de comunicação, o tipo de falha que o sistema consegue suportar, e a disponibilidade de primitivas criptográficas.

Modelos de Sincronia do Sistema de Comunicação - Os modelos de sincronia definem os aspectos relacionados com o tempo no sistema relativos ao processamento nos participantes e para entrega correta de mensagem pelo sistema de comunicação. Existem vários modelos de sincronia [Attiya and Welch 2004, Lynch 1996], a seguir são descritos os três mais importantes e práticos:

- Assíncrono [Fischer et al. 1985, Lynch 1996] - Este é o modelo mais fraco de sincronia para um sistema distribuído. Neste modelo, não existe um limite para o tempo de processamento local em um participante (processo) e para a entrega/comunicação de mensagens. Consequentemente, neste modelo não existe a noção de tempo e seria o modelo ideal de ser considerado em sistemas de comunicação. No entanto, a impossibilidade de FLP[13] (*FLP Impossibility*) afirma que não possível se atingir consenso em sistemas de comunicação assíncronos quando pelo menos um participante pode sofrer falha de parada;

- Síncrono [Lynch 1996] - Este é o modelo mais forte de sincronia, no qual existe um limite de tempo conhecido para os processamentos locais nos participantes e também para a comunicação/entrega das mensagens. Este modelo de sistema de comunicação é praticamente irreal de se obter;

- Parcialmente ou Eventualmente Síncrono [Dwork et al. 1988]. A ideia por trás destes modelos é de que o sistema trabalha de forma assíncrona (não respeitando nenhum limite de tempo) a maior parte do tempo. Porém, durante períodos de estabilidade, o tempo para a entrega entrega (*delivery*) de mensagens é limitado. Deve ser ressaltado que este modelo engloba o comportamento de redes de melhor esforço, como a Internet, que passam por tempos de estabilidade e tempos de perturbação. Este é modelo de sincronia mais fraco no qual o consenso possui solução determinística [Fischer et al. 1985]. Este é um modelo intermediário mais realístico, que geralmente é considerado na implementação de aplicações distribuídas tais como protocolos de consenso.

Modelos de Falhas do Participante do Consenso - Normalmente dois tipos de falhas são consideradas:

- Falhas por Parada ou Desastrosas (*crash*) - Neste modelo, um nó participante em falha não responde e não executa novas operações durante a execução do sistema.

[13]FLP vem de F-Fisher, L-Linch e P-Paterson.

- Falhas Bizantinas - Neste modelo, um nó participante em falha pode exibir um comportamento arbitrário, desviando do protocolo especificado e executando qualquer ação. A falha bizantina é muito mais complexa de se tolerar que a falha desastrosa, uma vez que um participante pode ser um agente malicioso que responde uma coisa para um participante e responder o contrário para outro participante.

Modelo de Mensagens com Autenticação Criptográfica – No modelo de falhas bizantinas, os protocolos geralmente necessitam de mensagens autenticadas, que podem ser implementados através de criptografia assimétrica para gerar assinatura digitais para as mensagens, que requer um sistema de infraestrutura de chaves públicas, ou através de criptografia simétrica e uso de código de autenticação de mensagens (*Message Authentication Codes* –MAC), que requer o compartilhamento de segredos [Bishop 2002].

3.3.2. Consenso em Corrente de blocos

Consenso é o processo pelo qual a partir de um grupo de participantes independentes, todos os participantes corretos atingem a mesma decisão coletiva de aceitar o novo bloco a ser inserido na corrente de blocos. A proposta de um novo bloco foi feita previamente por um dos participantes do consenso. Formalmente, este problema é definido em termos de duas primitivas [Hadzilacos and Toueg 1994]:

- *propose(P,b)*: o novo bloco, *b*, é proposto ao conjunto de participantes do consenso *P*;
- *decide(b)*: executado pelo protocolo de consenso para notificar ao(s) interessado(s), geralmente alguma aplicação, que *b* é o novo bloco a ser acrescido na corrente de blocos.

Para essas primitivas satisfazerem a correção (*safety*) e a vivacidade (*liveness*), elas devem verificar as seguintes propriedades [Aspnes 2003, Attiya and Welch 2004]:

- **Acordo**: Se um processo correto decide *v*, então todos os processos corretos terminam por decidir *v*.
- **Validade**: Um processo correto decide *v* somente se *v* foi previamente proposto por algum processo.
- **Terminação**: Todos os processos corretos terminam por decidir.

A propriedade de acordo garante que todos os processos corretos decidem o mesmo valor. A validade relaciona o valor decidido com os valores propostos e sua alteração dá origem a outros tipos de consensos. As propriedades de acordo e validade definem os requisitos de correção (*safety*) do consenso, já a propriedade de terminação define o requisito de vivacidade (*liveness*) deste problema.

Fisher, Lynch e Paterson (FLP) publicaram em 1985 um dos trabalhos mais importantes envolvendo o problema do consenso [Fischer et al. 1985], conhecido como a

impossibilidade de FLP. Eles provam que não é possível obter-se consenso em um sistema distribuído completamente assíncrono no qual pelo menos um processo pode falhar (qualquer tipo de falha). Em vista disso, a maioria das propostas de consenso encontradas na literatura consideram algum comportamento temporal do sistema, no qual componentes (processos e/ou canais) obedecem a requisitos temporais. Estes requisitos geralmente estão relacionados com temporizadores (*timeouts*) encapsulados em detectores de falhas [Chandra and Toueg 1996] ou são atendidos a partir da ocorrência de possíveis comportamentos síncronos do sistema subjacente.

3.3.3. O Teorema CAP

Os principais desafios de projeto de corrente de blocos com alta disponibilidade são a tolerância a falhas e a coordenação entre os nós mineradores. O teorema CAP (*Consistency*, *Availability*, *Partition*) prova que todo sistema distribuído apresenta um compromisso entre três propriedades [Brewer 2000]:

- **Consistência (*consistency*)**: todos os nós do sistema distribuído possuem os dados mais atuais da rede;
- **Disponibilidade (*availability*)**: o sistema está operante, acessível e é capaz de responder corretamente a novas requisições;
- **Tolerância à partição (*partition tolerance*)**: o sistema distribuído opera corretamente mesmo que um grupo de nós falhe. Se for possível haver comportamento malicioso dos nós, o sistema continua funcionando corretamente com alguns nós honestos, mesmo na presença de um grupo de nós maliciosos.

O teorema prova que é impossível para um sistema distribuído atingir plenamente todas as três propriedades [Brewer 2000]: consistência, disponibilidade e telerância a partição. Por isto, na prática, sistemas distribuídos privilegiam uma ou duas propriedades, sacrificando as demais. A replicação contribui para a tolerância a falhas. A consistência é obtida com o uso de algoritmos de consenso que garantem que todos os nós mineradores possuem os mesmos dados. No Bitcoin e em diversas outras aplicações [Nakamoto 2008, Wood 2014, Ali et al. 2016], a consistência é sacrificada em nome da disponibilidade e da tolerância a partições. Isto significa que a consistência não é obtida simultaneamente com as outras duas propriedades, mas sim gradativamente, com o tempo, à medida que os blocos são validados pelos nós da rede. Correntes de blocos baseadas em protocolos tolerantes a falhas bizantinas privilegiam fortemente a consistência, em detrimento da disponibilidade [Schwartz et al. 2014, Androulaki et al. 2018, Buchman 2016, Miller et al. 2016, Sousa et al. 2018].

3.3.4. Consenso Probabilístico: Protocolos Baseados em Provas

Apesar da inovação, a prova de trabalho do Bitcoin e do Ethereum apresentam limitações evidentes de desempenho. O Bitcoin possui uma latência de consenso de aproximadamente dez minutos, e uma vazão de transações, de apenas 7 transações por segundo. O Ethereum possui uma latência de 15 segundos e uma vazão de 15 transações por segundo. Este baixo desempenho consiste em um desafio para atender às necessidades dos sistemas atuais [Popov 2017, Eyal et al. 2016].

Outra importante limitação do consenso por prova de trabalho é o excessivo gasto energético de 57,7 TWh por ano[14]. Para se ter uma ideia, esse valor é três vezes maior que a energia gerada pelas usinas nucleares de Angra. O gasto energético do consenso por prova de trabalho é ecologicamente inaceitável. Outros algoritmos baseados em prova procuram mitigar o excesso de gasto energético e as limitações de desempenho presentes na prova de trabalho [Vasin 2014, Schwartz et al. 2014, Kiayias et al. 2017]. Uma breve explicação dos algoritmos mais conhecidos é mostrada a seguir:

- Prova de Posse[15] (*Proof of Stake* - PoS) - Em vez de gastar recursos computacionais, um nó minerador deve apostar parte de seus ativos para receber uma chance de minerar um bloco. Sua chance é proporcional à quantia de ativos apostados. Nos últimos anos, a prova de participação têm se destacado devido ao desejo do Ethereum de abandonar a prova de trabalho para implementar PoS [Tikhomirov 2018];

- Prova de Posse Delegada (*Delegated Proof of Stake* - DPoS) - Os nós mineradores utilizam seus ativos para eleger delegados em um quórum que define o bloco a ser adicionado. A quantidade de votos de um minerador é proporcional aos seus ativos [Kiayias et al. 2017];

- Prova de Queima ou Destruição (*Proof of Burn* - PoB) - Como o PoS, porém os ativos são imediatamente destruídos [Paul et al. 2014];

- Prova de Autoridade (*Proof of Authority* - PoA) - Similar ao DPoS, porém o conjunto de delegados (autoridades) é pré-determinado em acordo e suas identidades são públicas e verificáveis por qualquer participante da rede [Angelis et al. 2018];

- Prova de Capacidade de Armazenamento (*Proof of Capacity* - PoC) - A probabilidade de propor um bloco é proporcional ao espaço de armazenamento cedido à rede por um nó minerador. Quanto maior a capacidade de armazenamento em disco, maior o domínio sobre o consenso [Zheng et al. 2018];

- Prova de Tempo Decorrido (*Proof of Elapsed Time* - PoET) - Cada nó minerador recebe um temporizador aleatório decrescente e o nó cujo temporizador terminar primeiro propõe o próximo bloco [Olson et al. 2018]. Este protocolo de consenso funciona exclusivamente em *hardware* que suportam a tecnologia *Intel software guard extensions* (SGX). O Intel SGX garante a distribuição aleatória de temporizadores e que nenhuma entidade tem acesso a mais de um nó minerador.

Um exemplo de protocolo de prova de posse é o Ouroboros que é detalhado a seguir.

O Protocolo Ouroboros [Kiayias et al. 2017] baseado em prova de posse determina um comitê com partes interessadas (*stakeholders*) aleatoriamente selecionadas para a eleição de um líder, que propõe o próximo bloco da corrente. A probabilidade de um

[14]https://digiconomist.net/bitcoin-energy-consumptio

[15]*Proof of Stake* é também traduzido por prova de participação ou prova de montante.

nó ser selecionado como líder ou de participar do comitê para eleição de líder é diretamente proporcional à participação do nó na rede. O Ouroboros é utilizado pela plataforma Cardano[16], responsável por executar a corrente de blocos da criptomoeda ADA.

O protocolo determina dois tipos de intervalo de tempo: períodos (*slots*) e épocas (*epochs*). A Figura 3.5 ilustra a divisão do tempo no protocolo Ouroboros. Os períodos são intervalos de tempo curtos dentro de uma época em que, a cada período, um líder previamente eleito propõe o bloco a ser adicionado à corrente. As épocas são intervalos de tempo em que um comitê formado por partes interessadas aleatoriamente selecionadas executam uma rodada de consenso. A saída da execução define os participantes do comitê da próxima época e os líderes de cada período da próxima época.

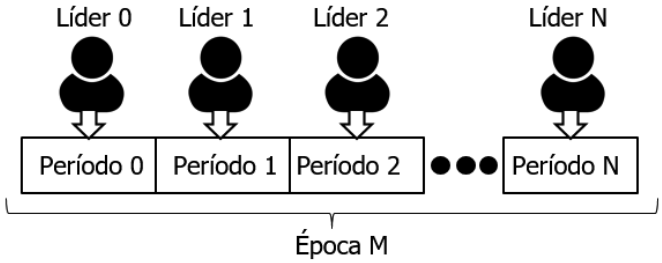


Figura 3.5: Divisão do tempo no protocolo Ouroboros.

O procedimento para a eleição do líder é feito através de uma computação segura multipartidária (*Secure Multi-party Computation* - MPC). A computação segura multipartidária permite que múltiplos participantes com entradas individuais obtenham a mesma saída como resultado [Rabin and Ben-Or 1989]. O processo de eleição é separada em três fases: efetivação (*commitment*), revelação (*reveal*) e recuperação (*recovery*). A Figura 3.6 ilustra as etapas para eleição do líder e escolha dos participantes do consenso no protocolo Ouroboros. Na fase de efetivação, cada participante do comitê gera um valor aleatório e difunde uma mensagem de efetivação. A mensagem de efetivação contém o valor gerado criptografado. Na fase de revelação, cada participante difunde uma mensagem de abertura, revelando o segredo enviado anteriormente. Na fase de recuperação, os participantes formam uma semente (*seed*) com os segredos revelados e usam a semente no algoritmo Siga-o-Satoshi (*Follow-the-Satoshi* - FTS) que seleciona uma moeda da rede para escolher o líder e os participantes do comitê da próxima época.

Durante a execução da fase de revelação, um membro do comitê pode decidir por abandonar o protocolo e não revelar o segredo ao observar os valores recebidos. O Ouroboros usa o esquema de compartilhamento de segredo verificável (*Verifiable Secret Sharing* - VSS), que divide o valor gerado em parcelas. As parcelas são criptografadas e enviadas aos membros do comitê na fase de efetivação, de maneira que os participantes do consenso acumulem mensagens de efetivação dos outros participantes. O esquema permite a recuperação dos segredos não-revelados durante a fase de recuperação, em que os membros honestos postam as parcelas recebidas dos membros que abandonaram o protocolo. Este procedimento permite a reconstrução do segredo e a conclusão do protocolo mesmo que alguns participantes do consenso sejam adversários.

[16]https://www.cardano.org/en/home/

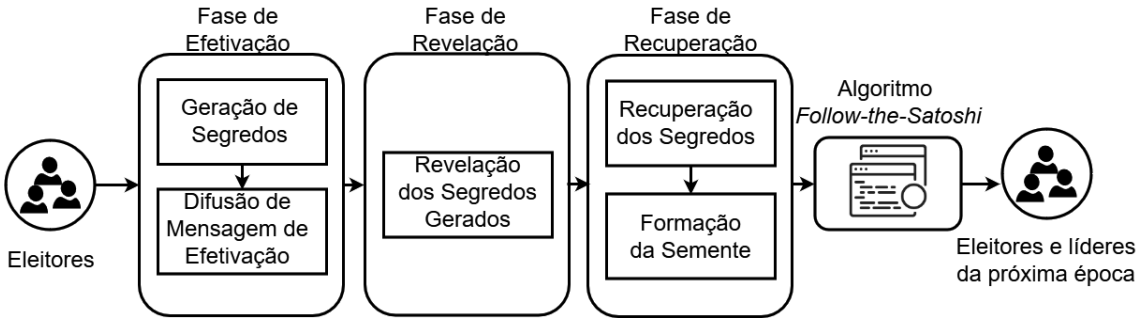


Figura 3.6: Visão geral das fases para formação do comitê responsável por eleger os líderes de períodos no protocolo Ouroboros

3.3.5. Consenso Determinístico: Protocolos Tolerantes à Falhas

Em redes bem-definidas, como redes permissionadas síncronas ou eventualmente síncronas, é possível obter consenso determinístico. Nesse tipo de consenso, todos os nós mineradores devem votar pela aprovação ou rejeição de um bloco e atingir um consenso antes de adicionar o novo bloco à corrente. Como consequência, todos os nós mineradores[17] devem ser conhecidos e identificados. O protocolo de consenso garante que a adição de um bloco à corrente ocorre de forma sincronizada para todas as réplicas, i.e., as réplicas adicionam a mesma sequência de blocos na corrente. Portanto, a consistência do sistema é garantida em qualquer momento e não existem bifurcações (*forks*) na corrente de blocos.

Os protocolos tolerantes a falhas são divididos em tolerantes a falhas de paradas (*Crash Tolerant Fault* - CFT) e tolerantes a falhas bizantinas. A classe de protocolos de consenso para as correntes de blocos tolerantes a falhas bizantinas (*Byzantine Fault Tolerant* - BFT) é particularmente interessante, pois que garantem a consistência da corrente de blocos sob comportamento malicioso. A ideia principal dos protocolos BFT é eleger um líder que inicia e comanda uma rodada de consenso distribuindo o novo bloco proposto a todos os participantes do consenso. Protocolos tolerantes a falhas bizantinas representam um meio termo entre um ambiente totalmente sem confiança e um ambiente totalmente confiável. Os protocolos BFT promovem uma camada de confiança a mais em comparação a protocolos tolerantes apenas a falhas por parada pois toleram comportamento malicioso na rede. No entanto, geralmente necessitam de mais réplicas, quando comparado com os protocolos que toleram falhas apenas por parada, para tolerar a mesma quantidade de falhas.

Existem muitos protocolos de consenso propostos na literatura, a seguir são apresentados um protocolo tolerante a falha de parada e quatro protocolos que foram projetados para tolerar falhas bizantinas (BFT) em um sistema parcialmente síncrono: PBFT [Castro and Liskov 2002] – o primeiro protocolo BFT prático, que utiliza vetores de MAC ao invés de assinaturas digitais para conseguir um desempenho similar aos protocolos que toleram apenas falhas por parada; MinBFT [Veronese et al. 2013] – que utiliza componentes seguros para diminuir tanto a quantidade de réplicas quanto os pas-

[17]É comum usar o termo nós validadores em vez de mineradores uma vez que eles votam no consenso sem executar um desafio ou sem apostar.

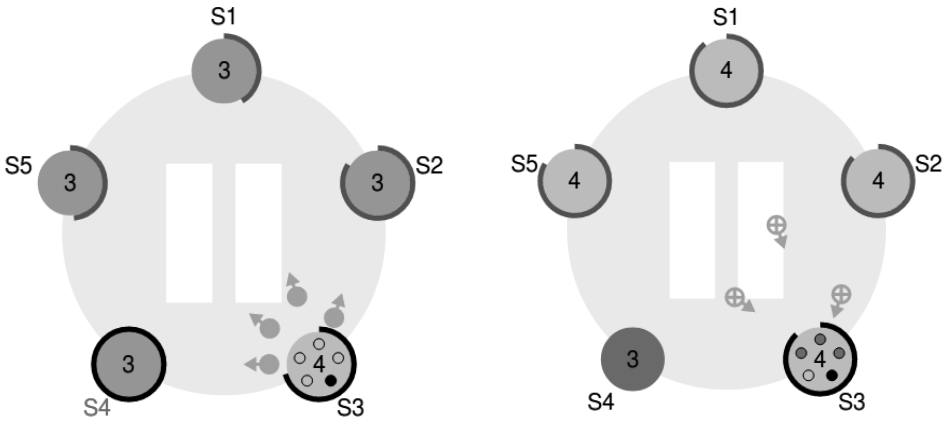
sos de comunicação necessários para finalizar o protocolo; Zyzzyva [Kotla et al. 2009] – que utiliza especulação com o objetivo de melhorar o desempenho; e BFT-CUP [Alchieri et al. 2018] – que foi projetado para ambientes em que os processos são, a priori, desconhecidos.

O Protocolo Raft é um protocolo de consenso tolerante a falhas de parada (*crash*) proposto por Ongaro em 2014 [Ongaro and Ousterhout 2014]. Desenvolvido como uma alternativa de fácil compreensão ao Paxos [Lamport 2001], o Raft possui eficiência e resultados equivalentes ao Paxos, além de providenciar uma fundamentação mais adequada para construção de sistemas práticos. A simplicidade na compreensão é consequência da divisão do consenso em 3 fases: a eleição do líder que ocorre para definir o primeiro líder ou caso o atual falhe; a replicação do registo nas máquinas de estado de todos os nós; e a segurança do registro, garantido que nenhum registro é apagado ou duplicado. O protocolo é baseado nas fortes premissas de que todos os nós são conhecidos, não existe comportamento bizantino na rede e os nós operam em um ambiente assíncrono com relógios defeituosos [Howard 2014]. O Raft utiliza a replicação de registros utilizando a replicação de máquina de estados (RME) [Schneider 1990] que, em conjunto com o protocolo de consenso, garante a existência dos mesmos comandos na mesma ordem em todos as máquinas de estados dos nós. Dessa forma, o Raft garante a integridade dos registros e o funcionamento do consenso com até f participantes em estado de falha de parada dos $n = 2f + 1$ participantes da rede.

O Raft divide o tempo em mandatos de duração indefinida e numerados com inteiros consecutivos. A qualquer momento, cada nó está em um dos três estados: líder, seguidor ou candidato. O líder aceita as entradas do cliente, replica as entradas nos outros nós e informa aos nós quando é seguro aplicar as entradas às máquinas de estado. O seguidor apenas responde a pedidos do líder e do candidato. O candidato é utilizado para eleger um novo líder caso o atual falhe, assim começando um novo mandato. A falha do líder é detectada utilizando o mecanismo de pulsação periódica (*heartbeat*) através de chamadas remotas de procedimento (*Remote Procedure Call* - RPC).

Para detectar falhas, cada nó possui um temporizador de duração escolhida aleatoriamente de um intervalo de 150 a 300 ms para marcar o tempo limite de eleição. Caso o tempo limite de eleição do seguidor seja atingido sem que haja sinal de comunicação do líder pelo mecanismo de pulsação, o seguidor assume que não existe um líder ou que líder está em estado de falha e começa a eleição de um novo líder ao incrementar o mandato atual e se tornar um candidato. Em seguida, o candidato vota em si mesmo e envia pedidos de voto para o restante dos nós por RPC. A fase de eleição está representada na Figura 3.7. O tempo limite de eleição de cada seguidor é redefinido no início de toda eleição. O candidato permanece nesse estado até que consiga $f + 1$ votos, outro candidato se estabeleça como líder ou o tempo limite de eleição seja atingido. A escolha aleatória do tempo limite de eleição garante que a existência de múltiplos candidatos é rara.

Uma vez que o líder é eleito, ele começa a atender as solicitações dos clientes. Cada solicitação de cliente consiste em um comando para a máquina de estados replicada executar. Essas solicitações são guardadas em forma de registros contendo o comando a ser executado, o número do mandato em que o líder recebeu a solicitação e um identificador que define a sua ordem no livro-razão. O líder registra a solicitação no seu



(a) Início da eleição do líder. (b) Término da eleição do líder.

Figura 3.7: A Figura 3.7(a) representa o início da eleição de um novo líder, onde o tempo limite de eleição do seguidor S3 esgotou-se. O seguidor S3 vira um candidato, incrementa o número de mandato de 3 para 4, representado pela cor rosa e verde respectivamente, e envia pedidos de voto aos demais seguidores. Logo em seguida, os seguidores respondem o pedido de voto e incrementam seus números de mandato, assim elegendo o candidato S3 como o novo líder, conforme representado na Figura 3.7(b). Como o antigo líder S4 não respondeu as chamadas do novo líder S3, detectou-se uma falha em S4, representada pela cor cinza.

livro-razão e informa, por RPC em paralelo, os seguidores para replicar o novo registro em seus livros-razão. Após receber $f+1$ respostas positivas dos seguidores, o líder aplica o comando do cliente em sua máquina de estados, envia uma ordem por RPC em paralelo para todos os seguidores aplicarem a entrada em suas máquinas de estados e informarem a saída de suas máquinas de estados para o líder. A partir desse ponto, a solicitação do cliente está efetivada.

O Protocolo Prático de Consenso Tolerante a Falhas Bizantinas - *Practical Byzantine Fault Tolerance* (PBFT) [Castro and Liskov 2002] foi o primeiro protocolo de consenso BFT que considera aspectos práticos, alcançando um desempenho semelhante aos protocolos para falhas por parada e resolve o problema dos generais bizantinos [Lamport et al. 1982]. Neste protocolo, a autenticidade das mensagens é garantida através de vetores de MACs (*Message Authentication Codes*) ao invés de assinaturas digitais, aumentando o seu desempenho ao usar criptografia simétrica em vez de criptografia assimétrica. O PBFT executa em rodadas (*rounds*), sendo que em cada rodada (*round*) um participante do consenso é escolhido líder e tentará fazer da sua proposta de novo bloco para a corrente de blocos. O PBFT necessita de $n = 3f + 1$ participantes do consenso, com réplicas da corrente de blocos, para tolerar até f falhas bizantinas e utiliza quóruns de $2f + 1$ participantes de consenso em cada passo do protocolo. Também é necessário um modelo de sistema de comunicação parcialmente síncrono para garantir terminação. A Figura 3.8 ilustra os passos do protocolo em uma execução normal, i.e., quando o líder é correto e o sistema está em um período de sincronia. Os passos do protocolo são os seguintes:

- o líder envia uma mensagem de PRE-PREPARE para todos os participantes com a sua proposta de novo bloco, p;
- os participantes do consenso fazem algumas validações para definir se a proposta de novo bloco, p, enviada pelo líder é válida. Em caso afirmativo aceitam esta proposta e enviam uma mensagem de PREPARE contendo a proposta de novo bloco, p, para todos os outros participantes do consenso;
- quando um participante do consenso receber $\lceil \frac{n+f}{2} \rceil$ mensagens PREPARE para uma mesma proposta de novo bloco, p, ele envia uma mensagem de COMMIT contendo p para todos os outros participantes do consenso;
- quando um participante do consenso receber $\lceil \frac{n+f}{2} \rceil$ mensagens de COMMIT para uma mesma proposta de novo bloco, p, então a proposta p, é decidida.

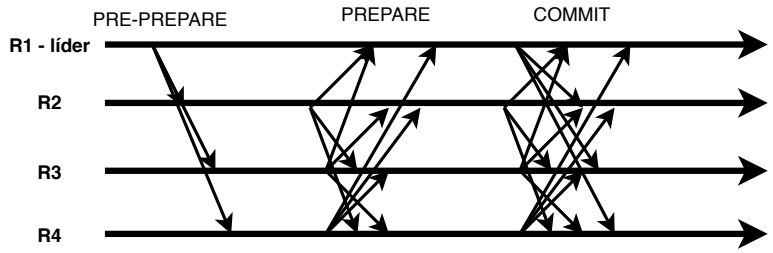


Figura 3.8: Execução normal do PBFT.

Quando o líder for malicioso e fizer propostas divergentes, um quórum de mensagens PREPARE ou COMMIT para uma mesma proposta não é obtido, o mesmo ocorre em um período de perturbação do sistema em que as mensagens não recebidas até um limite de tempo estipulado. Nestes casos, um protocolo de troca de visão é executado e uma nova rodada (*round*) é iniciada com um novo líder.

O Protocolo MinBFT [Veronese et al. 2013] requer a mesma quantidade de réplicas e passos de comunicação dos protocolos que toleram apenas falhas por parada, como por exemplo, o Paxos [Lamport 1998]. Para isso, faz uso de componentes seguros nos participantes do consesno para restringir as ações de participantes maliciosos. Os componentes seguros propostos para auxiliar no desenvolvimento de protocolos que tolerem falhas maliciosas possuem diferenças consideráveis tanto em aspectos de implementação quanto de desempenho. Quanto à implementação, estes componentes podem ser: i) implementados de forma distribuída, como o *Trusted Timely Computing Base* (TTCB) [Correia et al. 2004]; ii) locais baseados em memória, como o *Attested Append-Only Memory* (A2M) [Chun et al. 2007]; e iii) locais baseados em um simples contador, como o *Unique Sequential Identifier Generator* (USIG) [Veronese et al. 2013].

Basicamente, um atacante não consegue acessar estes componentes, mesmo que consiga invadir um servidor. Com isso, é possível projetar protocolos que restringem as ações que um processo malicioso (invadido) pode executar sem ser descoberto. O MinBFT utiliza o componente seguro USIG, que é muito simples e de fácil implementação e utilização no sistema. Este componente é local em cada replica e é usado para

atribuir um identificador único para uma mensagem, além de assiná-la. Para cada replica, os identificadores atribuídos às mensagens são únicos (um mesmo identificador nunca é atribuído a duas ou mais mensagens), monotônicos (o identificar atribuído a uma mensagem nunca é menor do que o anterior) e sequenciais (o identificador atribuído a uma mensagem sempre é o sucessor do anterior).

A interface definida para acessar este serviço é a seguinte [Veronese et al. 2013]:

- `createUI(m)`: retorna um certificado que contém um identificador único `UI` e a prova de que este identificador foi criado por este componente e atribuído a `m`. O identificador é basicamente um contador monotonicamente crescente que é incrementado a cada chamada desta função. Já a prova envolve criptografia e pode ser baseada em um *hash* (*Hash-based Message Authentication Code* – HMAC) ou em criptografia assimétrica (assinaturas digitais).
- `verifyUI(PK,UI,m)`: verifica se o identificador único `UI` é válido para `m`.

Com o uso deste componente, uma replica não consegue enviar duas mensagens diferentes para processos diferentes com um mesmo identificador. Assim, basta que cada replica mantenha armazenado o identificador da última mensagem recebida de uma replica para saber qual é o próximo identificador esperado, e assim reduz-se as ações de uma replica maliciosa: ou envia a mesma mensagem (que pode conter qualquer valor) para todos as outras replicas ou não envia nada.

Como o PBFT, o MinBFT também executa em *rounds*. Porém, o MinBFT necessita de apenas $n = 2f + 1$ replicas para tolerar até f falhas bizantinas e utiliza quóruns de $f + 1$ replicas em cada passo do protocolo. Também é necessário um modelo de sistema parcialmente síncrono para garantir terminação. A Figura 3.9 ilustra os passos do protocolo em uma execução normal, i.e., quando o líder é correto e o sistema está em um período de sincronia. Os passos do protocolo são os seguintes:

- O líder envia uma mensagem de PREPARE para todas as réplicas com a sua proposta p. Neste passo, o USIG é utilizado para evitar que o líder envie propostas diferentes para as replicas, i.e., um UI (identificar único é associado a esta mensagem).
- As replicas fazem algumas verificações para definir se a proposta p é válida (por exemplo, verificam se o UI é válido), e em caso afirmativo enviam uma mensagem de COMMIT contendo p para todas as outras replicas. Neste passo, o USIG também é utilizado para evitar que uma replica maliciosa envie mensagens diferentes para diferentes replicas.
- Quando uma replica receber $f + 1$ mensagens de COMMIT válidas (UI válido) para uma mesma proposta p, então p é decidido.

Como no PBFT, pode ser que nem todas as replicas consigam decidir em um *round*, neste caso um protocolo de troca de visão também é executado para inicializar um novo *round*, até que todas as replicas consigam decidir pelo mesmo valor.

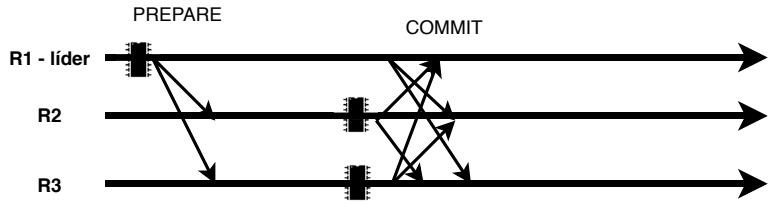


Figura 3.9: Execução normal do MinBFT.

O Protocolo Zyzzyva - é utilizado para implementar uma camada de ordenação em aplicações distribuídas. Por exemplo, podem implementar a camada de ordenação de uma Replicação Máquina de Estados ou a camada de ordenação que mantém a consistência de uma corrente de blocos. No contexto de uma corrente de blocos, podemos pensar como um minerador que envia um bloco para ser adicionado na corrente, e o protocolo de consenso define a ordem em que os blocos são adicionados na mesma. Neste cenário, o Zyzzyva [Kotla et al. 2009] utiliza execução especulativa para melhorar o desempenho destes protocolos, i,e., o bloco é adicionado na corrente antes da ordenação e o minerador verificará que não ocorreu desacordo. Usando este padrão de comunicação, o Zyzzyva requer menos passos de comunicação e melhora o desempenho do sistema. O Zyzzyva utiliza as mesmas suposições do PBFT em relação ao número de replicas e modelo de sistema. A Figura 3.10 apresenta o padrão de comunicação para o caso normal de execução, que é como segue:

- Primeiramente, o minerador (cliente) envia o bloco (requisição) a ser adicionado (executado) na corrente de blocos para a replica líder.
- A réplica líder define uma ordem e envia este bloco para todas as outras réplicas.
- As réplicas especulativamente incluem este bloco na corrente de blocos e enviam uma resposta para o minerador, que aguarda por todas as respostas $(3f + 1)$ e verifica se ocorreu desacordo. Note que isso somente ocorre quando o líder for malicioso e envia ordens diferentes de inclusão de blocos na corrente para diferentes replicas. Também é utilizado um *timeout* no minerador para verificar se seu bloco foi incluído na corrente. Caso ocorra um *timeout* antes de receber as respostas ou o minerador defina que ocorreu um desacordo, protocolos adicionais são utilizados e o minerador envia uma mensagem para as replicas com as informações recebidas. Neste caso as replicas podem executar uma troca de visão e inicializar um novo *round* (caso ocorreu desacordo) ou apenas executar o *commit* daquela operação (caso ocorreu o *timeout*, mas o minerador tinha recebido pelo menos $2f + 1$ respostas iguais).

Vale destacar que o processo a enviar o bloco para inclusão na corrente poderia ser o próprio líder (como exemplificado nas figuras anteriores para o PBFT e MinBFT), neste caso fica claro a redução na quantidade de passos de comunicação necessárias para incluir um bloco na corrente. De qquer forma, após a inclusão ainda existe um passo adicional para verificar se ocorreu desacordo. De forma resumida, caso tenha ocorrido desacordo, um protocolo é utilizado para reiniciar um novo *round* com um novo lider.

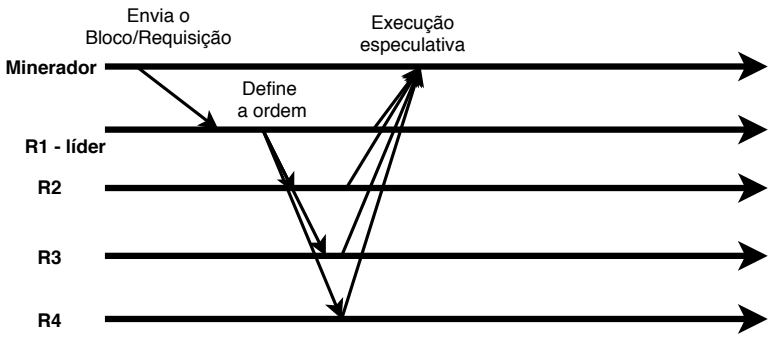


Figura 3.10: Execução normal do Zyzzyva.

O Protocolo de Consenso BFT entre Participantes Desconhecidos (*BFT Consensus with Unknown Participants* BFT-CUP) [Alchieri et al. 2008, Alchieri et al. 2018, Cavin et al. 2004, Cavin et al. 2005, Greve and Tixeuil 2007, Greve and Tixeuil 2010] foi proposto para a solução do problema do consenso em ambientes dinâmicos e auto-organizáveis. Nestes protocolos, cada participante inicialmente conhece apenas um subconjunto dos participantes da rede e, baseado neste conhecimento inicial, expandem este conhecimento a respeito dos outros participantes do sistema para então estabelecer um consenso.

O conhecimento inicial de cada participante é encapsulado em um módulo local chamado de detectores de participação. A união do conhecimento inicial obtido por cada participante da computação distribuída pode ser representado como um grafo direcionado de conectividade por conhecimento, no qual os vértices representam participantes e cada aresta representa o conhecimento entre dois participantes. É importante não confundir este grafo de conhecimento com o grafo de conectividade da rede. Por exemplo, na Figura 3.11, o participante 4 conhece o participante 9, enquanto que os participantes 5 e 6 se conhecem um ao outro. Também é importante perceber que este grafo não é conhecido pelos participantes do sistema, pois cada um apenas tem a informação sobre o seu conhecimento inicial e mesmo que posteriormente este conhecimento seja expandido, cada participante consegue obter apenas uma visão parcial do sistema.

Dado este conhecimento inicial, os protocolos de consenso primeiramente tentam expandir este conhecimento através de uma busca no grafo. Depois disso, é definido o componente poço do grafo, i.e., um subgrafo composto pelos participantes que se conhecem e compartilham a mesma visão do sistema (veja a Figura 3.11). Após a definição desta componente, os participantes no poço executam um protocolo de consenso tradicional (por exemplo, o PBFT [Castro and Liskov 2002]) e enviam o valor da decisão para os outros participantes do sistema fora do poço. Os trabalhos existentes nesta área estabeleceram o conhecimento inicial mínimo que os participantes devem obter para resolver o consenso considerando diferentes modelos de sistemas: síncrono sem falhas [Cavin et al. 2004], síncrono com falhas por parada [Cavin et al. 2005], parcialmente síncrono com falhas por parada [Greve and Tixeuil 2007], e parcialmente síncrono com falhas Bizantinas [Alchieri et al. 2008, Alchieri et al. 2018].

O BFT-SMaRt [Bessani et al. 2014] é uma implementação Java robusta para Replicação Máquina de Estados [Schneider 1990] que utiliza um protocolo consenso para

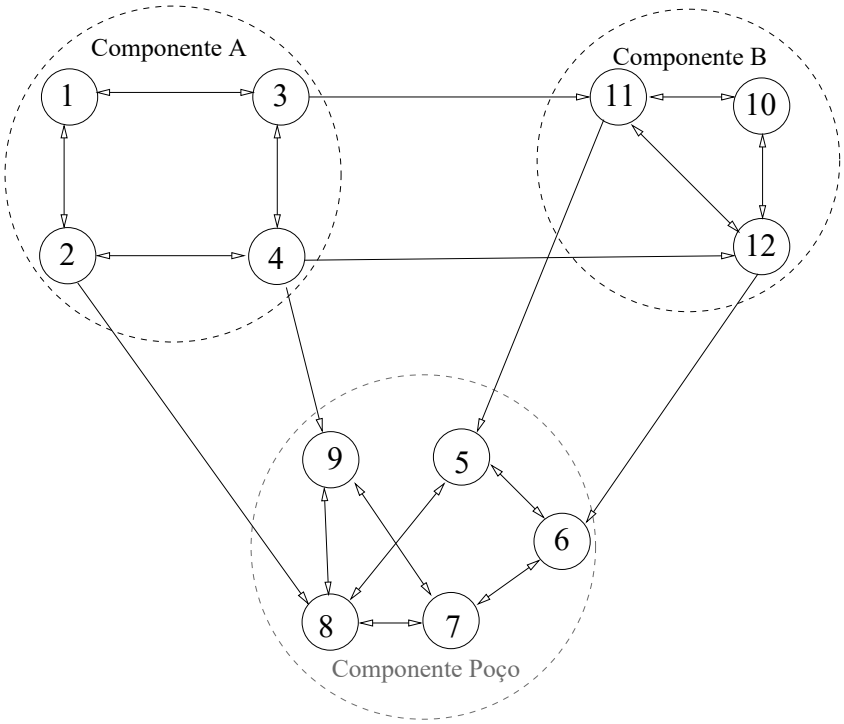


Figura 3.11: Grafo de conectividade por conhecimento.

ordenar requisições. O BFT-SMaRt pode ser configurado para tolerar tanto falhas por parada, e neste caso utiliza um protocolo similar ao Paxos [Lamport 2001], quanto bizantinas, e neste caso utiliza um protocolo similar ao PBFT [Castro and Liskov 2002]. Esta biblioteca de replicação foi desenvolvida buscando-se um alto desempenho em execuções livres de falhas e corretude nos casos em que replicas faltosas estarem presentes no sistema. Além disso, o BFT-SMaRt é a primeira implementação de Replicação Máquina de Estados que permite reconfigurações no conjunto de replicas [Lamport et al. 2010] e fornece um suporte eficiente e transparente para serviços duráveis [Bessani et al. 2013].

Apesar de não ser um protocolo de consenso, o BFT-SMaRt pode ser utilizado para ordenar os pedidos de inserção de blocos em uma cadeia de blocos, pois todas as requisições são entregues na mesma ordem em todas as replicas do sistema. Note que para isso é utilizado um protocolo de consenso subjacente, conforme já comentado. O BFT-SMaRt foi incorporado como a camada de ordenação no Hyperledger Fabric v1.3 [Sousa et al. 2018].

3.3.6. Protocolos de Consenso Híbridos

Os protocolos híbridos procuram combinar dois ou mais modelos de consenso com o objetivo de herdar as vantagens dos tipos de consenso usados. Um exemplo desta classe de protocolos de consenso é o protocolo Tendermint que é detalhado a seguir.

O Protocolo Tendermint [Kwon 2014, Buchman 2016] é um protocolo de consenso tolerante a falhas bizantinas, que não requer mineração por prova de trabalho e oferece proteção contra gasto duplo, simultaneamente apresentando uma taxa de milhares de tran-

sações por segundo e segurança baseada em fatores intrínsecos. O protocolo atua de modo similar a prova de participação (*proof of stake*), com os nós responsáveis por adicionar novos blocos a rede oferecendo parte de seus ativos como um modo de atuarem no processo de efetivação. Diferentemente da prova de participação, o Tendermint foi desenvolvido tendo ataques de nada a perder (*nothing at stake*) em mente, implementando medidas de segurança para proteger a rede destes ataques. Como o protocolo é tolerante a falhas bizantinas, garante-se a segurança da corrente contra até 1/3 de participantes bizantinos.

Cada bloco adicionado a corrente é constituído por 3 partes: o cabeçalho, os validadores do bloco anterior e as transações inclusas nesse bloco. Para a determinação do *hash* do bloco, primeiramente utiliza-se uma função resumo em cada uma dessas partes; a seguir, esta função é utilizada novamente sobre os hashs obtidos. Este hash do bloco é utilizado em uma Árvore de Merkle (Merkle Tree) de modo a permitir a verificação de qualquer bloco na corrente, garantindo assim sua auditabilidade.

Validadores são contas que possuem moedas em um depósito de títulos (*bond deposit*), com essas moedas determinando o poder de votação do validador. Para adicionar moedas ao depósito, o usuário deve transmitir uma transação de título (*bond transaction*), indicando à rede quantas moedas serão utilizadas com este propósito. Esses validadores participam do consenso difundindo assinaturas criptográficas, ou votos, para concordar na adição de um bloco à corrente. Essas moedas podem ser transferidas e gastas, desde que não estejam armazenadas no depósito de títulos; para removê-las desse depósito, deve-se transmitir uma transação de desvínculo (*unbond transation*) com este objetivo, e aguardar por um período pré-determinado denominado período de desvínculo (*unbonding period*). O funcionamento das transações de título e desvínculo é apresentado na Figura 3.12. O poder de votação da rede é igual a soma do poder de votação de todos os participantes.

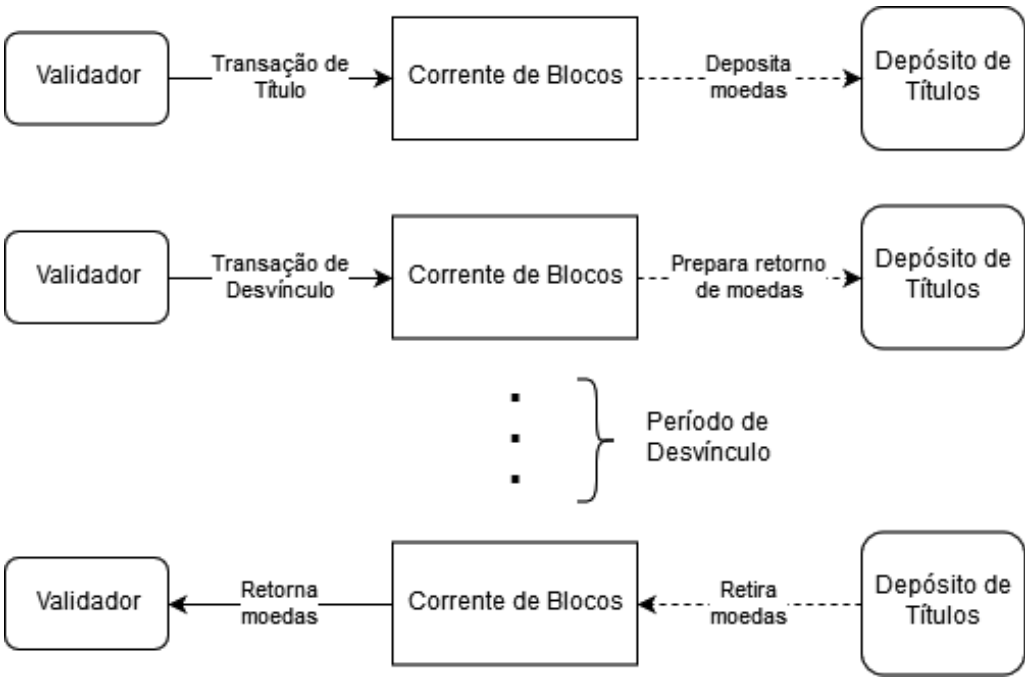


Figura 3.12: Transações de título e desvínculo do protocolo Tendermint, utilizadas para modificar o poder de votação de cada validador.

O processo de adição de novos blocos a corrente se dá através de rodadas. Cada rodada é composta por 3 passos: proposta (*propose*), pré-voto (*prevote*) e pré-efetivação (*precommit*), com os validadores enviando votos em cada passo da rodada. Existem 3 tipos de votos: o pré-voto (*prevote*), a pré-efetivação (*precommit*) e a efetivação (*commit*). Um bloco é dito efetivado pela rede quando assinado e difundido por mais de 2/3 do poder de votação dos validadores.

As etapas que constituem uma rodada estão apresentadas na Figura 3.13. Cada rodada é feita de modo que um proponente (*proposer*) seja escolhido entre os validadores, onde os que possuem maior poder de votação são escolhidos com maior frequência. Inicialmente o proponente escolhido anuncia o bloco através de uma proposta, com os outros nós confirmando sua validade, e que este foi recebido a tempo, através de um pré-voto. Quando mais de 2/3 do poder de votação da rede realizar o pré-voto para este bloco, realiza-se então a pré-efetivação antes que se inicie o processo de efetivação do bloco. Esta também ocorre quando mais de 2/3 do poder de votação da rede concordar na pré-efetivação do bloco.

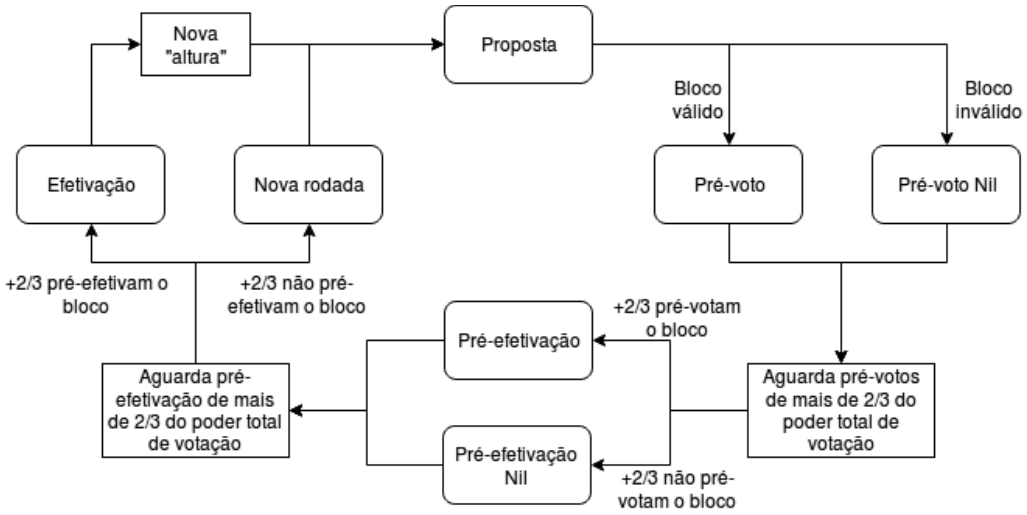


Figura 3.13: Etapas de uma rodada do protocolo Tendermint.

Um bloco é considerado efetivado quando pelo menos 2/3 do poder total de votação assina votos de efetivação para esse bloco. Uma bifurcação (*fork*) ocorre quando dois ou mais blocos na mesma "altura" são assinados por uma maioria de 2/3 do poder de votação, indicando que pelo menos 1/3 dos validadores votaram com duplicidade. Quando isto ocorre, uma transação de evidência (*evidence transaction*) é gerada, destruindo as moedas de título dos validadores culpados. O período de desvínculo garante que validadores maliciosos ainda possuirão suas moedas no depósito de título quando a transação de evidência for gerada, desse modo assegurando que a punição pelo ataque será devidamente realizada. Essa medida evita que validadores votem para efetivar diferentes blocos em uma única rodada, devido ao risco de ocorrer prejuízo financeiro, desencorajando assim ataques de nada a perder.

3.3.7. Protocolo baseado em Grafo Acíclico Direcionado - IOTA Tangle

O IOTA [Popov 2017] é uma criptomoeda construída para atender a micropagamentos máquina a máquina (*machine to machine* - M2M) característicos de um ambiente de Internet das Coisas. Seus mecanismos de pagamento e protocolo de consenso, formalizados por Popov em 2016 [Popov 2017], baseiam-se em uma estrutura de dados inovadora chamada *Tangle*. A principal inovação do *Tangle* é a estrutura de livro-razão distribuído, que reúne as transações da rede em um grafo acíclico direcionado (*directed acyclic graph* - DAG) em vez de utilizar uma corrente de blocos. Além disso, o Tangle elimina a distinção entre clientes e mineradores: todos os usuários do sistema devem realizar trabalho para emitir uma nova transação. Uma característica notável do *Tangle* em comparação ao consenso em corrente de blocos é que diferentes participantes na rede podem ter as diferentes visões das transações. Esta característica contrasta fortemente com a visão global da corrente de blocos, na qual todas as transações são idênticas em qualquer participante.

A Figura 3.14 mostra um exemplo da estrutura de dados do *Tangle*. Cada vértice do grafo representa uma transação e cada aresta representa o resultado da validação de uma transação. O usuário deve confirmar ao menos duas transações não-confirmadas para adicionar sua transação à *Tangle*[18]. As transações não-confirmadas são chamadas de "pontas"(*tips*) do *Tangle*. Para adicionar uma transação à rede, o usuário adiciona os resumos das duas pontas escolhidas à sua transação, resolve um desafio baseado em prova de trabalho, e difunde o resultado na rede. A prova de trabalho, neste caso, tem dificuldade bem menor que a do Bitcoin e serve apenas como um mecanismo para prevenir *spam* de transações. O procedimento de adição de uma transação cria duas novas arestas direcionadas no grafo que confirmam as transações anteriores e funcionam como uma versão generalizada da sequência de funções resumo da corrente de blocos. O IOTA Tangle não recompensa financeiramente os validadores de transações, pois o incentivo para validar transações é adicionar sua própria transação à rede. Todas as moedas trocadas na rede são criadas na primeira transação do sistema, chamada de transação gênesis.

Um dos conceitos fundamentais no IOTA é o peso de transações: o peso individual e peso acumulado. O peso individual de uma transação é um peso proporcional aos recursos gastos por um usuário para emitir a transação. A ideia dos pesos individuais é diferenciar transações mais importantes, que possuem maior peso individual, de transações com menor importância, que possuem menor peso individual. O peso acumulado de uma transação é definido pelo seu peso individual mais a soma de todos os pesos individuais das transações que aprovam, direta ou indiretamente, a transação. Uma transação aprova diretamente outra se há uma aresta conectando as duas, e indiretamente se há pelo menos um caminho composto de mais de uma aresta entre as duas transações.

O consenso do IOTA Tangle não é realizado *a priori* não precisam obter consenso sobre quais transações podem ser adicionadas ao grafo. Todas as transações são adicionadas, bastando que o usuário resolva o desafio computacional necessário para encadear as transações. No entanto, quando há pontas conflitantes, cada usuário precisa decidir quais transações serão consideradas válidas para escolher duas pontas que serão aprova-

[18]Na implementação do IOTA, o número de confirmações necessárias para adicionar uma transação à rede é exatamente dois.

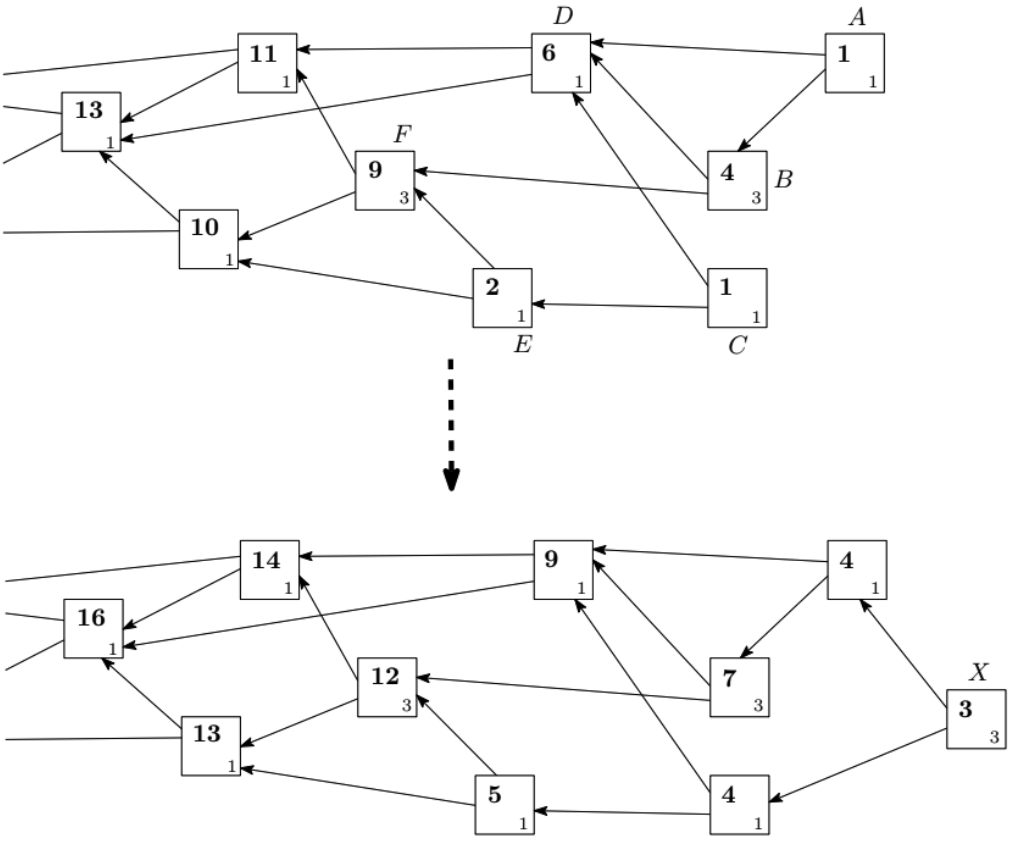


Figura 3.14: A adição de uma ponta, X, na estrutura de dados do *Tangle*. O peso individual de cada transação é representado no canto direito inferior e o peso acumulado é representado pelo número em negrito.

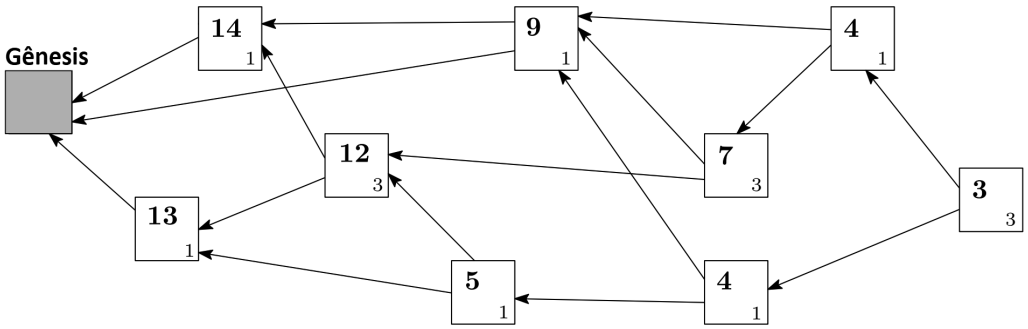


Figura 3.15: Um exemplo de estrutura de dados baseada em grafos acíclicos direcionados (DAG) do *Tangle*. Cada vértice do grafo representa uma transação e cada aresta representa o resultado da validação de uma transação. O número no canto inferior direito de cada transação representa o peso individual da transação, e o número ao centro, em negrito, representa o peso acumulado da transação.

das por sua nova transação. As pontas inválidas são ignoradas pelo usuário. O principal mecanismo para identificar pontas válidas é realizar múltiplas rodadas do algoritmo de seleção de pontas (*tip selection algorithm*) e verificar qual das duas pontas conflitantes tem a maior probabilidade de ser escolhida. Por exemplo, se uma ponta foi escolhida 95 vezes

em 100 rodadas do algoritmo de seleção, a ponta tem confança de 95%. O IOTA Tangle atualmente utiliza um algoritmo de seleção baseado em passeios aleatórios (*random walks*) e cadeias de Markov e Monte Carlo (*Markov-chain and Monte Carlo* - MCMC) que prioriza transações de maior peso acumulado.

Quando existem mais de duas pontas válidas disponíveis para seleção, o usuário escolhe as duas pontas de acordo com o peso acumulado de cada ponta.

3.4. Atividade Prática (*Hands-On*): Desenvolvimento de uma Corrente de Blocos para Telecomunicações

O objetivo da atividade prática é evidenciar a segurança proporcionada pelo uso de corrente de blocos em um ambiente de funções virtualizadas de rede (*Virtualized Network Function* - VNF), um cenário comum de Internet do Futuro em telecomunicações. A corrente de blocos desta prática registra operações de criação e configuração de VNFs, garantindo confiabilidade, integridade e auditabilidade das informações armazenadas. A atividade prática utiliza, como estudo de caso, a arquitetura de fatiamento de rede (*network slicing*) apresentada na Figura 3.16 com dois tipos de corrente de blocos [Rebello et al. 2019c]. A arquitetura apresentada fornece a capacidade de auditoria de criação e gerenciamento de fatias, registrando todas as operações de orquestração da VNF em uma corrente de blocos de gerenciamento. Além disso, as propriedades de automação e transparência dos contratos inteligentes são apropriadas para criar fatias de rede fim-a-fim seguras que envolvem VNFs em vários domínios concorrentes, pois garantem que todos os nós da rede obedeçam ao mesmo conjunto de regras e que o código executado seja visível para qualquer nó participante. A demonstração usa a plataforma *Hyperledger Fabric* para a criação de uma corrente de blocos permissionada. O *Hyperledger Fabric* é uma plataforma de código aberto para o desenvolvimento de correntes de blocos permissionadas em um ambiente sem confiança entre os participantes.

3.4.1. A Arquitetura da Plataforma Hyperledger Fabric

O Hyperledger[19] é uma iniciativa colaborativa e de código aberto da Linux Foundation que objetiva desenvolver tecnologias baseadas em corrente de blocos para a indústria. A colaboração global inclui empresas de tecnologia da informação, mercado financeiro, Internet das Coisas, cadeias de suprimentos, saúde e financiamento.

Um dos diversos projetos inclusos na iniciativa Hyperledger é o *Hyperledger Fabric* [Androulaki et al. 2018], uma plataforma proposta e desenvolvida pela IBM para a implementação de redes permissionadas baseadas em corrente de blocos. O Hyperledger Fabric é o projeto mais maduro da iniciativa Hyperledger e o mais utilizado por empresas que desenvolvem aplicações privadas baseadas em corrente de blocos. A arquitetura modular e plugável do Fabric permite o uso de diferentes tipos de protocolos de consenso, diferentes linguagens de programação para a criação de contratos inteligentes e diferentes bancos de dados para armazenar a corrente de blocos e seu estado atual. Outra proposta do Hyperledger Fabric é criar redes nas quais podem coexistir diversas correntes de blocos isoladas, independentes, e com diferentes configurações, contratos inteligentes e participantes. As entidades que participam de uma rede baseada em correntes de blocos

[19]Disponível em https://www.hyperledger.org/

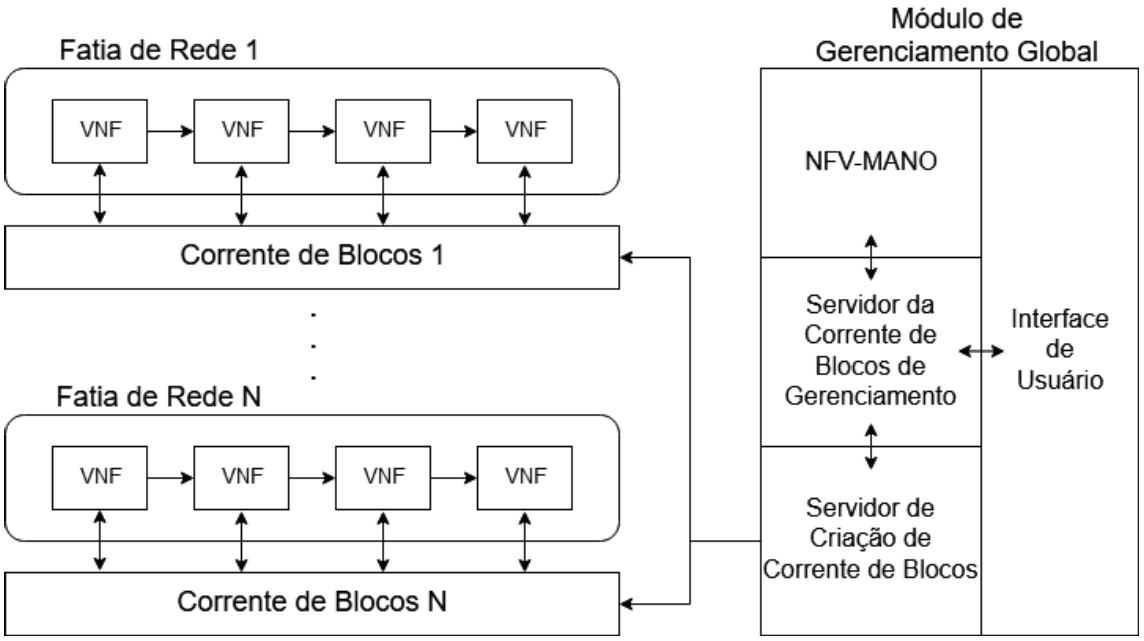


Figura 3.16: A arquitetura proposta por Rebello *et al.* baseada em corrente de blocos para fatiamento de rede [Rebello et al. 2019c]. O usuário interage com o módulo de gerenciamento global para criar fatias de rede seguras. Cada VNF em uma fatia de rede é conectada a uma corrente de blocos responsável por registrar solicitações de configuração e informações relevantes, conforme especificado pelo usuário.

do Fabric podem ter diferentes funções e permissões de acesso à corrente de blocos. As entidades com poder administrativo podem modificar as permissões de escrita e leitura das demais entidades em cada corrente de blocos através de transações de configuração, enquanto entidades comuns têm permissão para emitir transações que não sejam sensíveis ao funcionamento da rede, como uma transferência de recursos entre duas entidades. Há a possibilidade de definir políticas específicas e adaptadas para a validação de transações que exigem a assinatura por múltiplos validadores. Os desenvolvedores das correntes de blocos no Fabric podem configurar permissões de leitura e escrita para criar redes permissionadas, nas quais todos os nós da rede se conhecem. Isso é apropriado para ambientes empresariais ou de consórcio, tipicamente constituídos por até dezenas de nós. O Hyperledger Fabric fornece um serviço de identidade e associação de membros que gerencia os identificadores dos usuários, e autentica todos os participantes na rede automaticamente. Além disso, podem existir nós internos em cada organização/empresa. Todas essas configurações disponíveis no Hyperledger Fabric são adequadas e facilitam a implementação de correntes de blocos adaptadas a cada aplicação, com necessidades e características diferentes, em um ambiente empresarial.

Nós e canais são os conceitos-chave mais importantes de uma rede baseada em corrente de blocos permissionada do Hyperledger Fabric. Os nós representam as entidades que participam do processamento de uma transação ou mantêm uma cópia da corrente de blocos. O Hyperledger Fabric provê três tipos de nós: clientes, pares (*peers*) e ordenadores. Um cliente representa um usuário que envia transações aos pares para validação e assinatura, e transmite propostas de transação assinadas para o serviço de ordenação.

Os pares são um elemento fundamental da rede porque executam propostas de transação, validam transações e mantém os registros na corrente de blocos. Os pares também instanciam contratos inteligentes e armazenam o estado global, uma representação sucinta do estado mais recente da corrente de blocos. Os nós ordenadores formam coletivamente o serviço de ordenação, que é responsável por estabelecer a ordem total e o empacotamento de todas as transações em um bloco usando um protocolo de consenso. Os ordenadores não participam da execução da transação nem validam transações. O desacoplamento das funcionalidades de ordenação e validação aumenta a eficiência da rede, pois permite o processamento paralelo de cada fase. A Figura 3.17 descreve um exemplo de uma corrente de blocos permissionada com quatro organizações no Hyperledger Fabric. Cada organização recebe transações de clientes e as retransmite para os ordenadores após a validação pelos pares. Cada organização possui um único ordenador, garantindo a justiça no protocolo de consenso.

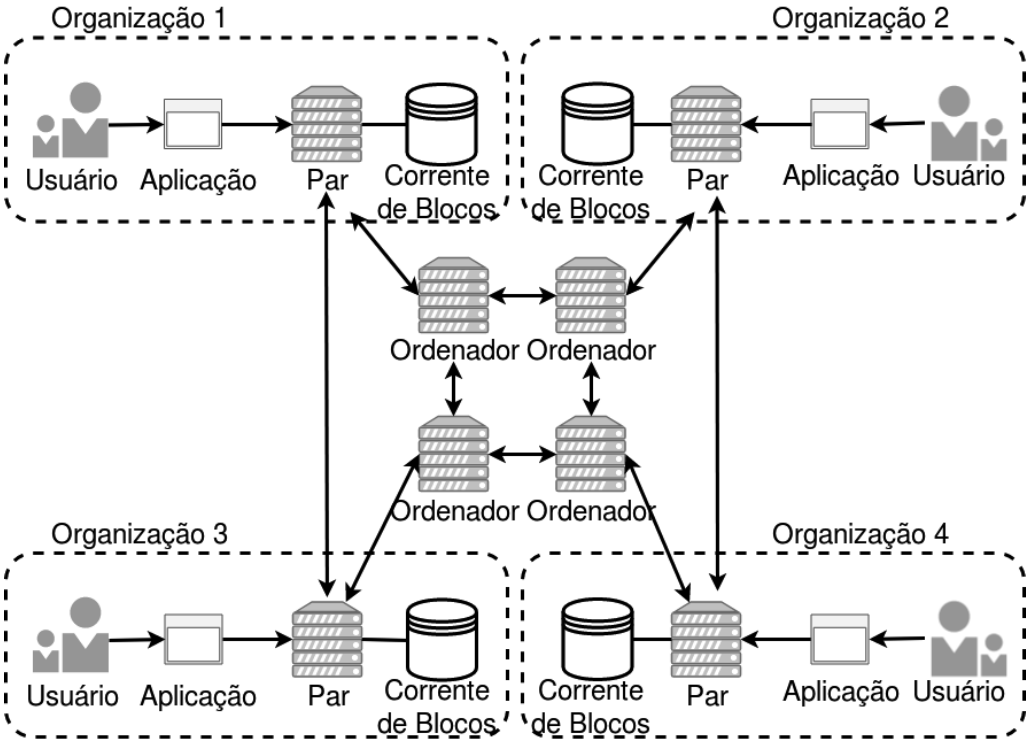


Figura 3.17: Um exemplo de corrente de blocos permissionada do Hyperledger Fabric. Usuários de cada organização usam aplicações para emitir transações assinadas e as submetem para validação nos pares. Os pares validam a transação e respondem à aplicação, que retransmite a transação validada para os ordenadores. Os ordenadores trocam mensagens para definir a ordenação global das transações recebidas em um intervalo de tempo e as adicionam em um novo bloco. Depois de um bloco ser proposto e aceito pelos ordenadores através do consenso, os pares atualizam a corrente de blocos e o estado global da rede.

Caminhos de mensagens diferentes, chamados canais, isolam as correntes de blocos. Um canal Hyperledger Fabric é uma sub-rede de comunicação privada e isolada entre um subconjunto de nós da rede específicos para fornecer privacidade e confidencialidade

às transações. Todos os dados transmitidos em um canal, incluindo transações, contratos inteligentes, configurações de associação e informações de canal, são invisíveis e inacessíveis a qualquer entidade externa a um canal. As mensagens trocadas em um canal são criptografadas. A funcionalidade do canal é ideal para a proposta de oferecer correntes de blocos personalizadas para diferentes serviços de rede, pois permite que os administradores dos canais estabeleçam diferentes formatos de bloco e transação, além do protocolo de consenso, para cada canal. Portanto, é possível usar canais para oferecer fatias de rede protegidas por correntes de blocos configuradas de forma específica. Formatos de transação são definidos em contratos inteligentes, chamados de *chaincode* no Hyperledger Fabric, escritos em Go, Node.js ou linguagem Java.

3.4.2. Configuração do Ambiente e Experimentação

Para realizar esta atividade, é necessário uma máquina com acesso à Internet que possua pelo menos 4 GB de memória RAM e processador Intel Core i3 ou equivalente. A arquitetura do sistema deve ser de 64 bits e recomenda-se o uso de um sistema operacional baseado em Linux por experiência de uso, maior facilidade para programação e por serem baseados em código aberto e gratuito. São disponibilizados dispositivos USB removíveis (*pen drives*) com imagens para criação de uma máquina virtual com sistema operacional *Debian 9.0 (Stretch)* que inclui todos os pré-requisitos necessários para utilização do Hyperledger Fabric já instalados. Vale ressaltar que o computador utilizado pelo participante deve possuir uma ferramenta de virtualização como o VirtualBox[20] ou VMWare Workstation Player[21] para utilizar a máquina virtual fornecida pelos autores.

Para dar maior dinamicidade à aula prática, os participantes que possuem acesso à Internet também podem baixar e instalar os pacotes necessários manualmente enquanto os demais participantes obtêm a imagem dos dispositivos removíveis. Os pré-requisitos estão listados na documentação do Hyperledger Fabric[22] e consistem dos seguintes itens:

- Instalar o cURL[23]. Para Debian, executar o comando `apt-get install curl`.
- Instalar o Docker[24] na versão 17.06.2-ce ou superior. Para Debian, executar os comandos na seguinte ordem:

  i) `apt-get install apt-transport-https ca-certificates gnupg2 software-properties-common`

  ii) `curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -`

  iii) `add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"`

[20]Disponível em https://www.virtualbox.org/
[21]Disponível em https://www.vmware.com/br/products/workstation-player.html
[22]Disponível em https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html
[23]Disponível em https://curl.haxx.se/download.html
[24]Disponível em https://www.docker.com/get-started

iv) `apt-get update && apt-get install docker-ce docker-ce-cli containerd.io`

- Instalar Docker Compose[25] na versão 1.14.0 ou superior. Para Debian, executar os comandos na seguinte ordem:

  i) `curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`

  ii) `chmod +x /usr/local/bin/docker-compose`

  iii) `ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose`

- Instalar a linguagem Go[26] na versão 1.11.x. Para Debian, executar os comandos na seguinte ordem:

  i) `wget https://dl.google.com/go/go1.11.5.linux-amd64.tar.gz`

  ii) `tar -C /usr/local -xzf go1.11.5.linux-amd64.tar.gz`

  iii) `export PATH=$PATH:/usr/local/go/bin && export GOPATH=$HOME/go && export PATH=$PATH:$GOPATH/bin`

  iv) `source $HOME/.profile`

- Python na versão 2.7. Verifique a versão com o comando `python --version`

- Instalar os executáveis e as imagens de Docker do Hyperledger Fabric versão 1.4.1. Para Debian, executar os comandos na seguinte ordem:

  i) `cd /root && git clone https://github.com/hyperledger/fabric-samples.git`

  ii) `cd fabric-samples`

  iii) `curl -sSL http://bit.ly/2ysbOFE | bash -s -- 1.4.1 1.4.1 0.4.15`

3.4.3. Instalação de uma Corrente de Blocos

Esta aula prática desenvolve um protótipo de aplicação que usa a plataforma Hyperledger Fabric [Androulaki et al. 2018] para implementar correntes de blocos entre organizações em ambientes sem confiança. Cada organização mantém uma réplica da corrente de blocos e pode acrescentar blocos através de um protocolo de consenso. O protótipo implementado segue a arquitetura de fatiamento de rede apresentada na Seção 3.4. O protótipo implementa cada nó da rede do Hyperledger Fabric como um contêiner em um único computador e envia transações simultaneamente. A aula prática implementa dois contratos inteligentes[27] escritos em Go, que são executados em todos os nós da rede [Alvarenga et al. 2018, Rebello et al. 2019a].

[25] Disponível em https://docs.docker.com/compose/install

[26] Disponível em https://golang.org/dl/

[27] O código completo está disponível em https://github.com/gta-ufrj/hpsr-smart-contracts

```
struct instructionTransaction
{
    command                 string
    transactionType         string
    transactionName         string
    issuer                  string
}
initialize queue
initInstruction (instruction <command,name,issuer>)
{
    if instruction is not unique or instruction is not well-formatted:
        return error
    putState (instruction.name, instruction)
    put (transactionID, queue)
    notify orchestrator
    return success
}
```

Lista 3.1: Parte do pseudocódigo do contrato inteligente que emite transações de instrução. O campo de comando contém a operação de orquestração a ser executada por um orquestrador. O contrato estabelece uma fila de instruções pendentes a serem processadas pelo orquestrador.

O primeiro contrato inteligente, parcialmente descrito na Lista 3.1, gerencia autonomamente o gerenciamento e a orquestração de VNF através de transações de instrução e resposta. Quando um cliente solicita uma fatia, o servidor da corrente de blocos de gerenciamento emite uma transação de instrução com o comando de instrução. O contrato coloca a transação de instrução em uma fila de transações pendentes de instrução. O código notifica o módulo NFV-MANO, que executa a instrução pendente e envia a saída do comando para o servidor da corrente de blocos de gerenciamento. O módulo NFV-MANO emite uma transação de resposta que inclui um campo contendo o identificador da transação de instrução correspondente. Isso fornece a rastreabilidade de cada transação executada na corrente de blocos e, portanto, possibilita a responsabilização de entidades maliciosas.

O segundo contrato inteligente, descrito na Lista 3.2, define e atualiza a configuração de uma VNF. Um cliente emite uma transação para a corrente de blocos conectada a cada VNF em uma fatia de rede. A transação contém um texto descritivo com a configuração associada no campo de descrição, assim como os dados de configuração no campo de configuração.

O protótipo usa os certificados de autoridade (*Certificate Authorities* - CA) do Hyperledger Fabric para criar e gerenciar certificados digitais em cada nó da rede de corrente de blocos. Certificados digitais garantem auditabilidade e que somente nós certificados e autorizados podem participar da rede de corrente de blocos.

Este parágrafo inicia um roteiro a ser executado pelos participantes desta aula prática realizarem a atividade prática descrita. A atividade prática foi desenvolvida utilizando a versão 1.4.1 do Hyperledger Fabric. Os comandos de versões anteriores ou posteriores podem ser incompatíveis com a versão utilizada nesta atividade. O roteiro

```
struct configurationTransaction
{
    configurationIdentifier string
    versionIdentifier       string
    description             string
    configuration           string
    transactionType         string
    transactionName         string
    issuer                  string
}
initConfiguration (configuration <description,configuration,name,issuer
    >){
    if configuration is not unique or configuration is not well-formed:
        return error
    putState (configuration.name, configuration)
    return success
}
```

Lista 3.2: Pseudocódigo parcial para emitir transações de configuração. O campo configurationIdentifier contem um identificador único para a configuração.

deve ser seguido utilizando o usuário *root*. Primeiramente, o participante deve criar o diretório `contract` dentro do diretório `fabric-samples/chaincode` e baixar o contrato inteligente que será utilizado ao longo desta atividade. De dentro do diretório `fabric-samples`, o participante deve executar:

```
cd chaincode && mkdir contract && cd contract

wget https://github.com/gta-ufrj/hpsr-smart-contracts/raw/
master/contract.go
```

O próximo passo é gerar todo o material criptográfico a partir do arquivo `crypto-config.yaml` e da ferramenta `cryptogen`. O arquivo `crypto-config.yaml` contém as informações referentes aos participantes iniciais da rede. As configurações presentes nesse arquivo informam os endereços de cada nó da rede, o número de ordenadores, as organizações participantes da rede e a quantidade de participantes por organização. Esses dados são utilizados pela ferramenta `cryptogen` para gerar pares de chaves assimétricas e certificados para os nós, que serão utilizados no processo de validação, controle de acesso e não-repúdio às transações realizadas por um membro da rede. De dentro do diretório `fabric-samples/first-network`, o usuário deve gerar os certificados que serão usados na rede utilizando a ferramenta `cryptogen` que cria um diretório chamado `crypto-config` contendo os certificados e chaves da rede:

```
../bin/cryptogen generate --config=<caminho do arquivo de
configuração>
```

- caminho do arquivo de configuração: ./crypto-config.yaml

Em seguida, o usuário deve configurar a rede do Hyperledger Fabric utilizando o arquivo `configtx.yaml` e a ferramenta `configtxgen`. Nesse arquivo estão contidas

informações sensíveis à rede como o tipo de consenso utilizado, tempo máximo para a criação de um novo bloco, tamanho máximo das transações contidas em um bloco, tamanho máximo em *bytes* do bloco, políticas para validação de transações, e perfil de cada nó. A ferramenta `configtxgen` utiliza o arquivo `configtx.yaml` para criar o bloco gênesis que contém as informações de configuração da rede. Portanto para configurar a rede, o usuário deve indicar o diretório do arquivo de configuração `configtx.yaml` para a ferramenta `configtxgen`. Feito isso, utilize a ferramenta `configtxgen` para gerar o bloco gênesis no diretório `channel-artifacts`. Esses dois passos são realizados da seguinte forma:

```
export FABRIC_CFG_PATH=$PWD
```

```
../bin/configtxgen -profile <perfil do canal>
-channelID <nome do canal> -outputBlock
./channel-artifacts/genesis.block
```

- perfil do canal: SampleDevModeKafka
- nome do canal: byfn-sys-channel

Para a criação do canal, o seguinte comando deve ser executado:

```
export CHANNEL_NAME=<nome do canal> &&
../bin/configtxgen -profile <perfil do canal>
-outputCreateChannelTx ./channel-artifacts/channel.tx
-channelID $CHANNEL_NAME
```

- perfil do canal: TwoOrgsChannel
- nome do canal: jaichannel[28]

O comando cria um arquivo chamado `channel.tx` contendo as configurações do canal dentro do diretório `channel-artifacts`.

Após a criação do arquivo com as configurações do canal, o usuário deve definir os pares-âncora (*anchor peers*). Pares-âncora conectam uma organização a outra. Pares de uma organização comunicam com os pares-âncora para descobrir pares de outras organizações. Cada organização possui ao menos um par-âncora. O comando para a definição do par-âncora para as organizações configuradas deve ser executado para cada organização listadas nos arquivos de configuração:

```
../bin/configtxgen -profile <perfil do canal>
-outputAnchorPeersUpdate ./channel-artifacts/<nome da
organização>anchors.tx -channelID $CHANNEL_NAME -asOrg
<nome da organização>
```

- perfil do canal: TwoOrgsChannel

[28]O nome do canal deve conter apenas letras minúsculas.

- nome da organização: Org1MSP

Como o exemplo desta atividade usa duas organizações:

```
../bin/configtxgen -profile <perfil do canal>
-outputAnchorPeersUpdate ./channel-artifacts/<nome da
organização>anchors.tx -channelID $CHANNEL_NAME -asOrg
<nome da organização>
```

- perfil do canal: TwoOrgsChannel
- nome da organização: Org2MSP

A rede de corrente de blocos do Hyperledger Fabric usa contêineres como nós. O usuário deve usar um arquivo de configuração em conjunto com a ferramenta *Docker Compose* para criar a rede e seus participantes. A plataforma Fabric disponibiliza os arquivos de configuração `docker-compose-cli.yaml` e `docker-compose-kafka.yaml` para facilitar a configuração e instanciação de contêineres. Para instanciar os contêineres, execute o seguinte comando:

```
docker-compose -f docker-compose-cli.yaml -f
docker-compose-kafka.yaml up -d
```

Para acessar o contêiner cliente:

```
docker exec -it cli bash
```

Agora, o usuário deve passar as configurações do canal criada anteriormente para criar o canal do Hyperledger Fabric:

```
export CHANNEL_NAME=jaichannel
```

```
peer channel create -o <nome do ordenador>:<porta do
ordenador> -c $CHANNEL_NAME -f ./channel-artifacts/
channel.tx --tls --cafile <caminho do certificado>
```

- nome do ordenador: orderer0.example.com
- porta do ordenador: 7050
- caminho do certificado: /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto /ordererOrganizations/example.com/orderers/orderer0.example.com /msp/tlscacerts/tlsca.example.com-cert.pem

Para inicializar um par no canal criado, os usuários devem executar os seguintes comando:

```
export CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/
hyperledger/fabric/peer/crypto/peerOrganizations/
org1.example.com/users/Admin@org1.example.com/msp
```

```
export CORE_PEER_ADDRESS=peer0.org1.example.com:7051
```

```
export CORE_PEER_LOCALMSPID="Org1MSP"

export CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

peer channel join -b jaichannel.block
```

Os participantes devem repetir os cinco comandos acima mudando o par e a organização. Como o exemplo desta atividade usa quatro pares, os comandos devem ser modificados para atender aos peer0 e peer1 da organização Org1 e aos peer0 e peer1 da organização Org2.

Com todos os pares adicionados, é necessário atualizar o canal para escolher os pares-âncoras de cada organização. Uma atualização no canal do Hyperledger Fabric adiciona uma informação de configuração do canal. Neste exemplo, os pares-âncoras são o peer0 da organização Org1 e peer0 da organização Org2.

```
export CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/
hyperledger/fabric/peer/crypto/peerOrganizations/
org1.example.com/users/Admin@org1.example.com/msp

export CORE_PEER_ADDRESS=peer0.org1.example.com:7051

export CORE_PEER_LOCALMSPID="Org1MSP"

export CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

peer channel update -o <ordenador> -c
$CHANNEL_NAME -f ./channel-artifacts/<org name>anchors.tx
--tls --cafile <caminho do certificado>
```

- ordenador: orderer0.example.com:7050
- caminho do certificado: /opt/gopath/src/github.com/hyperledger/fabric/peer /crypto/ordererOrganizations/example.com/orderers/orderer0.example.com /msp/tlscacerts/tlsca.example.com-cert.pem

Os participantes devem repetir os cinco comandos acima para atender ao peer0 da organização Org2.

Agora, é necessário instalar o contrato em cada par que executa e apoia as transações.

```
export CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/
hyperledger/fabric/peer/crypto/peerOrganizations/
org1.example.com/users/Admin@org1.example.com/msp

export CORE_PEER_ADDRESS=peer0.org1.example.com:7051

export CORE_PEER_LOCALMSPID="Org1MSP"
```

```
export CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

go get -u github.com/golang-collections/go-datastructures
/queue && peer chaincode install -n <nome do chaincode>
-v 1.0 -p <diretório do chaincode>
```

- nome do chaincode: mycc
- diretório raiz do chaincode: github.com/chaincode/contract

Os participantes devem repetir os cinco comandos acima mudando o par e a organização. Como o exemplo desta atividade usa quatro pares, os comandos devem ser modificados para atender aos peer0 e peer1 da organização Org1 e aos peer0 e peer1 da organização Org2.

Em seguida, o participante deve instanciar o *chaincode* no canal e também definir a política de endosso (*endorsement*) do canal:

```
peer chaincode instantiate -o <ordenador>
--tls --cafile <caminho do certificado> -C $CHANNEL_NAME
-n mycc -v 1.0 -c <mensagem em JSON para o chaincode>
-P <política do canal>
```

- ordenador: orderer0.example.com:7050
- caminho do certificado: /opt/gopath/src/github.com/hyperledger/fabric/peer /crypto/ordererOrganizations/example.com/orderers/orderer0.example.com /msp/tlscacerts/tlsca.example.com-cert.pem
- mensagem em JSON para o chaincode: ‘{"Args":["init"]}’
- política do canal: "AND (‘Org1MSP.peer’,‘Org2MSP.peer’)"

Neste caso, a política descrita acima define que uma transação efetuada no canal deve ser endossada por um par pertencente à organização Org1 e por um par pertencente à organização Org2.

Para gerar uma transação, o Fabric disponibiliza o comando `invoke`, que chama uma função do contrato instanciado no canal. O comando recebe como argumento o endereço e o caminho dos certificados dos pares, o serviço de ordenação, o nome do canal, o nome do contrato instanciado e a mensagem para o contrato em formato JSON. Como exemplo, o comando abaixo emite uma transação na rede.

```
peer chaincode invoke -o <ordenador>
--tls --cafile <caminho do certificado> -C $CHANNEL_NAME
-n mycc --peerAddresses <endereço do par de org1>
--tlsRootCertFiles <caminho do certificado do par>
--peerAddresses <endereço do par de org2>
```

```
--tlsRootCertFiles <caminho do certificado do par>
-c <mensagem em JSON para o chaincode>
```

- ordenador: orderer0.example.com:7050
- caminho do certificado: /opt/gopath/src/github.com/hyperledger/fabric/peer /crypto/ordererOrganizations/example.com/orderers/orderer0.example.com /msp/tlscacerts/tlsca.example.com-cert.pem
- endereço do par de org1: peer0.org1.example.com:7051
- caminho do certificado do par: /opt/gopath/src/github.com/hyperledger/fabric/peer /crypto/peerOrganizations/org1.example.com/peers /peer0.org1.example.com/tls/ca.crt
- endereço do par de org2: peer0.org2.example.com:7051
- caminho do certificado do par: /opt/gopath/src/github.com/hyperledger/fabric /peer/crypto/peerOrganizations/org2.example.com/peers /peer0.org2.example.com/tls/ca.crt
- mensagem em JSON para o chaincode: ‘{"Args":["initInstructionTransaction", "transaction","issuer","instruction"]}’

O Fabric permite serviços de busca e pesquisa de histórico de transações através do comando `peer chaincode query`. O comando recebe como argumento o nome do canal, o nome do contrato instanciado e a mensagem para o contrato em formato JSON. Um exemplo:

```
peer chaincode query -C $CHANNEL_NAME -n mycc -c
<chaincode JSON message>
```

- mensagem em JSON para o chaincode: ‘{"Args":["getHistoryForTransaction" , "transaction"]}’

3.5. Considerações Finais, Perspectivas e Problemas em Aberto

A tecnologia de corrente de blocos é muito recente, pois tem apenas dez anos. O fato desta tecnologia prover uma camada de confiança distribuída faz com que ela seja considerada disruptiva e também a tecnologia mais importante depois da Internet. As criptomoedas já são um sucesso e existem predições que afirmam que continuará a crescer fortemente o volume de "dinheiro virtual"registrado em correntes de blocos. Esta seção aborda os principais desafios e problemas em aberto das correntes de blocos assim como as oportunidades de pesquisa na área.

3.5.1. Oportunidades e Aplicações da Tecnologia de Corrente de Blocos

Além da transferência de ativos utilizando criptomoedas, a tecnologia de corrente de blocos pode ser aplicada em diversas áreas. Praticamente em todas aplicações que requerem intermediários a corrente de blocos pode eliminar este intermediário provendo a camada de confiança distribuída. Também para fazer registros permanentes e imutáveis.

Papel da Corrente de Blocos em uma Economia Compartilhada - A economia do compartilhamento, ou compartilhada, é um modo de consumo onde bens e serviços não são de posse de um único usuário. Estes bens e serviços são temporariamente disponibilizados por outros indivíduos (terceiros), que recebem um incentivo financeiro para oferecer o serviço, através de uma plataforma que atua como intermediário entre o provedor do serviço e o consumidor. Os principais exemplos de empresas que utilizam este modelo são a Uber e a Airbnb, com outros exemplos incluindo: Cohealo, BlaBlaCar, JustPark, Skillshare, RelayRides e Landshare. Estas empresas conseguem seu lucro tomando uma parte dos ganhos dos usuários que provêm o serviço, ao fornecer uma plataforma capaz de conectar pessoas com interesses coincidentes.

A tecnologia de corrente de blocos permite prover confiança entre o provedor de serviço e o usuário do serviço sem necessidade de uma empresa intermediária. As empresas centralizadoras são assim eliminadas [Hawlitschek et al. 2018]. Além disso, contratos inteligentes permitem definir e impor regras para um acordo entre duas partes, possibilitando a fácil responsabilização de qualquer partícipe caso ocorra uma violação desse contrato. Plataformas de economia compartilhada implementadas com corrente de blocos são desse modo administradas coletivamente por todos seus usuários. Como é do interesse dos membros da plataforma que esta continue funcionando, há um incentivo para que os participantes atuem com honestidade.

Um exemplo de aplicação da corrente de blocos na economia compartilhada é a plataforma descentralizada de carona solidária Arcade City, que propõe um serviço similar ao Uber. Todo o processamento necessário para tratar do compartilhamento de caronas e outras funções é executado pela corrente de blocos. Como não existem intermediários, os custos das caronas podem ser reduzidos significativamente. Segurança é garantida usando um sistema de classificação no próprio aplicativo, garantindo que avaliações feitas por usuários encontrem-se sempre disponíveis e inalteradas graças a características da corrente de blocos.

Corrente de Blocos para Cidades Inteligentes - O uso de corrente de blocos em sistemas de Cidades Inteligentes possui restrições de segurança e privacidade que ainda são desafios [Khatoun and Zeadally 2016]. As correntes de dados públicas, como o Bitcoin, efetuam transações anônimas identificadas por chaves públicas. No entanto, a anonimidade não é absoluta, pois todas as transações são visíveis para todos os participantes da corrente de blocos e, assim, as atividades do usuário podem ser rastreadas. Combinando as informações dessas transações com alguns dados externos permite revelar a identidade real do usuário. Uma vez que a identidade do usuário é descoberta, todas suas ações serão rastreadas e dados confidencias, como padrões de compra e venda, serão vazados. Portanto, estes sistemas não garantem a privacidade dos dados dos usuários o que viola um dos princípios da segurança da informação [Talari et al. 2017].

O Uso de Corrente de Blocos na Área de Saúde - A alta produção e transferência de dados na área da saúde podem ser beneficiadas pelo uso de corrente de blocos. Sistemas baseados em corrente de blocos permite que diferentes serviços de saúde possam compartilhar e armazenar dados e registros médicos em uma rede permissionada [Azaria et al. 2016]. Enquanto o uso de contratos inteligentes permite que pacientes controlem o acesso e a transferência de seus dados privados, a cópia da corrente de blocos em diferentes locais e a auditabilidade atendem à demanda de acesso de diferentes interessados ao mesmo documento.

A corrente de blocos também possui aplicações na rastreabilidade de dados na área da saúde. Sistemas baseados em corrente de blocos permitem o estabelecimento de uma rede privada e segura entre serviços de saúde para armazenar prescrições de remédios [Zhang et al. 2018]. As mudanças no estado ou posição do medicamento, assim como a transferência de propriedade são registradas no livro-razão distribuído, formando um registro histórico de dados do remédio desde a origem. A auditabilidade permite o acesso das entidades credenciadas às informações, facilitando a identificação de medicamentos falsos ou desviados.

3.5.2. Atividades em Organismos de Normalização

A aplicabilidade da tecnologia de corrente de blocos tem se demonstrado enorme. Assim, é essencial o pronto estabelecimento de normas internacionais para prover diretrizes relacionadas à uma nova tecnologia que propiciem um maior envolvimento e mais investimentos das indústrias. Em relação aos tópicos abordados neste capítulo, diferentes organismos de normalização criaram grupos de trabalho com objetivos variados.

A organização internacional de normalização (*International Organization for Standartization* - ISO) criou um comitê técnico que visa normalizar a tecnologia de corrente de blocos e a tecnologia de livro-razão distribuído [ISO/TC 307 2016]. O comitê possui projetos para formalizar os riscos de segurança, ameaças e vulnerabilidades da tecnologia, além de normalizar a arquitetura de referência, taxonomia, contratos inteligentes e a proteção de privacidade e de informações pessoais. A *Internet Research Task Force* (IRTF) criou o grupo de pesquisa da infraestrutura descentralizada da Internet (*Decentralized Internet Infrastructure Research Group* - DINRG) que estuda os serviços de infraestrutura beneficiados pela descentralização [IRTF 2017]. A *World Wide Web Consortium* (W3C) criou o grupo da comunidade de corrente de blocos (*Blockchain Community Group*) que tem como objetivo normalizar os formatos das mensagens em sistemas baseados em corrente de blocos [W3C 2016]. A comunidade usa o formato de mensagem definido na ISO 20022 como base para as normas. O grupo também busca definir diretrizes para o uso de armazenamento. A união internacional de telecomunicações (*International Telecommunications Union* - ITU), através do grupo de foco na aplicação da tecnologia de livro-razão distribuído (*Focus Group on Applications of Distributed Ledger Technology* - ITU-T FG DLT), pretende normalizar os serviços interoperáveis baseados na tecnologia de livro-razão distribuído [ITU-T FG DLT 2017]. O grupo de trabalho de corrente de blocos da Europa (*Europe Blockchain Working Group*) da associação internacional de segurança para comunicação institucional do comércio (*International Securities Association Trade Communication* - ISITC) discute a adoção da tecnologia de livro-razão distribuído pela indústria de serviços financeiros [Radford 2016].

3.5.3. Desafios e Problemas em Aberto

Os desafios em corrente de blocos são enormes: escalabilidade, vazão de transações, segurança e *software*, algoritmos criptográficos eficientes, entre outros.

Escalabilidade e Vazão de Transações - Todos os protocolos de consenso e tipos de corrente de blocos apresentados possuem limitações em termos de taxa de transferência, latência de transação ou quantidade de nós [Popov 2017]. Os sistemas baseados em correntes de blocos ainda são incapazes de atingir o desempenho dos atuais sistemas centralizados de transferência de ativos. Enquanto as plataformas do PayPal e Visa processam aproximadamente 2000 transações por segundo em média e até 56.000 transações em pico [Visa 2019, PayPal 2019] com tempos de resposta da ordem de segundos [BitcoinWiki 2019], o Ethereum processa um máximo de 20 transações por segundo e o Bitcoin atinge uma vazão de apenas 7 transações por segundo. Ainda, o processo de mineração da prova de trabalho no Bitcoin gera gastos insustentáveis de energia sem retorno proporcional, atingindo em média 50 TW consumidos por ano [Digiconomist 2019]. Algumas correntes de blocos permissionadas atingem taxas da ordem de milhares de transações por segundo, porém reduzem o número de participantes possíveis [Schwartz et al. 2014, Buchman 2016, Kiayias et al. 2017].

Outro desafio de correntes de blocos é a eficiência no armazenamento e na busca de transações, pois, para prover segurança descentralizada, é necessário que todos os participantes do consenso processem todas as transações da rede e armazenem todo o histórico de transações. A verificação de transações é fundamental para garantir segurança em um ambiente sem confiança entre os pares como o de corrente de blocos. No entanto, verificar a existência ou corretude de uma transação pode envolver uma busca custosa em uma lista crescente de todas as transações presentes em todos os blocos da rede. Além disso, a quantidade de dados armazenados em cada participante da corrente de blocos cresce constantemente, uma vez que nenhuma transação ou bloco jamais é descartado. As principais implementações de correntes de blocos implementam estruturas auxiliares chamadas de árvores de Merkle para contornar o problema[29]. O trilema entre descentralização, segurança e escalabilidade dificulta que os sistemas baseados em correntes de blocos atinjam o desempenho de sistemas atuais e que atendam às necessidades do futuro.

Revogação de Chaves criptográficas - Um dos principais desafios de sistemas baseados em correntes de blocos é a perda de chaves privadas, pois todas as transferências de ativos na rede são realizadas através de transações assinadas. A ausência de uma autoridade central confiável que armazene as identidades dos participantes da rede dificulta a revogação de chaves perdidas ou roubadas e consiste em um risco constante de perda de ativos. Em especial, é impossível revogar uma chave perdida em correntes de blocos públicas, nas quais o único identificador de um usuário é um endereço baseado na sua chave pública correspondente. Estima-se que mais de 30% dos *bitcoins* minerados estão inutilizados devido à perda de chaves, totalizando uma perda média de mais de 1000 *bitcoins* ou 8 milhões de dólares por dia [Chainalysis 2019].

Gerenciadores de chaves de correntes de blocos públicas implementam mecanismos de múltiplas assinaturas (*multisignature* ou *multisig*) para prevenir a perda de ativos

[29] A estrutura e fundamentos de árvores de Merkle são apresentados no Apêndice B.

decorrente da perda de uma chave privada. O mecanismo *multisignature* cria 3 pares de chaves assimétricas e exige a assinatura de pelo menos duas para emitir uma transação. As chaves podem ser armazenadas em locais diferentes e, possivelmente, controladas por entidades diferentes. Caso uma chave seja perdida, é possível assinar com as duas restantes. Nenhuma chave pode emitir uma transação individualmente. O mecanismo permite tolerar perdas de até uma chave ou repartir a posse de um ativo, pois, caso as chaves pertençam a entidades diferentes, pelo menos duas entidades devem concordar para emitir uma transação. Caso uma chave seja perdida no mecanismo de assinaturas múltiplas, basta criar um novo endereço e emitir uma transação assinada transferindo o ativo com as duas chaves restantes. Sistemas baseados em correntes de blocos permissionadas podem implementar mecanismos baseados em identidades pessoais para criar listas locais de revogação de chaves que devem ser alteradas através de consenso [Androulaki et al. 2018].

Segurança da Corrente de Blocos - Falhas de segurança em *software* de corrente de blocos podem causar um desastre. Existe uma enorme carência de profissionais que programem de forma segura e de ferramentas que consigam analisar, validar e testar a segurança de um *software* de corrente de blocos. Existem mais de 700 criptomoedas, no entanto, não existe nenhuma garantia que os softwares usando nas criptomoedas é seguro. Até mesmo, a maioria das implementações dos algoritmos de consenso não passaram por algum tipo de prova. Uma falha em um contrato inteligente pode ser catastrófica. Um exemplo foi a falha da chamada recursiva (*recursive call bug*) no acesso ao objeto (*data access object* - DAO) que causou o roubo de mais de 50 milhões de dólares no Ethereum[30].

No consenso com prova de trabalho (*Proof of Work* – PoW) usado no Bitcoin e Ethereum existe o conhecido ataque de 51% ou ataque de maioria, que se refere quando um atacante ou grupo de atacantes possui mais de 50% do poder computacional da rede, porque, neste caso, os atacantes podem fazer gasto duplo. Embora um ataque de 51% nunca tenha sido executado com sucesso no Bitcoin[31], no entanto, ele foi demonstrado que funciona em moedas alternativas do bitcoin (*altcoin*)[32] [33].

Ainda em relação ao mecanismo de consenso por prova de trabalho, Eyal and Sirer da Universidade de Cornell apresentaram em 2013 o ataque de mineração egoísta (*selfish mining*) [Eyal and Sirer 2013]. Este ataque se aplica ao consenso de prova de trabalho quando mineradores em conluio procuram aumentar seus lucros para receberem mais incentivos. A ideia chave deste ataque é que um conjunto de mineradores em conluio achem o resultado do desafio para formar um bloco, mas mantém a o resultado em segredo, forçando uma bifurcação de forma intencional. O conjunto de mineradores em conluio continuam minerando o ramo da bifurcação que eles forçaram sem divulgar novos blocos, enquanto mineradores honestos vão minerar o outro ramo. Caso os mineradores

[30]https://medium.com/@MyPaoG/explaining-the-dao-exploit-for-beginners-in-solidity-80ee84f0d470

[31]O conjunto de mineradores "Ghash.io"ultrapassou 50% do poder computacional da rede bitcoin em julho de 2014. Este conjunto de mineradores se comprometeu a reduzir o poder computacional para um valor sempre menor que 40%.

[32]A moeda Bitcoin Gold, na época a 26^a maior moeda, sofreu um ataque de 51% em maio de 2018. Os atacantes efetuaram gasto duplo por diversos dias e roubaram mais de US$18 milhões em Bitcoin Gold.

[33]As correntes de blocos Krypton e Shift, baseadas em Ethereum, sofreram ataques de 51% em agosto de 2016.

em conluio, mais de 25% da capacidade total, resolvam mais desafios, e não divulguem o resultado, eles passam a ter mais vantagem para conseguir obter o ramo mais longo. Quando o ramo minerado de forma pública pelos mineradores honestos se aproxima em tamanho do ramo minerado pelos malfeitores em conluio, os mineradores egoístas divulgam os blocos. Os mineradores honestos perderão dinheiro porque o ramo que eles mineraram não vai vingar.

Assiste-se hoje uma colaboração das indústrias sem precedentes mesmo entre empresas que eram concorrentes. O interesse na tecnologia de corrente de blocos cresce exponencialmente e o profissionais especialistas desta área estão muito requisitados e estão entre os mais bem pagos. Muitos desafios ainda persistem e muitos avanços são esperados para os próximos anos. Com certeza é uma área com enormes perspectivas e oportunidades.

A corrente de blocos pode resolver problemas em diversos setores da sociedade como economia, saúde, transporte, educação, entre outros. A tecnologia de corrente de blocos é uma poderosa ferramenta que oferece muitas possibilidades de atacar problemas de injustiça, desigualdade, carência, entre outras. No entanto, as soluções para problemas humanos na maioria das vezes não é técnica, mas sim humanas. A corrente de blocos não corrige seres humanos.

A. Criptografia Assimétrica, Assinatura e Função *Hash*

Este apêndice apresenta alguns conceitos básicos de criptografia que são usados pelas correntes de blocos.

Criptografia Assimétrica - A criptografia assimétrica consiste em um sistema criptográfico baseado em um par de chaves assimétricas: uma chave pública e uma chave privada. Enquanto a chave pública pode ser amplamente disseminada, a chave privada deve ser mantida em segredo. Assim, a chave pública é usada para criptografar uma mensagem que apenas quem possui a chave privada, par desta chave pública, é capaz de descriptografar. A criptografia RSA (Rivest-Shamir-Adleman) e a criptografia de curvas elípticas são as principais tecnologias de chaves assimétricas. As duas tecnologias podem prover a segurança necessária a qualquer sistema. No entanto, para um mesmo nível de segurança, mas as curvas elípticas requerem chaves de menor tamanho.

Assinatura Digital - A operação de assinatura criptográfica é quando a originador de uma mensagem a encripta com sua chave privada. Apenas a chave pública, par desta chave privada, consegue decriptar a mensagem. Assim, ao decriptar uma mensagem tem-se a garantia da autenticidade da origem da mensagem uma vez que a chave privada pertence apenas a originador da mensagem. O mecanismo de assinatura digital deve prover as propriedades básicas de autenticidade, não repúdio e integridade. A propriedade de autenticidade deve permitir a confirmação da autenticidade de uma mensagem, ou seja, que somente o signatário deve ser capaz de gerar sua assinatura digital para aquela mensagem. O não repúdio deve garantir que o signatário de uma mensagem assinada digitalmente não possa negar a autoria da assinatura. Por fim, é necessário a garantia de integridade: para sustentar as características anteriores de autenticidade e de não repúdio, a mensagem não pode ser adulterada.

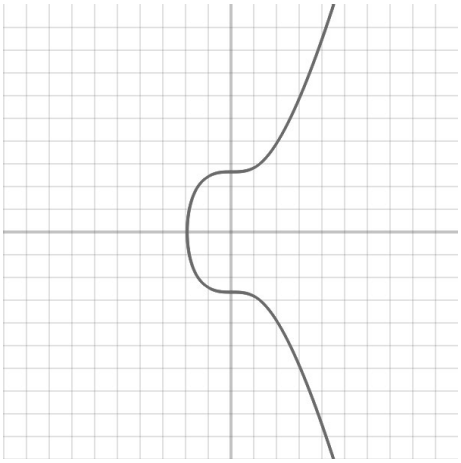


Figura 3.18: Curva Elíptica secp256k1 usada na moeda eletrônica Bitcoin.

A utilização de curvas elípticas em criptografia assimétrica foi sugerida por Neal Koblitz e Victor S. Miller em 1985. Uma curva elíptica é o locus dos pontos do plano cujas coordenadas satisfazem a equação cúbica $y^2 = x^3 + ax + b$ junto com um ponto na infinidade O. A Figura 3.18 ilustra a curva elíptica "secp256k1", usada na criptografia assimétrica do Bitcoin e definida nas normas para criptografia eficiente (*Standards for Efficient.00Cryptography* - SEC)[34].

O algoritmo de assinatura digital por curvas elípticas (*Elliptic Curve Digital Signature Algorithm - ECDSA* é uma variante do tradicional algoritmo de assinatura digital (*Digital Signature Algorithm* - DSA). O algoritmo baseado em curvas elípticas implementa uma modificação da norma de assinatura digital (*Digital Signature Standard* - DSS) que executa operações sobre pontos de curvas elípticas, ao invés das operações de exponenciação usadas no DSS. A maior vantagem do ECDSA sobre o DSA é que o ECDSA requer tamanhos bem menores de chave para propiciar a mesma segurança[35].

Função Resumo (*Hash*) - Uma função resumo, ou função *hash*, é uma função que mapeia dados de comprimento variável em dados de comprimento fixo. Uma função *hash* criptográfica é uma transformação matemática que, a partir de uma mensagem de comprimento arbitrário ou uma sequência de bits de tamanho arbitrário, obtém uma sequência curta e com tamanho fixo de bits. A função *hash* é descrita por $H(m) = h$, onde m é a mensagem em claro, $H()$ é a função *hash* e h é o valor *hash* resultante. A função *hash* deve ter as seguintes propriedades:

- a função *hash*, $H(m)$, pode ser aplicada a uma mensagem, m, de qualquer tamanho;
- a função *hash*, $H(m)$, gera uma saída h de tamanho fixo (por exemplo, o tamanho é 256 bits se $H(m)$ for o algoritmo SHA256);
- a função *hash* é simples, ou seja, é computacionalmente eficiente calcular $H(m)$

[34]Certicom Research, http://www.secg.org/sec2-v2.pdf

[35]Uma chave de criptografia de curvas elípticas (*Eliptic Curve Criptography* -ECC) de 163 bits corresponde à mesma garantia de segurança de uma chave RSA (*Rivest-Shamir-Adleman*) de 3.072 bits e uma chave de criptografia simétrica AES (*Advanced Encription Standard*) de 128 bits.

para qualquer m (a computação de $H(m)$ utiliza operações lógicas e evita as multiplicações e exponenciações usadas em criptografia assimétrica);

- a função *hash* é unidirecional ou não-invertível, isto é, para um h qualquer é computacionalmente impossível achar m tal que $H(m) = h$;

- a função *hash* é livre de colisões, ou seja, para uma mensagem, m, é computacionalmente impossível achar uma mensagem diferente, m', tal que $H(m) = H(m')$. Deve ser ressaltado que embora possam existir muitas mensagens que geram o mesmo *hash*, a função *hash* criptográfica deve ser projetada para dificultar ao máximo a colisão de *hashes*;

- é computacionalmente impossível achar um par de mensagens (m, m') tal que $H(m) = H(m')$.

A aplicação principal da função *hash* é a garantia de integridade de uma mensagem. Em geral, o emissor de uma mensagem calcula o seu *hash* e o insere no final da mensagem. O receptor recalcula o *hash* da mensagem recebida e compara o resultado obtido com o *hash* recebido. Se os *hashes* forem diferentes, há garantia de que a mensagem recebida é diferente da mensagem enviada. Se os *hashes* forem iguais, há uma probabilidade muito alta da mensagem recebida ser igual a mensagem enviada. Assim, qualquer alteração na mensagem pode ser detectada pelo recálculo e comparação do *hash*, o que garante a integridade da mensagem.

Existem diferentes funções *hash*, mas o Bitcoin e maioria das moedas criptográficas usam a função *hash* SHA256, que resulta em 256 bits. O algoritmo seguro de *hash* (*Secure Hash Algorithm* - SHA) foi desenvolvido pela Agência de Segurança Nacional (*National Security Agency* - NSA) dos EUA.

Assinatura da Mensagem e Assinatura do *Hash* da Mensagem - Uma mensagem criptografada por uma chave privada só pode ser descriptografada pela chave pública correspondente à chave privada que criptografou a mensagem. Esta operação garante o sigilo da mensagem, uma vez que apenas o destino pode descriptografar seu conteúdo, e a autenticidade do emissor, pois só ele possui a chave privada que foi usada na criptografia, e qualquer um que possua sua chave pública correspondente pode verificar a mensagem. Porém, a mensagem pode ter sido adulterada no caminho. Por outro lado, se o emissor computa o *hash* da mensagem, criptografa o *hash* com sua chave privada e envia o *hash* criptografado para o destinatário junto da mensagem, esta operação garante tanto a integridade da mensagem quanto autenticidade do emissor. A mensagem vai em claro e, portanto, não garante o sigilo, mas o destinatário garante ao mesmo tempo que a mensagem chegou íntegra e que o emissor é autêntico. Para isso, basta recalcular o *hash* a partir da mensagem, descriptografar o resultado com a chave pública do emissor e compará-lo com o *hash* assinado contido na mensagem.Este procedimento é muito usado quando não se requer o sigilo, pois o tempo de processamento para encriptar o *hash* é muito menor do que o necessário para encriptar uma mensagem inteira.

B. Árvores de Merkle e Verificação Simples de Pagamento

As árvores de Merkle são a principal estrutura auxiliar utilizada em correntes de blocos para verificar a integridade e não-repúdio de uma transação de maneira eficiente [Nakamoto 2008]. Nomeadas em homenagem a Ralph Merkle, que patenteou o conceito em 1979, as árvores de Merkle são estruturas de dados em forma de árvore na qual cada nó não-folha, denominado de "nó-galho,"é o resultado de um *hash* de seus respectivos nós filhos. Cada nó-folha da árvore é o resultado de um *hash* de um conjunto de dados que, no caso da corrente de blocos, representam as transações da rede. A Figura 3.19 ilustra a estrutura de uma corrente de blocos que utiliza uma árvore de Merkle binária para armazenar transações. A estrutura da árvore pode ser construída calculando o *hash* de cada transação T para criar o nível mais baixo da árvore, e, posteriormente, utilizando os *hashes* de cada nível para construir o próximo nível, até chegar à raiz da árvore. A complexidade de construção de uma árvore de Merkle de um bloco é $O(n.log_2(n)$ onde n é o número de transações contidas no bloco. A árvore binária é o tipo de árvore de Merkle mais utilizado em correntes de blocos, porém algumas implementações utilizam tipos diferentes de árvore [Wood 2014].

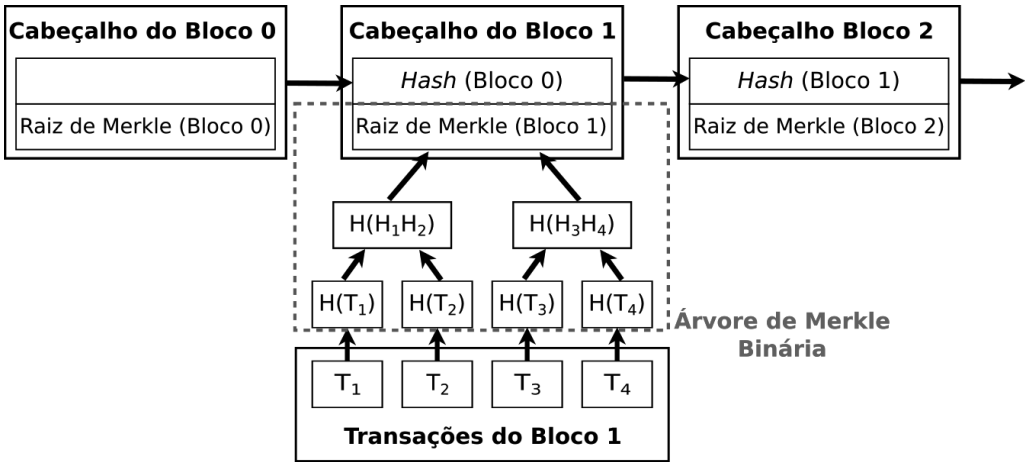


Figura 3.19: Estrutura de uma árvore de Merkle binária utilizada em correntes de blocos. O uso de árvores de Merkle permite armazenar apenas a raiz da árvore sem prejuízo à verificação das transações.

O uso de árvores Merkle permite obter uma prova de Merkle (*Merkle proof*), que verifica a presença e a corretude de uma transação em um bloco de maneira eficiente. Para verificar uma transação, basta fornecer os *hashes* necessários para reconstruir o caminho da árvore correspondente à transação verificada. A reconstrução de um caminho e, logo, a verificação de uma transação, tem complexidade $O(n)$. Assim, é possível verificar transações rapidamente mesmo em meio a milhares de transações, e não é necessário armazenar o bloco inteiro para verificar uma transação. Além disso, o uso de *hashes* diminui o tráfego na rede, pois não é necessário transferir a transação ou bloco completos. Isto permite criar "nós leves"e um mecanismo simplificado de verificação, pois, para verificar uma transação, basta armazenar o cabeçalho do bloco e solicitar a nós que possuem todas as transações os *hashes* do caminho até a transação. As provas de Merkle são a base do mecanismo de verificação simples de pagamento (*Simple Payment Verification* - SPV)

proposto por Nakamoto e utilizado na maioria das implementações de corrente de blocos atuais [Nakamoto 2008, Wood 2014].

Referências

[Alchieri et al. 2008] Alchieri, E. A., Bessani, A. N., Silva Fraga, J., and Greve, F. (2008). Byzantine consensus with unknown participants. In *OPODIS '08: Proceedings of the 12th International Conference on Principles of Distributed Systems*, pages 22–40, Berlin, Heidelberg. Springer-Verlag.

[Alchieri et al. 2018] Alchieri, E. A. P., Bessani, A., Greve, F., and d. S. Fraga, J. (2018). Knowledge connectivity requirements for solving byzantine consensus with unknown participants. *IEEE Transactions on Dependable and Secure Computing*, 15(2):246–259.

[Ali et al. 2016] Ali, M., Nelson, J. C., Shea, R., and Freedman, M. J. (2016). Blockstack: A global naming and storage system secured by blockchains. In *USENIX Annual Technical Conference*, pages 181–194.

[Alvarenga et al. 2018] Alvarenga, I. D., Rebello, G. A. F., and Duarte, O. C. M. B. (2018). Securing Management, Configuration, and Migration of Virtual Network Functions Using Blockchain. In *IEEE/IFIP NOMS*.

[Androulaki et al. 2018] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger Fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM.

[Angelis et al. 2018] Angelis, S. D., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V. (2018). PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain. In *Italian Conference on Cyber Security (06/02/18)*.

[Aspnes 2003] Aspnes, J. (2003). Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2-3):165–175.

[Attiya and Welch 2004] Attiya, H. and Welch, J. (2004). *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, 2nd edition.

[Azaria et al. 2016] Azaria, A., Ekblaw, A., Vieira, T., and Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In *Open and Big Data (OBD), International Conference on*, pages 25–30. IEEE.

[Back 2002] Back, A. (2002). Hashcash - a denial of service counter-measure. `http://www.hashcash.org/papers/hashcash.pdf`. Acessado em 12 de maio de 2019.

[Bano et al. 2017] Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. (2017). SoK: Consensus in the age of blockchains. Technical report, University College London, United Kingdom. https://arxiv.org/pdf/1711.03936.pdf.

[Bashir 2018] Bashir, I. (2018). *Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained*. Packt Publishing Ltd.

[Bessani et al. 2013] Bessani, A., Santos, M., Felix, J., Neves, N., and Correia, M. (2013). On the efficiency of durable state machine replication. In *Proc. of USENIX ATC 2013*.

[Bessani et al. 2014] Bessani, A., Sousa, J., and Alchieri, E. (2014). State machine replication for the masses with BFT-SMaRt. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[Bishop 2002] Bishop, M. (2002). *Computer Security: Art and Science*. Addison-Wesley.

[Biswas and Muthukkumarasamy 2016] Biswas, K. and Muthukkumarasamy, V. (2016). Securing smart cities using blockchain technology. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1392–1393.

[BitcoinWiki 2019] BitcoinWiki (2019). Bitcoin Scalability. `https://en.bitcoin.it/wiki/Scalability`. Acessado em 12 de maio de 2019.

[Bozic et al. 2017] Bozic, N., Pujolle, G., and Secci, S. (2017). Securing virtual machine orchestration with blockchains. In *1st Cyber Security in Networking Conference*.

[Brewer 2000] Brewer, E. A. (2000). Towards Robust Distributed Systems. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, New York, NY, USA. ACM.

[Buchman 2016] Buchman, E. (2016). *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph.

[Buterin 2014] Buterin, V. (2014). A Next Generation Smart Contract & Decentralized Application Platform. Technical report, white paper.

[Castro and Liskov 2002] Castro, M. and Liskov, B. (2002). Practical Byzantine Fault-Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

[Cavin et al. 2005] Cavin, D., Sasson, Y., and Schiper, A. (2005). Reaching agreement with unknown participants in mobile self-organized networks in spite of process crashes. Technical Report IC/2005/026, EPFL - LSR.

[Cavin et al. 2004] Cavin, D., Sasson, Y., and Schiper, A. (2004). Consensus with unknown participants or fundamental self-organization. In *Proc. of the 3rd Int. Conf. on Ad hoc Networks and Wireless - ADHOC-NOW 2004*, pages 135–148.

[Chainalysis 2019] Chainalysis (2019). Crypto Crime Report: Decoding increasingly sophisticated hacks, darknet markets, and scams. Technical report, Chainalysis Inc.

[Chandra and Toueg 1996] Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267.

[Chaum 1983] Chaum, D. (1983). Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, Hong-Ning.

[Chicarino et al. 2017] Chicarino, V. R. L., Jesus, E. F., de Albuquerque, C. V. N., and de A. Rocha, A. A. (2017). Uso de blockchain para privacidade e segurança em internet das coisas. In *Minicursos do SBSeg'2017*, pages 149–200.

[Christidis and Devetsikiotis 2016] Christidis, K. and Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. *IEEE Access*, 4:2292–2303.

[Chun et al. 2007] Chun, B.-G., Maniatis, P., Shenker, S., and Kubiatowicz, J. (2007). Attested append-only memory: Making adversaries stick to their word. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 189–204, New York, NY, USA. ACM.

[Correia et al. 2004] Correia, M., Neves, N. F., and Veríssimo, P. (2004). How to tolerate half less one Byzantine nodes in practical distributed systems. In *Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems*, pages 174–183.

[Dai 1998] Dai, W. (1998). B-money. `http://www.weidai.com/bmoney.txt`. Acessado em 12 de maio de 2019.

[Digiconomist 2019] Digiconomist (2019). Bitcoin Energy Consumption Index. `https://digiconomist.net/bitcoin-energy-consumption`. Acessado em 12 de maio de 2019.

[Dwork et al. 1988] Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

[Esposito et al. 2018] Esposito, C., De Santis, A., Tortora, G., Chang, H., and Choo, K. R. (2018). Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1):31–37.

[Eyal et al. 2016] Eyal, I., Gencer, A. E., Sirer, E. G., and Van Renesse, R. (2016). Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 45–59.

[Eyal and Sirer 2013] Eyal, I. and Sirer, E. G. (2013). Majority is not Enough: Bitcoin Mining is Vulnerable. Technical report, Department of Computer Science, Cornell University. https://arxiv.org/pdf/1311.0243.pdf.

[Eyal and Sirer 2018] Eyal, I. and Sirer, E. G. (2018). Majority is Not Enough: Bitcoin Mining is Vulnerable. *Commun. ACM*, 61(7):95–102.

[Finney 2005] Finney, H. (2005). RPOW: Reusable Proofs of Work. `http://nakamotoinstitute.org/finney/rpow/theory.html`. Acessado em 12 de maio de 2019.

[Fischer et al. 1985] Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374–382.

[Frantz and Nowostawski 2016] Frantz, C. K. and Nowostawski, M. (2016). From institutions to code: Towards automated generation of smart contracts. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 210–215. IEEE.

[Greve et al. 2018] Greve, F., Sampaio, L., Abijaude, J., Coutinho, A. A. R., Brito, I. V. S., and Queiroz, S. (2018). Blockchain e a Revolução do Consenso sob Demanda. In *Minicursos do SBRC'2018*, volume 36.

[Greve and Tixeuil 2010] Greve, F. and Tixeuil, S. (2010). Conditions for the Solvability of Fault-Tolerant Consensus in Asynchronous Unknown Networks: Invited Paper. In *III ACM SIGACT-SIGOPS International Workshop on Reliability, Availability, and Security. Co-located with the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*.

[Greve and Tixeuil 2007] Greve, F. G. P. and Tixeuil, S. (2007). Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks. In *Proc. of the Int. Conf. on Dependable Systems and Networks - DSN 2007*, pages 82–91.

[Hadzilacos and Toueg 1994] Hadzilacos, V. and Toueg, S. (1994). A Modular Approach to the Specification and Implementation of Fault-Tolerant Broadcasts. Technical report, Department of Computer Science, Cornell University, New York - USA.

[Hawlitschek et al. 2018] Hawlitschek, F., Notheisen, B., and Teubner, T. (2018). The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy. *Electronic commerce research and applications*, 29:50–63.

[Howard 2014] Howard, H. (2014). ARC: Analysis of Raft Consensus. Technical Report UCAM-CL-TR-857, University of Cambridge, Computer Laboratory.

[Huh et al. 2017] Huh, S., Cho, S., and Kim, S. (2017). Managing IoT Devices Using Blockchain Platform. In *Advanced Communication Technology (ICACT), 2017 19th International Conference on*, pages 464–467. IEEE.

[IRTF 2017] IRTF (2017). Decentralized Internet Infrastructure Research Group. `https://trac.ietf.org/trac/irtf/wiki/blockchain-federation`. Acessado em 9 abril de 2019.

[ISO/TC 307 2016] ISO/TC 307 (2016). Blockchain and distributed ledger technologies. `https://www.iso.org/committee/6266604.html`. Acessado em 12 de maio de 2019.

[ITU-T FG DLT 2017] ITU-T FG DLT (2017). Focus Group on Application of Distributed Ledger Technology. `https://itu.int/en/ITU-T/focusgroups/dlt/Pages/default.aspx`. Acessado em 12 de maio de 2019.

[Khatoun and Zeadally 2016] Khatoun, R. and Zeadally, S. (2016). Smart Cities: Concepts, Architectures, Research Opportunities. *Commun. ACM*, 59(8):46–57.

[Kiayias et al. 2017] Kiayias, A., Russell, A., David, B., and Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In Katz, J. and Shacham, H., editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham. Springer International Publishing.

[Kotla et al. 2009] Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. (2009). Zyzzyva: Speculative Byzantine fault tolerance. *ACM Transactions on Computer Systems*, 27(4):7:1–7:39.

[Kwon 2014] Kwon, J. (2014). Tendermint: Consensus without mining. *Draft v. 0.6, fall*.

[Lamport 1998] Lamport, L. (1998). The Part-Time Parliament. *ACM Transactions Computer Systems*, 16(2):133–169.

[Lamport 2001] Lamport, L. (2001). Paxos Made Simple. *ACM SIGACT News*, 32(4):18–25.

[Lamport et al. 2010] Lamport, L., Malkhi, D., and Zhou, L. (2010). Reconfiguring a state machine. *SIGACT News*, 41:63–73.

[Lamport et al. 1982] Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

[Lee et al. 2016] Lee, K., James, J. I., Ejeta, T. G., and Kim, H. J. (2016). Electronic voting service using block-chain. *Journal of Digital Forensics, Security and Law*, 11(2):8.

[Lynch 1996] Lynch, N. A. (1996). *Distributed Algorithms*. Morgan Kauffman.

[Machina Research 2016] Machina Research (2016). IoT global forecast anaysis 2015-2025. Technical report. `https://machinaresearch.com/login/?next=/report/iot-global-forecast-analysis-2015-25/`. Acessado em 31 de outubro de 2018.

[Mattos et al. 2018] Mattos, D. M. F. et al. (2018). Blockchain para Segurança em Redes Elétricas Inteligentes: Aplicações, Tendências e Desafios. In *Minicursos do SBSeg'2018*, pages 140–194.

[Mello et al. 2017] Mello, A. M., Marino, F. C. H., and Santos, R. R. (2017). Segurança de Aplicações Blockchain Além das Criptomoedas. In *Minicursos do SBSeg'2017*, pages 99–148.

[Miller et al. 2016] Miller, A., Xia, Y., Croman, K., Shi, E., and Song, D. (2016). The Honey Badger of BFT Protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42, New York, NY, USA. ACM.

[Mora et al. 2018] Mora, O. B., Rivera, R., Larios, V. M., Beltrán-Ramírez, J. R., Maciel, R., and Ochoa, A. (2018). A Use Case in Cybersecurity based in Blockchain to deal with the security and privacy of citizens and Smart Cities Cyberinfrastructures. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–4.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[Olson et al. 2018] Olson, K., Bowman, M., Mitchell, J., Amundson, S., Middleton, D., and Montgomery, C. (2018). Sawtooth: An Introduction. *The Linux Foundation, Jan*.

[Ongaro and Ousterhout 2014] Ongaro, D. and Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA. USENIX Association.

[Paul et al. 2014] Paul, G., Sarkar, P., and Mukherjee, S. (2014). Towards a More Democratic Mining in Bitcoins. In *International Conference on Information Systems Security*, pages 185–203. Springer International Publishing.

[PayPal 2019] PayPal (2019). Paypal Holdings, Inc. (PYPL) SEC Filing 10-K Annual report for the fiscal year ending Monday, December 31, 2018. `https://filings.last10k.com/sec-filings/1633917/000163391719000043/pypl201810-k.htm.pdf`. Acessado em 12 de maio de 2019.

[Pieroni et al. 2018] Pieroni, A., Scarpato, N., Di Nunzio, L., Fallucchi, F., and Raso, M. (2018). Smarter City: Smart Energy Grid Based on Blockchain Technology. *International Journal on Advanced Science, Engineering and Information Technology*, 8(1):298–306.

[Popov 2017] Popov, S. (2017). The Tangle. *cit. on*, page 131. `http://www.descryptions.com/Iota.pdf`. Acessado em 31 de outubro de 2018.

[Rabin and Ben-Or 1989] Rabin, T. and Ben-Or, M. (1989). Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 73–85, New York, NY, USA. ACM.

[Radford 2016] Radford, D. (2016). Europe Blockchain Working Group. Standard, The International Securities Association for Institutional Trade Communication. `https://isitc-europe.com/isitc-europe-blockchain-working-group`. Acessado em 12 de maio de 2019.

[Rahman et al. 2019] Rahman, M. A., Rashid, M. M., Hossain, M. S., Hassanain, E., Alhamid, M. F., and Guizani, M. (2019). Blockchain and IoT-Based Cognitive Edge Framework for Sharing Economy Services in a Smart City. *IEEE Access*, 7:18611–18621.

[Rebello 2019] Rebello, G. A. F. (2019). Encadeamento Seguro de Funções Virtuais de Rede Baseado em Correntes de Blocos. Projeto Final de Graduação, Universidade Federal do Rio de Janeiro, Brasil.

[Rebello et al. 2019a] Rebello, G. A. F., Alvarenga, I. D., Sanz, I. J., and Duarte, O. C. M. B. (2019a). BSec-NFVO: A Blockchain-based Security for Network Function Virtualization Orchestration. In *IEEE International Conference on Communications (ICC)*. A ser publicado.

[Rebello et al. 2019b] Rebello, G. A. F., Camilo, G. F., Silva, L. G. C., de Souza, L. A. C., Guimarães, L. C. B., and Duarte, O. C. M. B. (2019b). Segurança na Internet do Futuro: Provendo Confiança Distribuída através de Correntes de Blocos na Virtualização de Funções de Rede. In *Minicursos do SBRC'2019*.

[Rebello et al. 2019c] Rebello, G. A. F., Camilo, G. F., Silva, L. G. C., Guimarães, L. C. B., de Souza, L. A. C., Alvarenga, I. D., and Duarte, O. C. M. B. (2019c). Provendo uma Infraestrutura de Software Fatiada, Isolada e Segura de Funções Virtuais através da Tecnologia de Corrente de Blocos. Technical report, Grupo de Teleinformática e Automação (GTA/COPPE/UFRJ).

[Schneider 1990] Schneider, F. B. (1990). Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.*, 22(4):299–319.

[Schwartz et al. 2014] Schwartz, D., Youngs, N., and Britto, A. (2014). The Ripple Protocol Consensus Algorithm. *Ripple Labs Inc White Paper*, 5.

[Smolensk 2018] Smolensk, M. (2018). Lightstreams White Paper. `https://s3.amazonaws.com/lightstreams/lightstreams_whitepaper.pdf.Acessadoem12demaiode2019.`

[Sousa et al. 2018] Sousa, J., Bessani, A., and Vukolic, M. (2018). A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 51–58.

[Statista 2018] Statista (2018). Internet of things - number of connected devices worldwide. Technical report. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Acessado em 31 de outubro de 2018.

[Talari et al. 2017] Talari, S., Shafie-khah, M., Siano, P., Loia, V., Tommasetti, A., and Catalão, J. P. S. (2017). A Review of Smart Cities Based on the Internet of Things Concept. *Energies*, 10(4).

[Tikhomirov 2018] Tikhomirov, S. (2018). Ethereum: State of Knowledge and Research Perspectives. In *Foundations and Practice of Security*, pages 206–221. Springer International Publishing.

[Truong et al. 2018] Truong, N. B., Um, T., Zhou, B., and Lee, G. M. (2018). Strengthening the Blockchain-Based Internet of Value with Trust. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7.

[Vasin 2014] Vasin, P. (2014). Blackcoin's Proof-of-Stake Protocol v2. *URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf*, 71.

[Veronese et al. 2013] Veronese, G., Correia, M., Bessani, A., Chung, L., and Verissimo, P. (2013). Efficient Byzantine fault tolerance. *IEEE Transactions on Computers*, 62(1):16–30.

[Visa 2019] Visa (2019). About VisaNet. `https://usa.visa.com/about-visa/visanet.html`. Acessado em 12 de maio de 2019.

[W3C 2016] W3C (2016). Blockchain Community Group. `https://www.w3.org/community/blockchain`. Acessado em 12 de maio de 2019.

[Wilkinson et al. 2014] Wilkinson, S., Boshevski, T., Brandoff, J., and Buterin, V. (2014). Storj: a Peer-to-Peer Cloud Storage Network. `https://storj.io/storj2014.pdf`. Acessado em 12 de maio de 2019.

[Wood 2014] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger.

[Wüst and Gervais 2018] Wüst, K. and Gervais, A. (2018). Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE.

[Xie et al. 2019] Xie, J., Tang, H., Huang, T., Yu, F. R., Xie, R., Liu, J., and Liu, Y. (2019). A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. *IEEE Communications Surveys Tutorials*.

[Xu et al. 2017] Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., and Rimba, P. (2017). A Taxonomy of Blockchain-Based Systems for Architecture Design. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 243–252.

[Zhang et al. 2018] Zhang, P., Schmidt, D. C., White, J., and Lenz, G. (2018). Chapter One - Blockchain Technology Use Cases in Healthcare. In Raj, P. and Deka, G. C., editors, *Blockchain Technology: Platforms, Tools and Use Cases*, volume 111 of *Advances in Computers*, pages 1 – 41. Elsevier.

[Zheng et al. 2018] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., and Wang, H. (2018). Blockchain Challenges and Opportunities: A Survey. *International Journal of Web and Grid Services*, 14(4):352–375.