Capítulo 1

Segurança na Internet do Futuro: Provendo Confiança Distribuída através de Correntes de Blocos na Virtualização de Funções de Rede

Gabriel Antonio F. Rebello, Gustavo F. Camilo, Leonardo G. C. Silva, Lucas A. C. de Souza, Lucas C. B. Guimarães e Otto Carlos M. B. Duarte

Grupo de Teleinformática e Automação/PEE-COPPE/DEL-Poli
Universidade Federal do Rio de Janeiro - RJ, Brasil

Resumo

A Internet do Futuro, além de interconectar bilhões de dispositivos, deverá atender com agilidade a uma enorme diversidade de requisitos de serviços, alavancar novas aplicações e continuar sendo a maior infraestrutura para crescimento econômico e social das nações. Esta Internet de nova geração, baseada na tecnologia de Virtualização de Funções de Rede (Network Function Virtualization - NFV) e Redes Definidas por Software (Software-Defined Networking - SDN) será a essência das redes móveis de quinta geração (5G) e das cidades inteligentes (smart cities). Um dos desafios fundamentais de pesquisa é garantir o provisionamento seguro de serviços em um ambiente de nuvem onde múltiplos inquilinos e usuários sem confiança mútua compartilham recursos. O objetivo deste capítulo é mostrar de forma clara, objetiva e sucinta os fundamentos-chave de corrente de blocos (blockchain), e relacionar estes conceitos aos desafios de pesquisa em segurança nas redes de nova geração. Além disso, este capítulo dedica uma parte significativa à realização de uma atividade prática que mostra o desenvolvimento de duas correntes de blocos baseadas 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.

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

1.1. Introdução

A Internet do Futuro interconectará bilhões de dispositivos através de sistemas construídos a partir de milhares de serviços distribuídos. Aglomerados com um enorme número de máquinas físicas e virtuais manipularão grandes quantidades de dados gerados a partir de inúmeros sensores e atuadores da Internet das Coisas (*Internet of Things* - IoT), gerando um volume de informações jamais visto. A Internet será constituída de fatias de rede (*network slices*) adaptadas a cada tipo de aplicação, que serão criadas, alteradas e destruídas de maneira ágil e escalável para oferecer alta disponibilidade com baixo custo de operação. Esta Internet de nova geração, baseada na tecnologia de Virtualização de Funções de Rede (*Network Function Virtualization* - NFV) e Redes Definidas por Software (*Software-Defined Networking* - SDN), será a essência das redes móveis de quinta geração (5G) e das cidades inteligentes (*smart cities*).

Neste novo paradigma, um dos desafios fundamentais de pesquisa é garantir o provisionamento seguro de serviços em um ambiente de nuvem onde múltiplos inquilinos e usuários sem confiança mútua compartilham recursos. Uma forma de garantir a confiança de forma distribuída e permitir a análise forense para identificação dos responsáveis pelos problemas é através do uso da tecnologia de corrente de blocos (*blockchain*). A corrente de blocos, conhecida por suas aplicações em criptomoedas como o Bitcoin [Nakamoto 2008] e Ethereum [Wood 2014], 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. Pode-se prever um futuro no qual a computação e a Internet associadas à tecnologia de corrente de blocos permitirão uma computação planetária distribuída que executa aplicações e contratos inteligentes através de um consenso entre computadores sem nenhuma entidade intermediária. Esta confiança distribuída sem intermediários provida pela corrente de blocos permitirá múltiplas ofertas de funções virtualizadas de rede e seleção criteriosa entre diversos provedores de infraestrutura, produzindo uma concorrência sem precedentes na área de telecomunicações.

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 são ainda maiores, pois prevê-se que em 2025 o uso de dispositivos IoT gere até 3 trilhões de dólares em valor de mercado [Arnott et al. 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*), cidades inteligentes (*smart cities*) e redes elétricas inteligentes (*smart grids*).

Este capítulo aborda as tecnologias de virtualização de funções de rede, de redes definidas por software e de corrente de blocos para prover segurança à Internet do Futuro em um ambiente multiusuário e multi-inquilino. O minicurso foca na criação e gerenciamento distribuído de fatias de redes, de forma ágil, confiável e segura, através do encadeamento de funções de rede em um cenário de múltiplos centros de dados com diversos provedores de funções virtuais de rede. O minicurso é organizado em quatro partes principais: i) a tecnologia de virtualização de funções de redes; ii) a tecnologia de

corrente de blocos; iii) fatiamento seguro de redes através de corrente de blocos e iv) a elaboração prática de uma aplicação que se serve de uma corrente de blocos programada no Hyperledger Fabric [2]. A aplicação desenvolvida na parte prática segue as funcionalidades providas pelo sistema SINFONIA (*Secure vIrtual Network Function Orchestrator for Non-repudiation, Integrity, and Auditability*) [Rebello et al. 2018], um sistema desenvolvido pelo Grupo de Teleinformática e Automação (GTA) que usa corrente de blocos para garantir segurança na criação das redes. O protótipo desenvolvido também oferece funcionalidades de garantia de análise forense de configuração, gerenciamento e migração de máquinas virtuais [Alvarenga 2018].

O objetivo deste capítulo é mostrar de forma clara, objetiva e sucinta os fundamentos chave de corrente de blocos, e relacionar estes conceitos aos desafios de pesquisa em segurança nas redes de nova geração. O principal diferencial deste minicurso é a utilização da corrente de blocos em ambientes de rede virtualizados, analisando os impactos, provendo segurança e gerando oportunidades de pesquisa para o futuro.

1.2. Virtualização de Redes e Segurança

Esta seção apresenta os conceitos básicos da virtualização de funções de rede (*Network Function Virtualization* - NFV) e do encadeamento de funções de serviço (*Service Function Chaining* - SFC), discutindo os principais desafios de segurança em ambientes multi-inquilino e multiusuário característicos da virtualização.

1.2.1. Virtualização de Funções de Rede

A tecnologia de virtualização de funções de rede (*Network Function Virtualization* – NFV) é a aplicação de conceitos de virtualização e computação em nuvem para as telecomunicações [Medhat et al. 2017]. O NFV permite reduzir gastos e flexibilizar o gerenciamento e a operação das redes de comunicações através da substituição de recursos em *hardware* dedicado por funções virtualizadas em *software*, que são executadas em *hardware* de uso geral [Pattaranantakul et al. 2016]. Assim, os sistemas intermediários (*middleboxes*), como *firewalls*, sistemas de detecção e prevenção de intrusão, balanceadores de carga, roteadores, *proxies*, etc. que hoje são implementados em *hardware* específicos de um fabricante, serão substituídos por funções em *software* virtualizadas que podem ser providas por diferentes fornecedores [Sekar et al. 2012]. Dentre os principais benefícios da tecnologia NFV estão a redução dos gastos de capital (*CAPital EXpenditure* - CAPEX) e gastos operacionais (*OPerational EXpenditure* - OPEX) com equipamentos dedicados que requerem alocação de espaço físico, refrigeração adequada, gasto energético e formação de recursos humanos. Outra importante vantagem da tecnologia NFV é o menor tempo de chegada até o mercado (*time-to-market* - TTM) para uma função de rede desde a sua concepção e desenvolvimento até a implementação no domínio de um operador de rede [Mijumbi et al. 2016]. Estima-se reduzir o TTM, que hoje usualmente leva quatro ou cinco anos, para três a quatro meses.

Algumas definições são reforçadas para o melhor entendimento desta proposta. O orquestrador de nuvem é a entidade que cria a cadeia de funções de rede através da plataforma de virtualização utilizada no centro de dados. O administrador da nuvem é a

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

companhia ou a operadora responsável por um ou mais centros de dados, na qual a virtualização é realizada e que cada orquestrador implementa suas funções. Um inquilino é um cliente do centro de dados que usufrui de um serviço provido pelo encadeamento de funções de rede. É possível ainda orquestrar diferentes cadeias de funções que possuem uma mesma VNF em comum, ou seja, fatias de recursos de uma VNF podem ser compartilhadas entre inquilinos. Dessa forma, um ambiente de nuvem com infraestrutura de NFV é definido como um ambiente multi-inquilino, o que permite uma gama de novos ataques, como *side-channel*, inspeção de tráfego, esgotamento de recursos etc. Portanto, requisitos de segurança adicionais são necessários para prover o correto isolamento entre todos os inquilinos em um cenário NFV.

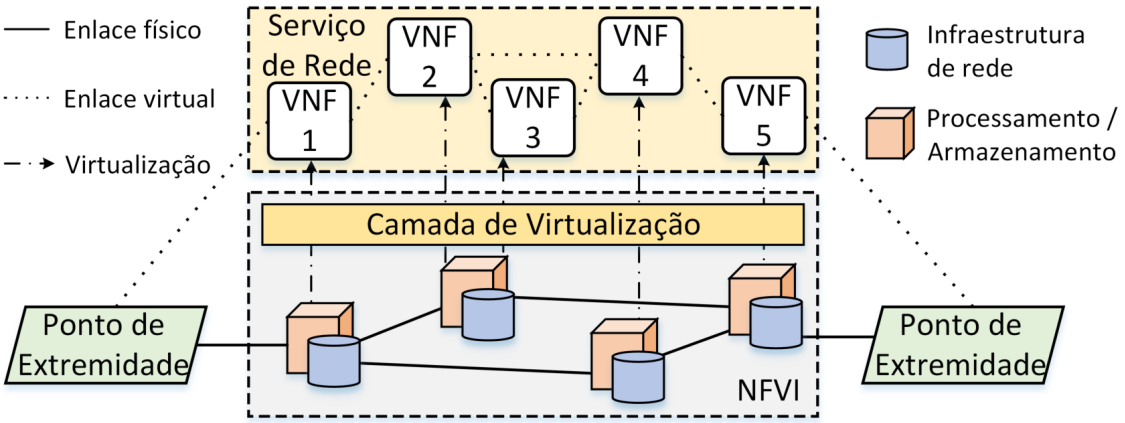

Figura 1.1. A infraestrutura de virtualização de funções de rede, que permite criar enlaces virtuais para serviços fim-a-fim através do encaminhamento de pacotes entre funções virtuais de rede.

A infraestrutura de virtualização de funções de rede (*NFV Infrastructure* – NFVI), proposta pelo *European Telecommunications Standards Institute* (ETSI) na principal arquitetura de gerência e orquestração das funções virtuais de rede (*Network Function Virtualization MANagement and Orchestration* – NFV-MANO) [ETSI 2014], tem como objetivo padronizar a composição de serviços fim-a-fim sob medida para cada aplicação. A NFVI fornece as abstrações de processamento, armazenamento e acesso à rede para que máquinas virtuais se comportem como funções virtuais de rede. A Figura 1.1 mostra uma simplificação de uma rede virtualizada através da infraestrutura de virtualização. Na arquitetura, o NFVI transforma os nós de processamento, os nós de armazenamento e a infraestrutura de rede de uma infraestrutura física em uma camada de virtualização que aloca os recursos necessários para cada função virtualizada. Na arquitetura do NFV-MANO, cada provedor de infraestrutura administra centros de dados com NFVIs capazes de realizar operações de gerenciamento e orquestração de funções virtualizadas de rede. As funções virtualizadas podem ser encadeadas através de enlaces virtuais para prover um serviço fim-a-fim entre duas extremidades da rede.

1.2.2. Encadeamento de Funções de Serviço

Nas redes da próxima geração, baseadas na tecnologia NFV [ETSI 2014], o encadeamento de funções de serviço (*Service Function Chaining* – SFC) [Halpern and Pignataro 2015] é o procedimento fundamental para fornecer controle e gerenciamento flexível

do tráfego de um serviço ou de uma aplicação. Uma função de serviço é uma função de rede virtualizada (*Virtual Network Function* - VNF) ou não virtualizada que compõe um serviço fim-a-fim. Por simplicidade, no cenário abordado deste trabalho, consideram-se funções de serviço como sinônimos de funções virtualizadas de rede. Alguns pesquisadores, no entanto, consideram uma função de serviço como uma composição de uma ou mais funções de rede encadeadas [Li and Chen 2015]. O encadeamento de funções de serviço no caminho da origem até o destino fornece, sob demanda, um serviço adaptado para cada aplicação ou usuário.

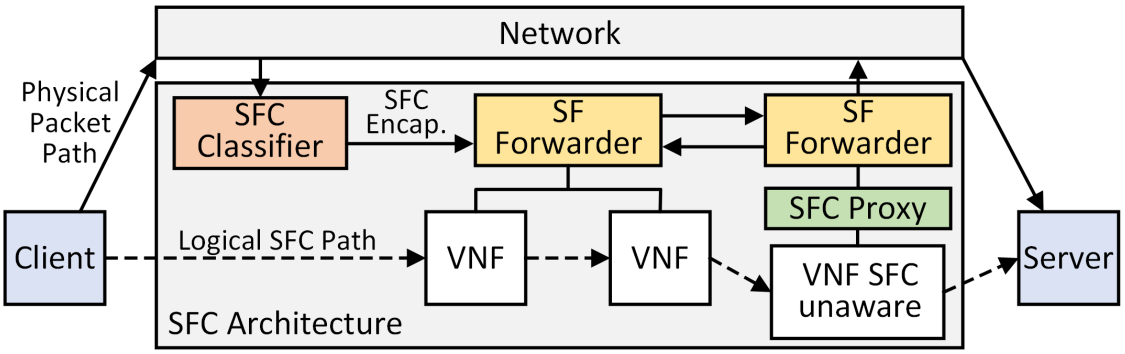

Figura 1.2. Arquitetura-padrão do encadeamento de funções de serviço [Halpern and Pignataro 2015]. O classificador SFC encapsula o tráfico ingressante e encaminhador de funções de serviço (SFF) encaminha o tráfego encapsulado para a cadeia correta baseado no cabeçalho SFC. Um proxy SFC realiza o encapsulamento e desencapsulamento para funções de rede sem suporte ao cabeçalho SFC.

A Figura 1.2 ilustra a arquitetura padrão do encadeamento de funções de serviço, proposta na RFC 7665 [Halpern and Pignataro 2015]. Considerando um ambiente multisserviço, configura-se um classificador com regras específicas para identificar e especificar a cadeia de VNFs que cada fluxo de serviço deve atravessar. A cadeia de funções de serviço é definida por uma sequência lógica e ordenada de VNFs, chamado caminho de função de serviço (*Service Function Path* - SFP) e cada cadeia possui um identificador. Assim, fluxos de diferentes serviços podem ser isolados e atravessar simultaneamente uma mesma VNF. As funções virtualizadas de rede também podem ser hospedadas em diferentes nós. Portanto, deve existir um encaminhador de função de serviço (*Service Function Forwarder* - SFF) em cada nó da infraestrutura de virtualização de funções de rede (NFVI) para fornecer enlaces virtuais para as VNFs hospedadas. Na implementação padrão de encadeamento de funções de serviço, o isolamento entre fluxos de serviços que atravessam a mesma VNF é realizado através de encapsulamento de pacotes. VNFs podem estar cientes ou inconscientes do encapsulamento SFC. VNFs que desconhecem o encapsulamento requerem um elemento precedente, o Proxy SFC, para desencapsular pacotes SFC e transformá-los em pacotes comuns da pilha TCP/IP. VNFs conscientes do SFC realizam o encapsulamento e desencapsulamento através de um módulo do *kernel* ou um comutador virtualizado compatível com o encapsulamento SFC.

1.2.3. Segurança em Ambientes de Funções Virtuais de Rede

Funções de rede podem ser estrategicamente posicionadas de acordo com sua função. Uma VNF de um sistema de detecção de intrusão (VNF IDS) é mais eficaz quando instanciada próximo de uma fonte de ataques ou em um centro de dados especializado em monitoramento de tráfego que realiza a correlação de informações e registros de detecção com outras VNF IDS. Uma VNF que realiza transcodificação de serviços de fluxo (*streaming*) deve estar localizada próximo à fonte do conteúdo para evitar perdas de dados na rede. Ainda, os requisitos de posicionamento de VNFs tornam-se pontos críticos quando não há um centro de dados da operadora na localização desejada, necessitando o uso de recursos virtuais de outra operadora mais próxima. Portanto, neste ambiente de nuvem flexível, uma cadeia de funções de rede pode ter componentes instanciados em domínios de operadoras diferentes. Em um cenário virtualizado, a comunicação entre pontos de extremidade é realizada através da conexão entre grandes centros de dados (*data centers*), que são capazes de prover, sob demanda, serviços fim-a-fim adaptados a cada aplicação através do encadeamento de funções de serviço (*Service Function Chaining* - SFC). Um cenário de encadeamento de funções para um serviço de rede é ilustrado na Figura 1.3. Neste cenário, uma cadeia de funções de rede é composta por VNFs hospedadas em diferentes domínios de operadoras de telecomunicações. Cada operadora possui um centro de dados, que contém um único orquestrador de VNFs e uma infraestrutura de virtualização baseada em máquinas de propósito geral que pode hospedar máquinas virtuais e VNFs.

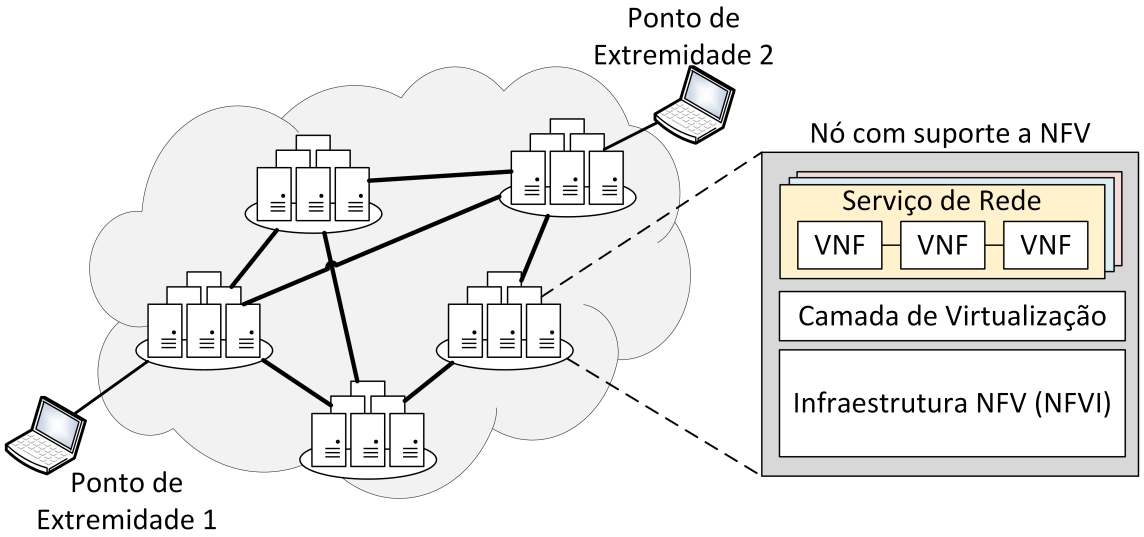

Figura 1.3. O cenário da virtualização de funções de rede. Uma conexão fim-a-fim entre dois pontos de extremidade atravessa conexões entre centros de dados que possuem infraestruturas de virtualização. Cada centro de dados possui uma infraestrutura de virtualização que gerencia e encadeia funções virtualizadas de rede para prover diferentes serviços fim-a-fim.

A virtualização de funções de rede e o encadeamento de funções de serviço provê a flexibilidade, a agilidade e o baixo custo desejado para as telecomunicações, mas traz novos desafios de segurança [Medhat et al. 2017]. Neste novo cenário, os inquilinos compartilham a mesma infraestrutura de nuvem, e cadeias podem envolver funções virtu-

alizadas instanciadas em domínios de operadoras concorrentes [Han et al. 2015, Mijumbi et al. 2016]. Desta forma, é necessário garantir que a cadeia de funções virtualizadas de rede seja construída de maneira confiável em um ambiente sem confiança entre os pares, sejam estes inquilinos ou domínios. O ambiente multi-inquilino e multi-domínio aumenta a possibilidade de ataques dentro da nuvem e dificulta a responsabilização ou uma possível indenização pelos provedores do serviço devido a um mau funcionamento. Além disso, os impactos de possíveis ataques tornam-se bem maiores, uma vez que ataques ao hospedeiro de funções de rede podem comprometer milhares de usuários simultaneamente [Bouras et al. 2017].

1.2.3.1. O modelo FCAPS e Terminologia

O modelo FCAPS (*Fault, Configuration, Administration, Performance and Security)* é o padrão de gestão de redes de telecomunicação [Raman 1998] em ambientes centralizados e de provedor único. O modelo provê padrões e boas práticas para assegurar propriedades fundamentais de segurança, como autenticidade, integridade e não-repúdio. No entanto, o cenário de virtualização de funções de rede é um ambiente distribuído que não possui confiança entre os pares, de forma que o FCAPS não é diretamente aplicável. Neste cenário, diversos provedores oferecem serviços distribuídos através de múltiplas nuvens, trazendo novos desafios, tais como: i) selecionar as métricas que garantem a corretude do serviço fim-a-fim de uma cadeia de funções de rede; ii) identificar o provedor de nuvem, dentre os participantes de uma cadeia de funções de rede, responsável por uma falha; e iii) estabelecer a proveniência, o impacto e o tempo que uma determinada falha permaneceu indetectada. Portanto, uma solução com o uso de corrente de blocos é apropriada para solucionar estes desafios.

1.2.3.2. Modelo de Atacante

Este trabalho considera um modelo de atacante de Dolev [Dolev and Yao 1983], no qual um atacante pode *ler*, *enviar* e *descartar* uma transação endereçada à corrente de blocos, ou qualquer pacote de rede. O atacante pode agir de maneira passiva, se conectando na rede e capturando toda troca de mensagens, ou de maneira ativa, injetando, repetindo, filtrando ou trocando informações. Os ataques podem atingir inquilinos, VNFs, a corrente de blocos ou a rede.

Ataques à corrente de blocos consistem em impedir que uma transação ou bloco legítimo seja incorporado à corrente de blocos. Para realizar este tipo de ataque, o atacante necessita controlar uma parcela significativa da rede para afetar o consenso. Esse tipo de ataque é mitigado pela tolerância a falhas do protocolo de consenso utilizado.

Ataques ao encadeamento de funções de rede consistem em modificar de forma maliciosa uma cadeia de funções de serviço. Este tipo de ataque é realizado por provedores que gerenciam as cadeias de funções de serviço e as funções virtualizadas de rede. Dentre os possíveis ataques estão o desvio de tráfego de uma cadeia de funções através de uma VNF maliciosa, a remoção ou modificação de uma VNF em uma cadeia, ou a sub-alocação de recursos para VNFs e enlaces virtuais.

Ataques a inquilinos ou VNFs consistem em personificar o alvo emitindo ou retransmitindo transações. Ataques de personificação são mitigados através do uso de chaves criptográficas assimétricas para assinar toda transação enviada à corrente de blocos. Toda transação é assinada por seu emissor e verificada pelos participantes do consenso. Este trabalho não trata casos de comprometimento de um inquilino ou VNF através da invasão de seu terminal ou roubo de suas chaves privadas. No entanto, a arquitetura proposta neste trabalho permite a auditoria das transações caso o atacante realize um ataque de personificação utilizando pares de chaves roubados. Outro trabalho relacionado propõe mitigar ataques a VNFs eliminando qualquer serviço de escuta ativo em uma VNF e utilizando seus terminais apenas em modo de leitura [Alvarenga et al. 2018]. Pares de chaves roubados ou perdidos podem ser substituídos para impedir maiores danos.

Ataques à rede consistem em isolar um ou mais inquilinos ou VNFs da rede, impedindo a emissão de transações e leitura do conteúdo da corrente de blocos. Essa categoria de ataque contempla ataques clássicos de rede, que podem ser mitigados através do estabelecimento de caminhos redundantes entre a corrente de blocos e as VNFs ou inquilinos. O foco deste trabalho não são esses ataques, e sim os ataques à corrente de blocos, aos inquilinos, às VNFs, e ao encadeamento de funções de rede.

Este capítulo foca no uso da tecnologia de corrente de blocos para mitigar possíveis vetores de ataques ao encadeamento de funções de rede, ataques aos inquilinos, ataques às VNF e ataques à própria corrente de blocos.

1.3. 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.

1.3.1. 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 subtrair o valor do ativo a ser transferido da origem e acrescentá-lo 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 1.4 ilustra o problema do gasto duplo. No exemplo, suponha que Alice deseja transferir moedas para

[3]Esta seção é baseada no projeto de fim de curso de Gabriel Antonio Fontes Rebello [Rebello 2019].

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.

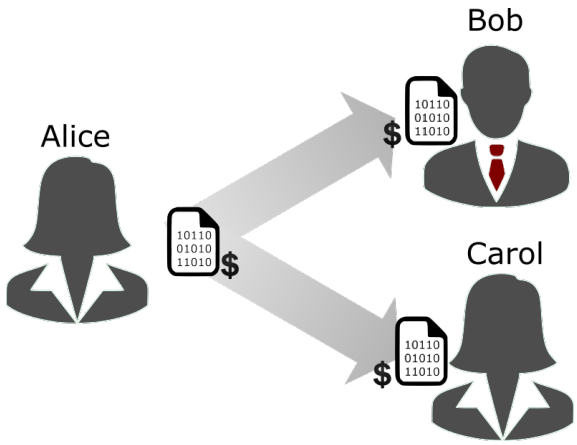

Figura 1.4. Exemplo do problema do gasto duplo no qual a mesma representação digital de uma moeda é replicada e gasta duas vezes.

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 atrasos para a transferência.

1.3.2. Criptomoedas e Corrente de Blocos

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*,

[4]Disponível em https://www.iana.org/

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. Finalmente, em 2008, Satoshi Nakamoto publicou "Bitcoin: A peer-to-peer electronic cash system," propondo e implementando pela primeira vez uma criptomoeda totalmente descentralizada, combinando a prova de trabalho com uma nova e revolucionária estrutura de dados organizada em blocos de transações: a corrente de blocos (*blockchain*) [Nakamoto 2008]. O mais impressionante é que a proposta de solução do problema utiliza tecnologias já conhecidas e dominadas como criptografia, consenso, redes de comunicação e incentivos.

1.3.2.1. 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 1.3.2.3, 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[6], para gravar, compartilhar e sincronizar transações em um sistema de controle

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

[6]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.

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. Assim, 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, utilizando o protocolo de consenso, um bloco que oculte uma de suas transações [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.

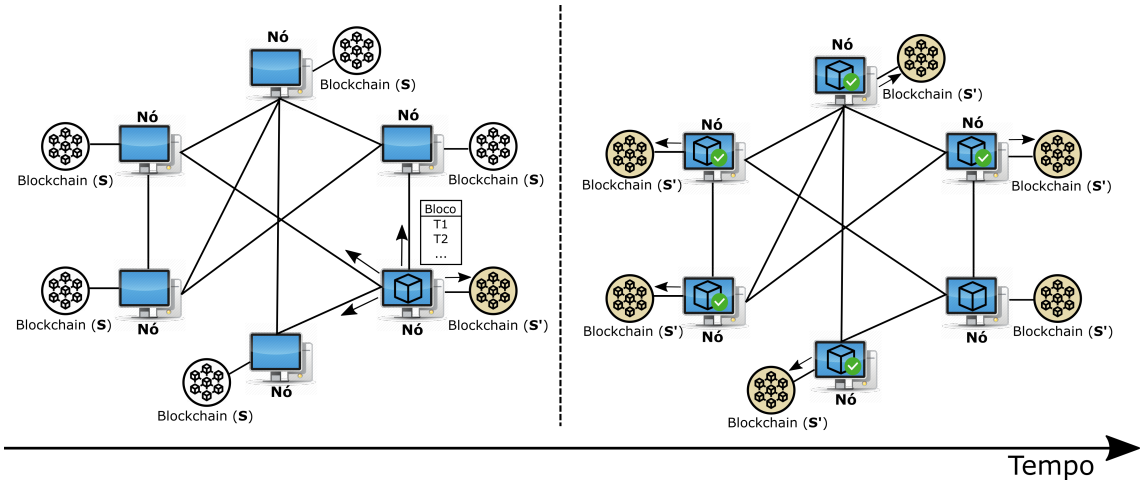

Figura 1.5. Atualização de uma corrente de blocos (*blockchain*) através de um protocolo de consenso genérico. Um nó minerador propõe um novo bloco em difusão e os demais nós mineradores verificam e adicionam independentemente o bloco proposto à corrente de blocos, incrementando o estado global S de maneira consistente.

A Figura 1.5 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 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 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 1.3.3.

A Figura 1.6 mostra o ciclo de vida 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. Por exemplo, para transferir 0,5 BTC a Bob, Alice cria uma transação contendo: i) uma entrada com o 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 de irá possuir após a transferência do valor desejado. A seguir, Alice utiliza um algoritmo criptográfico para assinar a transação, criando uma transação assinada, 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. A partir deste momento, 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".

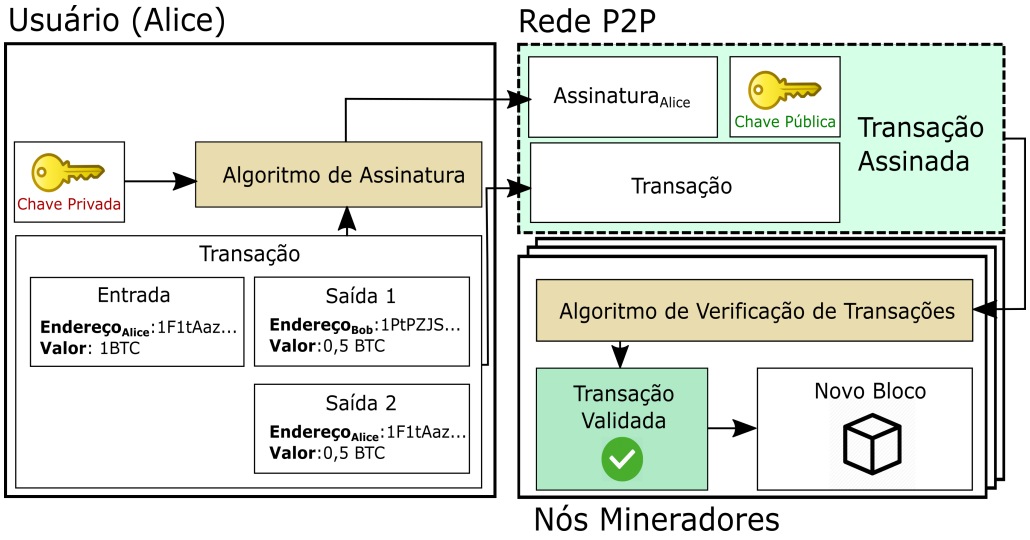

Figura 1.6. Ciclo de vida de uma transação em uma corrente de blocos. O usuário A difunde na rede par a par uma transação assinada que pode ser verificada e adicionada a um bloco por qualquer nó minerador.

1.3.2.2. Estruturas de Dados

A estrutura de dados da corrente de blocos proposta por Nakamoto como parte da criptomoeda Bitcoin [Nakamoto 2008] 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 1.7 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[7], 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 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 [Storj Labs 2018, 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 conteúdo de uma corrente de blocos, no entanto, 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 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

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

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.

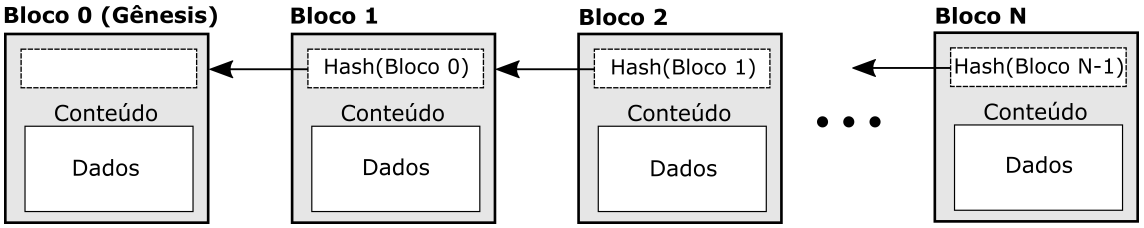

Figura 1.7. Estrutura de uma corrente de blocos na qual cada bloco é associado ao bloco seguinte e a integridade das transações de um bloco é garantida por uma uma função resumo (*hash*) criptográfica.

1.3.2.3. 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 correntes de bloco são acessíveis por todos os participante da rede. Em correntes de blocos públicas, como o Bitcoin [Nakamoto 2008] e o Ethereum [Wood 2014], 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.

1.3.2.4. Categorias de Corrente de Blocos

A utilização de correntes de blocos constrói uma máquina de estados replicada que garante a consistência do sistema e um estado global compartilhado entre todos os participantes da rede. No entanto, a corrente de blocos deve ser configurada para atender às demandas de cada aplicação através de decisões de projeto. Através da taxonomia proposta na literatura [Christidis and Devetsikiotis 2016, Zheng et al. 2018, Xu et al. 2017] e das principais aplicações de correntes de blocos [Nakamoto 2008, Wood 2014, Androulaki et al. 2018], é possível categorizar as correntes de blocos em três tipos: **correntes de blocos públicas**, **correntes de blocos federadas** e **correntes de blocos privadas**. Os tipos de correntes de blocos diferenciam-se nos seguintes aspectos:

- **Permissão de escrita.** Correntes de blocos públicas, ou **não-permissionadas**, não impõem restrições de permissão a participantes da rede que desejam tornar-se mineradores e adicionar blocos através do protocolo de consenso [Nakamoto 2008, Wood 2014]. Em correntes de blocos federadas e privadas, também conhecidas como **permissionadas**, um novo nó minerador deve obter permissão de escrita dos demais mineradores através, respectivamente, de consenso ou de uma decisão centralizada.
- **Permissão de leitura.** Transações em uma corrente de blocos pública são publicamente acessíveis. Em correntes de blocos federadas, o acesso pode ser público ou restrito aos membros das federações, e varia para cada aplicação [Androulaki et al. 2018, Smolensk 2018]. Em correntes privadas, a leitura só é permitida às entidades autorizadas.

Tabela 1.1. Comparação entre diferentes categorias de correntes de blocos.

	Pública	**Federada**	**Privada**
Permissão de Escrita	Não-permissionada	Permissionada	Permissionada
Permissão de Leitura	Pública	Pública ou Privada	Privada
Modelo de Consenso	Baseado em prova	Baseado em mensagens	Opcional
Incentivo	Obrigatório	Opcional	Opcional
Eficiência	Baixa	Média	Alta
Confiabilidade	Alta	Média	Baixa

- **Modelo de consenso.** Em correntes públicas, o protocolo de consenso considera alterações arbitrárias no conjunto de nós mineradores, resultantes da liberdade de escrita, e implica no uso de protocolos baseados em prova com alto custo computacional [Nakamoto 2008, Wood 2014, Ali et al. 2016]. Em correntes federadas, as restrições de permissão de escrita permitem adotar protocolos de consenso mais eficientes que se baseiam em trocas de mensagens entre mineradores [Alvarenga et al. 2018, Androulaki et al. 2018]. Em correntes privadas, o protocolo de consenso é opcional e as decisões na rede podem ser tomadas de forma centralizada.

- **Incentivo.** Correntes de blocos públicas, devido principalmente ao alto custo dos protocolos de consenso, adotam mecanismos para incentivar nós mineradores a gastar recursos computacionais para propor novos blocos. Em correntes de blocos federadas e privadas, o incentivo é opcional pois os nós mineradores normalmente têm interesse direto na manutenção da corrente de blocos e adotam protocolos menos custosos.

- **Eficiência.** Os protocolos baseados em prova e as grandes quantidades de participantes tornam as correntes de blocos públicas pouco eficientes. Correntes de blocos federadas possuem eficiência maior que as correntes públicas, mas a tomada colaborativa de decisões implica em menor eficiência que as correntes de blocos privadas [Vukolić 2016].

- **Confiabilidade.** A grande descentralização de correntes de blocos públicas dificulta que um atacante domine uma parcela significativa do consenso e confere maior confiabilidade à rede. No entanto, em correntes federadas e principalmente em correntes privadas, a quantidade de mineradores e pode trazer riscos à segurança da rede. Em especial, correntes federadas e privadas são menos tolerantes a ataques de conluio e ataques *Sybil* [Douceur 2002].

Além das características citadas, a anonimidade na rede é uma decisão de projeto fundamental que define a forma de identificar indivíduos na rede. A maior parte das aplicações de correntes de blocos utiliza uma chave pública ou um *hash* assinado como endereço [Androulaki et al. 2018,Nakamoto 2008,Azaria et al. 2016,Ali et al. 2016]. Este tipo de identificação provê autenticidade às transações e garante a pseudo-anonimidade dos usuários. Ainda que a anonimidade não seja totalmente garantida, a maior parte das carteiras automatiza a criação um novo endereço para cada transação realizada, dificultando a associação entre endereço e pessoa física. Para casos em que deseja-se revelar a

identidade de um usuário, é possível associar identificadores a pessoas através de certificados públicos ou dicionários registrados na própria corrente de blocos. Esta identificação é especialmente importante em aplicações que proveem auditabilidade e rastreabilidade de transações [Alvarenga et al. 2018, Bozic et al. 2017, Huh et al. 2017, Lee et al. 2016].

1.3.3. Consenso em Correntes de Blocos

A corrente de blocos é um livro-razão distribuído que reúne uma lista crescente de registros controlada por múltiplas entidades sem garantia de confiança mútua. Os nós mineradores comunicam-se através de uma rede par a par para construir a corrente de blocos de forma colaborativa. No entanto, a conectividade da rede pode ser interrompida e mineradores podem individualmente sofrer falhas desastrosas (*crash failures*), agir de forma maliciosa ou entrar em conluio para controlar parte significativa da rede (falhas bizantinas). É necessário, portanto, adotar um protocolo tolerante a falhas para garantir a consistência do sistema e que todos os mineradores atinjam o mesmo estado global. Assim, a corrente de blocos torna-se uma entidade resiliente que provê segurança, confiabilidade, disponibilidade, auditabilidade e integridade aos usuários.

Consenso é o processo pelo qual um grupo de entidades independentes atingem a mesma decisão de maneira distribuída. Formalmente, todo protocolo de consenso em correntes de blocos deve garantir, sob a presença de falhas, as propriedades:

- **Terminação (*liveness*)**: todo minerador correto [8] eventualmente decide por um bloco B_i a ser inserido na corrente.
- **Acordo (*safety*)**: o bloco B_i de todos os mineradores corretos é idêntico.
- **Validade (*validity*)**: o bloco B_i é o bloco proposto por todos os mineradores corretos no início do consenso.

A propriedade de validade pode ser interpretada como a corretude da decisão, garantindo que o bloco adicionado à corrente é o bloco proposto pelos mineradores corretos. Schneider *et al.* definem dois elementos necessários para obter consenso entre processos distribuídos sob a presença de falhas [Schneider 1993]: i) uma máquina de estados replicada e determinística que armazena um estado global da rede; e ii) um protocolo de consenso que realiza transações de estado de forma descentralizada. Através destes dois elementos, é possível implementar a replicação de máquina de estados (*State Machine Replication* - SMR), uma técnica de tolerância a falhas que garante a consistência de um sistema distribuído.

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;

[8] Um minerador correto é um minerador que não está em estado de falha.

- **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]. 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].

1.3.3.1. Premissas de um Ambiente de Corrente de Blocos

Um dos pontos mais importantes para a construção de um protocolo de consenso de correntes de blocos são as premissas assumidas em um ambiente distribuído. As premissas devem cobrir todos os elementos essenciais do sistema, como o número máximo de falhas de mineradores, o modelo de sincronia da rede, o modelo de falha dos mineradores, o modelo de atacante, etc. Uma premissa representa uma idealização do mundo real e deve ser avaliada constantemente para cada caso de implementação [Cachin and Vukolić 2017, Vukolić 2016, Zheng et al. 2018]. A seguir, apresenta-se três premissas fundamentais a todo sistema baseado em correntes de blocos.

Uma premissa comum de confiabilidade em uma corrente de blocos com n mineradores independentes é a de que no máximo $f < n/k$ mineradores estarão em estado de falha, onde $k = 2, 3, ..., n$ e os demais $n - f$ mineradores são corretos. Em correntes de blocos públicas, como o Bitcoin e Ethereum [Nakamoto 2008, Wood 2014], assume-se que até 50% da rede pode estar em estado de falha sem comprometer a corretude do sistema. Em implementações federadas e privadas baseadas em protocolos tolerantes a falhas bizantinas [Androulaki et al. 2018, Schwartz et al. 2014, Buchman 2016, Sousa et al. 2018], a tolerância é de até 33% da rede em estado de falha.

O **modelo de falha** de uma corrente de blocos define o tipo de falha que pode ocorrer em nós mineradores. No modelo de falhas desastrosas (*crash failures*), os mineradores podem parar de responder às mensagens do consenso devido a panes, perda de conectividade ou quedas de energia [Ongaro and Ousterhout 2014, Lamport 1998]. No modelo de falhas bizantinas, os mineradores podem, além de não responder, responder às mensagens de forma arbitrária e maliciosa para subverter o sistema [Lamport et al.

1982, Dolev 1982]. No modelo de falhas bizantinas, um processo falho pode exibir qualquer comportamento, incluindo panes, omissão de envios e entregas de mensagens, ou emissão de mensagens falsas de forma proposital. Devido à característica fundamental de desconfiança mútua entre os participantes de uma corrente de blocos, o modelo de falhas bizantinas é amplamente mais utilizado em sistemas baseados em correntes de blocos [Nakamoto 2008, Androulaki et al. 2018, Wood 2014]. O modelo de falha de um sistema de corrente de blocos influencia diretamente no funcionamento do protocolo de consenso a ser adotado.

O **modelo de sincronia** do sistema define o tipo de sincronia existente entre os nós mineradores. No modelo síncrono, assume-se que as mensagens do consenso são sempre entregues sem atraso ou com um atraso bem definido. No modelo eventualmente síncrono [Dwork et al. 1988], as mensagens podem atrasar arbitrariamente, mas chegam ao destino em um tempo finito. No modelo totalmente assíncrono, não há garantia de que as mensagens cheguem ao destino. Não há possibilidade de consenso no modelo assíncrono [Fischer et al. 1985]. As principais propostas de protocolos de consenso em correntes de blocos assumem o modelo eventualmente síncrono devido à sua simplicidade e aplicabilidade ao mundo real [Buchman 2016, Sousa et al. 2018]. Outras propostas utilizam o modelo assíncrono, sem assumir qualquer premissa sobre a infraestrutura de rede [Miller et al. 2016, Duan et al. 2018].

1.3.4. Modelos de Consenso

Um dos principais desafios de consenso em sistemas distribuídos é o resultado de impossibilidade do consenso conhecido como FLP[9]. O resultado prova que, em um sistema com modelo assíncrono e com n participantes identificados, o consenso não possui solução determinística mesmo na presença de uma única falha desastrosa [Fischer et al. 1985]. Assim, para contornar o problema de obtenção de consenso em correntes de blocos, há duas abordagens possíveis: relaxar as garantias do consenso, comprometendo o acordo (*safety*), ou relaxar o modelo de sincronia, assumindo pelo menos o modelo eventualmente síncrono. Como consequência das duas abordagens, o problema do consenso pode ser tratado, respectivamente, de maneira probabilística ou determinística. O consenso probabilístico assume a consistência eventual da rede, isto é, que, na ausência de novas transações, todo participante eventualmente atinge o mesmo estado global [Tseng 2017, Decker et al. 2016]. Isto representa um enfraquecimento da propriedade de acordo (*safety*), uma vez que os participantes podem divergir sobre os dados mais recentes até que a consistência seja atingida. A propriedade de terminação, entretanto, é garantida.

Protocolos de consenso probabilístico baseiam-se em difundir uma decisão local para os vizinhos e, eventualmente, atingir todos os nós mineradores da rede. Em correntes de blocos, isto significa que um bloco é proposto por um nó minerador e adicionado à corrente de blocos antes de ser difundido na rede. Caso o bloco esteja correto, garante-se que os demais nós mineradores eventualmente o validarão e atingirão o mesmo estado global. Consensos probabilísticos possuem como principal vantagem a escalabilidade, uma vez que não é necessário conhecer todos os nós mineradores da rede para se obter consenso. Portanto, este tipo de consenso é mais adequado a correntes de blocos públi-

[9]O teorema recebe esta sigla em homenagem a seus autores: Fisher, Lynch e Patterson.

cas com muitos nós mineradores. Em consensos probabilísticos, dois mineradores podem propor blocos corretos simultaneamente, causando uma bifurcação na corrente de blocos. Nesse caso, todo protocolo de consenso probabilístico deve adotar um algoritmo de desempate para definir uma verdade global. A regra da cadeia mais longa no Bitcoin é a regra de desempate mais conhecida em correntes de blocos [Nakamoto 2008].

Os protocolos de consenso mais comuns em sistemas de consistência eventual são os algoritmos baseados em prova, nos quais um nó minerador que propõe um bloco deve apresentar uma prova de que pode liderar o consenso [Nakamoto 2008, Vasin 2014, Olson et al. 2018]. Nakamoto propôs a prova de trabalho (*Proof of Work* - PoW) como um algoritmo de consenso no qual os mineradores devem provar que gastaram recursos computacionais para resolver independentemente um desafio matemático computacionalmente custoso. Este desafio depende apenas do estado mais recente da corrente de blocos e portanto é totalmente paralelizável. A dificuldade do desafio 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 e, 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.

Apesar da inovação, a prova de trabalho do Bitcoin apresenta limitações evidentes. A latência de consenso, de aproximadamente dez minutos, e a vazão de transações, de apenas 7 transações por segundo, consistem em um desafio para atender às necessidades dos sistemas atuais [Popov 2016, Eyal et al. 2016]. 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.

- **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].

- **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].

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

- **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].

- **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].
- **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.

O consenso tolerante a falhas bizantinas utiliza a primeira abordagem, oferecendo garantias determinísticas e apoiando-se em redes parcialmente síncronas para assegurar o progresso.

Em redes bem-definidas, como redes permissionadas síncronas ou eventualmente síncronas, é possível obter consenso determinístico apoiando-se nas garantias de entrega de mensagens da rede. Neste caso, enquanto a rede não oferece as condições de estabilidade (sincronia) para que haja consenso, a terminação (*liveness*) é comprometida, mas não o acordo (*safety*). Assim, o consenso determinístico sempre assegura a consistência do sistema, possivelmente sacrificando seu progresso.

O consenso determinístico é alcançado através de protocolos de consenso baseados em quórum. Neste 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 devem ser conhecidos, identificados e deve haver sincronia entre os nós para a troca de mensagens. O consenso baseado em quórum garante que a adição de um bloco à corrente ocorre ao mesmo tempo para todas as réplicas. Portanto, a consistência do sistema é garantida em qualquer momento e não existem bifurcações na corrente de blocos. A tolerância a falhas e comportamento malicioso é definida de acordo com as especificações de cada protocolo de consenso. Um consenso determinístico baseado em quórum é mais eficiente que algoritmos baseados em prova [Schwartz et al. 2014, Dinh et al. 2017].

Uma classe de protocolos de replicação de máquina de estados particularmente interessante para as correntes de blocos é a de protocolos tolerantes a falhas bizantinas (*Byzantine Fault Tolerant* - BFT), que garantem a consistência da corrente de blocos sob a presença de comportamento malicioso [Lamport et al. 1982]. Protocolos BFT são capazes de atingir até dezenas de milhares de transações por segundo com baixa latência, sob a restrição de operarem em redes com sincronia eventual e com poucos nós mineradores [Cachin and Vukolić 2017, Vukolić 2016]. Isto faz com que os protocolos BFT sejam ideais para redes permissionadas nas quais os participantes podem agir de maneira maliciosa. Em especial, a tolerância a falhas bizantinas assume as condições propostas no problema dos generais bizantinos, descrito por Lamport [Lamport et al. 1982]:

- todos os participantes são conhecidos e identificados[10];

[10]Existem trabalhos que estudam o consenso bizantino em que os participantes são desconhecidos [Al-

- todos os participantes podem trocar mensagens diretamente;
- as mensagens trocadas chegam ao destino em tempo finito;
- toda mensagem é assinada por seu emissor;
- todo participante pode falhar, incluindo o líder do consenso;
- participantes em falha podem mentir ou omitir mensagens;
- as mensagens podem ser alteradas por um terceiro no meio do caminho;
- as mensagens podem ser perdidas, reordenadas ou duplicadas;
- o caminho entre dois participantes pode ser interrompido;

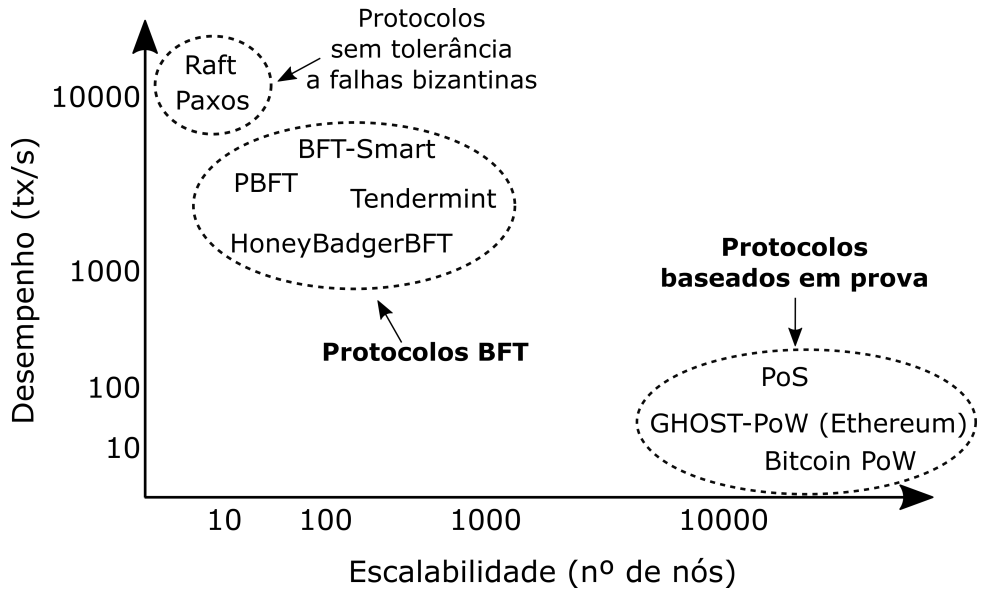

Figura 1.8. Comparação de desempenho e escalabilidade de protocolos sem tolerância a falhas bizantinas, protocolos BFT e protocolos baseados em prova. Os protocolos BFT sacrificam escalabilidade para tolerar falhas bizantinas com maior desempenho em redes permissionadas.

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. Os participantes do consenso respondem ao líder com suas avaliações e as difundem para todos os outros participantes, para gerar provas coletivas da avaliação e prevenir uma possível ação maliciosa do líder. Ao receber uma quantidade suficiente de votos positivos, cada participante considera que o quórum foi alcançado e os participantes escrevem o novo bloco na corrente simultaneamente. Protocolos BFT toleram no máximo $f = \frac{n}{3} + 1$ falhas de participantes. Alguns protocolos toleram menos falhas devido a decisões de projeto [Schwartz et al. 2014, Buchman 2016].

chieri et al. 2018]

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 desastrosas [Ongaro and Ousterhout 2014, Lamport 1998] pois toleram comportamento malicioso na rede. No entanto, toleram menos falhas do que os clássicos 50% dos protocolos baseados em prova [Nakamoto 2008].

A Figura 1.8 apresenta uma comparação em alto nível dos principais protocolos de consenso adotados em sistemas baseados em livro-razão distribuído [Mingxiao et al. 2017, Han et al. 2018, Santos and Schiper 2012, Buchman 2016, Bessani et al. 2014, Vukolić 2016, Cachin and Vukolić 2017]. A eventual consistência dos protocolos baseados em prova provê a escalabilidade necessária para as redes públicas, em detrimento da eficiência. Os protocolos BFT proveem alto desempenho para ambientes com número limitado e conhecido de nós. A eficiência e escalabilidade de cada classe de consenso está diretamente associada às suas premissas. Os protocolos baseados em prova são os mais seguros, escaláveis e menos eficientes. Os protocolos BFT e tolerantes a falhas desastrosas são mais eficientes e menos seguros. A análise minuciosa do modelo de segurança e requisitos de eficiência da rede, portanto, é fundamental para a escolha do protocolo de consenso.

1.4. Fatiamento Seguro de Redes Através de Corrente de Blocos

Esta seção mostra um estudo de caso de aplicação da tecnologia de corrente de blocos para prover segurança em um ambiente com virtualização de funções de rede multi-inquilino e multiusuário que não possuem confiança entre si.

As redes móveis de próxima geração fornecem um modelo de conectividade com múltiplos serviços de rede adaptados para atender a demanda de cada segmento de cliente. As redes definidas por *software* (*Software-Defined Networking* - SDN) e a virtualização de funções de rede (*Network Function Virtualization* - NFV) são as principais tecnologias que utilizam a virtualização para fornecer agilidade, automação e capacidade de programação da rede. Assim, as tecnologias NFV e SDN criam uma cadeia de funções de rede (*Service Function Chain* - SFC) fim-a-fim [Halpern and Pignataro 2015] para fornecer serviços sob demanda e adaptados a cada aplicação. Apesar de a associação de NFV e SDN fornecer a agilidade e o baixo custo desejados pelas telecomunicações, surgem novos desafios de segurança [Medhat et al. 2017]. Além disso, o impacto de possíveis ataques aumenta, pois ataques a hospedeiros de funções de rede podem comprometer simultaneamente milhares de usuários [Bhamare et al. 2016]. Portanto, é de grande importância reduzir os possíveis vetores de ataque a funções virtuais de rede (*Virtual Network Functions* - VNF) e fornecer um gerenciamento de configuração seguro e confiável. Garantir o isolamento entre fatias de rede é essencial para evitar ataques comuns em infraestruturas compartilhadas. Além disso, os inquilinos de cada fatia compartilham a mesma infraestrutura de nuvem, e as cadeias podem envolver funções virtualizadas instanciadas em domínios de provedores concorrentes. O ambiente multi-inquilino e multi-domínio aumenta a possibilidade de ataques dentro da nuvem, ao mesmo tempo que dificulta a responsabilização dos provedores de serviços quando ocorre uma falha. É necessário garantir que a cadeia de serviços seja construída de maneira confiável em um ambiente sem confiança entre os pares. Logo, a capacidade de auditoria é obrigatória para identificar

uma configuração de VNF defeituosa ou comprometida, e a tecnologia de corrente de blocos fornece as características necessárias de não repúdio e imutabilidade do histórico de configuração de uma função virtual.

Rebello *et al.* propõem uma arquitetura na qual cada fatia de rede baseada em corrente de blocos aborda um ou mais casos de uso do 5G, criando redes isoladas com segurança e confiança [Rebello et al. 2019b]. A Figura 1.9 descreve um cenário que usa corrente de blocos para três fatias de rede: uma fatia de rede móvel, uma fatia de indústria 4.0 e uma fatia de redes veiculares. Para garantir a justiça no protocolo de consenso, cada centro de dados pode hospedar no máximo um nó de corrente de blocos por fatia de rede. Os nós de corrente de blocos em uma fatia são invisíveis para qualquer entidade fora da fatia.

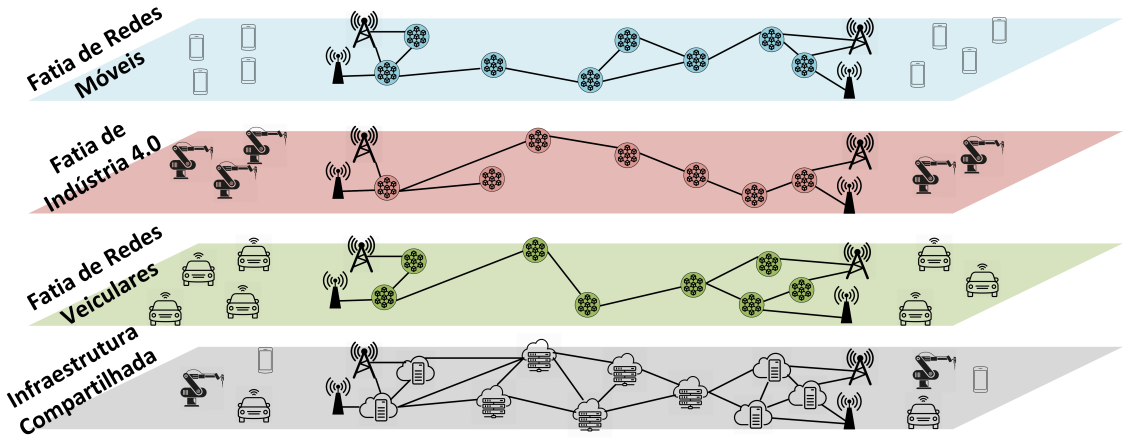

Figura 1.9. Fatias de rede isoladas através de corrente de blocos em uma infraestrutura física compartilhada. Cada fatia de rede é adaptada às necessidades de um caso de uso.

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. A corrente de blocos de gerenciamento registra as operações de orquestração que criam ou modificam uma fatia de rede. Toda operação é assinada pelo cliente que solicitou a modificação. Os participantes da corrente de blocos devem validar cada operação por consenso e fornecer uma prova assinada irrefutável de que a transação foi aceita antes que as operações sejam realizadas. A solicitação assinada combinada com o registro permanente fornecido pela corrente de blocos garante que um comportamento malicioso seja rastreável. Assim, o emprego de uma corrente de blocos gerencial garante a procedência, a prestação de contas e a rastreabilidade das falhas em um ambiente NFV multi-inquilino e multi-domínio. Além disso, a corrente de blocos pode se servir de contratos inteligentes para prover automação e transparência em ambientes sem confiança distribuídos, em vez de confiar em um determinado nó para receber e processar transações. As propriedades de automação e transparência dos contratos inteligentes são ideais 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.

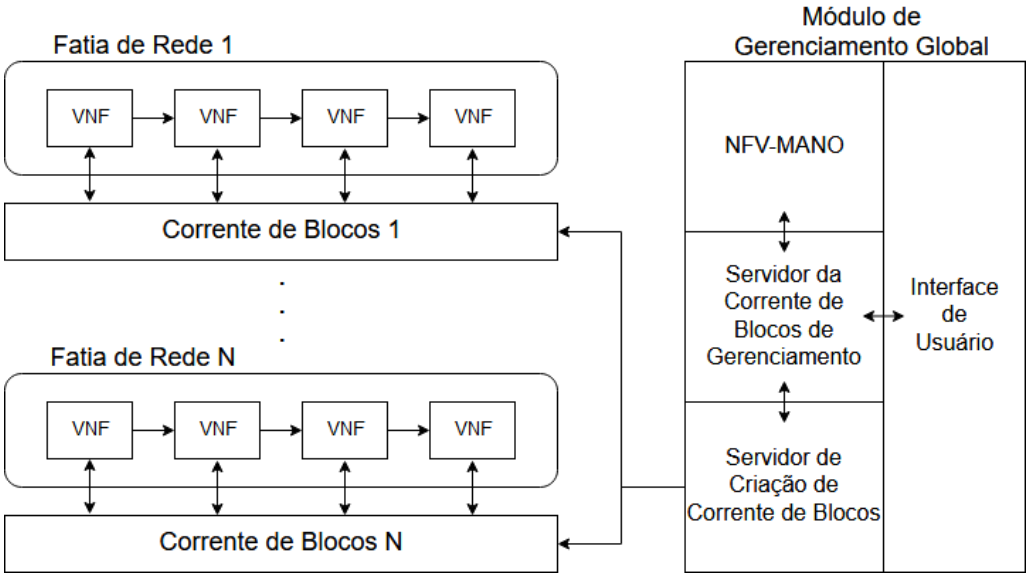

Figura 1.10. A arquitetura proposta por Rebello *et al.* baseada em corrente de blocos para fatiamento de rede. O usuário interage com o módulo de gerenciamento global para criar fatias de rede seguras. Cada VNF em uma fatia de rede é conectada a uma corrente de blocos responsável por registrar solicitações de configuração e informações relevantes, conforme especificado pelo usuário.

A Figura 1.10 mostra os quatro componentes principais: uma interface de usuário, o módulo de gerenciamento e orquestração NFV (NFV-MANO), um módulo de servidor de criação de corrente de blocos e um módulo de corrente de blocos de gerenciamento. Os módulos compõem o módulo de gerenciamento global, que é responsável por conectar o cliente aos serviços oferecidos. Nesta arquitetura, o cliente interage com o módulo de gerenciamento global por meio de uma interface de usuário para criar/modificar uma fatia de rede segura ou para solicitar informações de fatia, como configurações de VNFs e cadeias de funções. O cliente especifica as características da fatia, como VNFs desejadas e restrições de posicionamento, e a categoria de corrente de blocos correspondente. O módulo de interface do usuário converte as especificações em operações de criação de fatias/corrente de blocos e as envia como transações assinadas para aprovação na corrente de blocos de gerenciamento. Depois que as transações são aprovadas, o módulo NFV-MANO e o módulo de criação de corrente de blocos verificam a corrente de blocos de gerenciamento para obter operações pendentes. Os módulos executam as operações para criar novas fatias seguras e emitem transações de resposta assinadas para a corrente de blocos de gerenciamento. O cliente pode então interagir com o módulo de interface do usuário para verificar se a fatia segura solicitada foi criada com sucesso.

1.5. 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, registrando 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 apresentada na seção anterior com dois tipos de corrente de blocos. 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.

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[11] ou VMWare Workstation Player[12] 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[13].

1.5.1. Auditoria de Criação, Leitura, Atualização e Remoção de Funções de Rede

A programabilidade do núcleo da rede expõe as funções de redes a um alto número de ameaças. Em um cenário multi-inquilino e multi-domínio, as ameaças podem afetar o tráfego que flui por uma VNF afetada. Desta maneira, a identificação de funções comprometidas de rede é essencial para garantir a segurança de um ambiente de funções virtualizadas de rede. O uso de corrente de blocos permite o registro permanente de operações envolvendo funções de rede, garantindo a auditabilidade do histórico de operações. Além disso, o uso de criptografia de chaves assimétricas garante a identificação e facilita a responsabilização dos envolvidos no comprometimento da função.

1.5.1.1. A Plataforma Hyperledger Fabric

O Hyperledger[14] 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 empre-

[11]Disponível em https://www.virtualbox.org/
[12]Disponível em https://www.vmware.com/br/products/workstation-player.html
[13]Disponível em https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html
[14]Disponível em https://www.hyperledger.org/

sas 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 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. Isto é ideal 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 1.11 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.

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

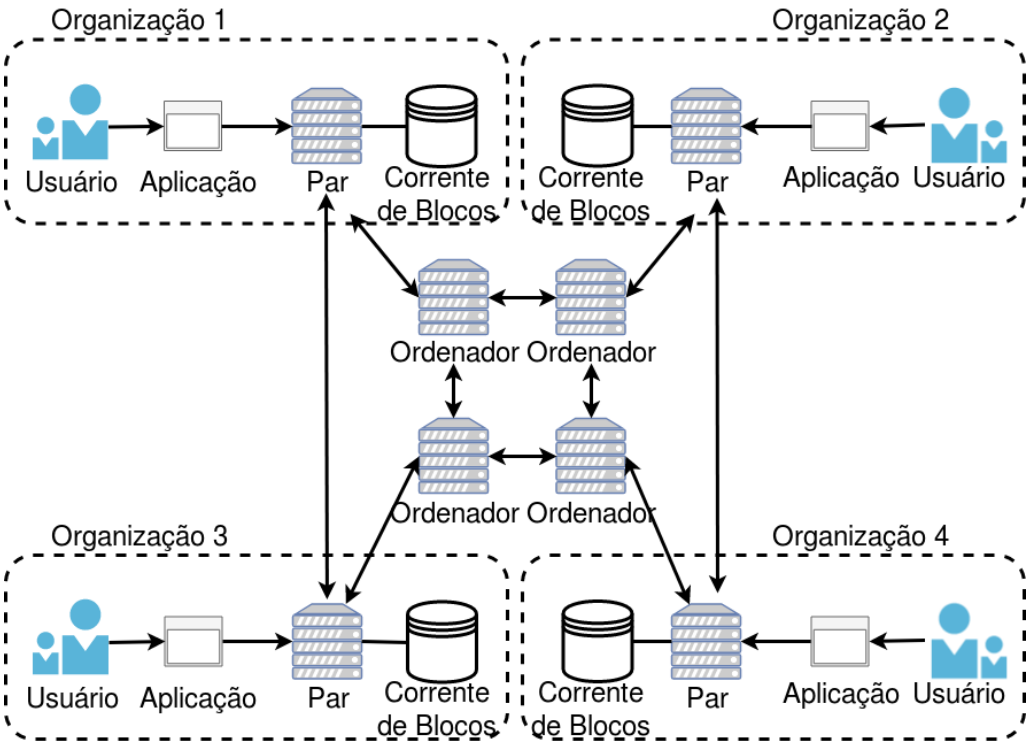

Figura 1.11. Uma corrente de blocos permissionada do Hyperledger Fabric. Usuários de cada organização usam aplicações para emitir transações assinadas e as submetem para validação nos pares. Os pares validam a transação e respondem à aplicação, que retransmite a transação validada para os ordenadores. 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.

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.

1.5.1.2. Exemplo de Desenvolvimento 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

```
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 1.1. 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.

protótipo implementado segue a arquitetura de fatiamento de rede apresentada na seção anterior. 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[15] escritos em Go, que são executados em todos os nós da rede [Alvarenga et al. 2018, Rebello et al. 2019a].

O primeiro contrato inteligente, parcialmente descrito na Lista 1.1, 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 1.2, 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

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

```
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 1.2. Pseudocódigo parcial para emitir transações de configuração. O campo configurationIdentifier contêm um identificador único para a configuração.

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 1.4.0 do Hyperledger Fabric. Os comandos de versões anteriores ou posteriores podem ser incompatíveis com a versão utilizada nesta atividade. Para baixar os arquivos necessários, os participantes da aula prática devem executar os seguintes comandos:

```
git clone https://github.com/keenkit/fabric-sample-
with-kafka.git
```

```
cd fabric-sample-with-kafka
```

```
sed -i 's/1.2.0/1.4.0/' get-byfn.sh && ./get-byfn.sh
```

```
cd first-network
```

```
curl -sSL http://bit.ly/2ysbOFE | bash -s 1.4.0 1.4.0
0.4.14
```

O participante deve criar o diretório `contract` dentro do diretório `examples/chaincode` e baixar o contrato inteligente que será utilizado ao longo desta atividade. De dentro do diretório `first-network`, o participante deve executar:

```
cd examples/chaincode && mkdir contract && cd contract
```

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

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. Também é necessário o arquivo `configtx.yaml` para configuração da rede. Nele 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.

De dentro do diretório `first-network`, o usuário deve gerar os certificados que serão usados na rede utilizando o comando:

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

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

O comando cria um diretório chamado `crypto-config` que contém os certificados e chaves da rede. Em seguida, 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: TwoOrgsOrdererGenesis
- 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: sbrcchannel[16]

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

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

Pares-âncora (*anchor peers*) 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
- 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. A plataforma Fabric disponibiliza o arquivo de configuração `docker-compose-kafka.yaml` para facilitar a configuração e instanciação de contêineres.

```
IMAGE_TAG=latest docker-compose -f <arquivo do Docker
Compose> up -d
```

- arquivo do Docker Compose: docker-compose-kafka.yaml

Para acessar o contêiner cliente:

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

Agora, o usuário deve passar as configurações do canal criada em um dos passos anteriores para criar o canal do Hyperledger Fabric:

```
export CHANNEL_NAME=sbrcchannel
```

```
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 sbrcchannel.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 1 e por um par pertencente à organização 2.

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/orderer.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"]}’

1.6. Considerações Finais, Perspectivas e Problemas em Aberto

A tecnologia de virtualização de funções de rede permite diferenciar serviços encadeando funções virtualizadas entre diferentes provedores de infraestruturas de nuvem sem garantia de confiança entre os pares. Cada operadora possui um centro de dados, que contém um orquestrador de funções virtualizadas de rede (*Virtual Network Functions* - VNF) e uma infraestrutura de virtualização baseada em máquinas de propósito geral para hospedar máquinas virtuais e VNFs. A comunicação entre pontos de extremidade é realizada através da conexão entre os grandes centros de dados (*data centers*), que são capazes de prover, sob demanda, serviços fim-a-fim adaptados a cada aplicação. Neste cenário multi-inquilino e multi-domínio, é imprescindível identificar a ocorrência de falhas e responsabilizar agentes maliciosos, que podem comprometer o bom comportamento e qualidade de serviço de milhares de usuários de um serviço simultaneamente. Portanto, uma solução de segurança baseada em corrente de blocos que garante confiança e imutabilidade dos registros em ambientes distribuídos é apropriada para solucionar estes desafios.

Com objetivo de apresentar um estudo de caso de corrente de blocos para garantir segurança em um ambiente de virtualização de redes, este capítulo apresentou as tecnologias de virtualização de redes de correntes de blocos. Em relação a corrente de blocos o capítulo apresentou a estrutura, propriedades e características de algumas correntes de blocos e, além disso, descreveu as categorias de correntes de blocos, comparando algoritmos de consenso e modelos de falha. Focou-se em mapear as principais características de correntes de blocos para atender às necessidades de segurança de ambientes de virtualização de funções de rede.

Também foi apresentado um estudo de caso de fatiamento de redes usando corrente de blocos. A ideia básica é propor correntes de blocos específicas e sob medida para cada cadeia de funções de rede que proveem serviços de rede. Assim, cada proposta atende a um requisito do ambiente virtualizado, como auditabilidade, autenticidade e rastreabilidade. No protótipo apresentado a decisão pelo protocolo de consenso proposto considerou a baixa quantidade e alta disponibilidade de nós identificados.

1.6.1. 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 2016]. 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 [Wiki 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[17]. 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.

1.6.2. Atividades em Organismos de Normalização

A aplicabilidade de da tecnologia de livro razão distribuído 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].

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

1.6.3. 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.

1.6.4. Revogação de Chaves em Correntes de Blocos

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 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].

1.6.5. 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].

1.6.6. 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 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 dados 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 permite 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.

1.6.7. Considerações e Perspectivas

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 a tecnologia mais importante depois da Internet. As criptomo-

edas já são um sucesso e existem predições que afirmam que todo o dinheiro do mundo estará em uma corrente de blocos nos próximos dez anos. Existe uma série de aplicações de sucesso com contratos inteligentes e garantia de proveniência em diferentes áreas. 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. 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 o mesmo nível de segurança, mas as curvas elípticas requerem chaves de menor tamanho.

Assinatura Digital

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.

Uma mensagem criptografada por uma chave privada só pode ser descriptografada pelo seu par de chave pública. Esta operação é conhecida como assinatura, pois identifica inequivocamente a pessoa que possui a chave privada, que é secreta e apenas seu proprietário possui.

A utilização de curvas elípticas em criptografia 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 1.12 ilustra a curva elíptica "secp256k1", usada na criptografia assimétrica do Bitcoin e definida nas normas para criptografia eficiente (*Standards for Efficient.00Cryptography* - SEC)[18].

O algoritmo de assinatura digital por curvas elípticas (*Elliptic Curve Digital Sig-*

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

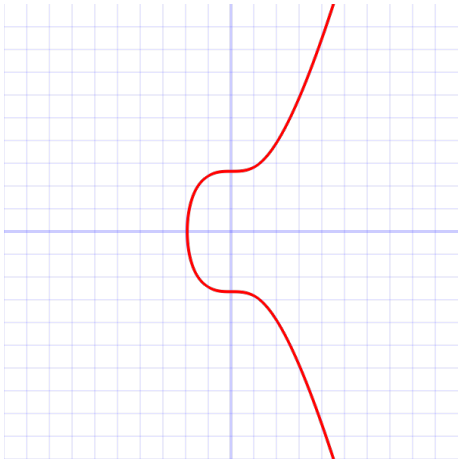

Figura 1.12. Curva Elíptica secp256k1 usada na moeda eletrônica Bitcoin.

nature 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[19].

Função *Hash*

Uma função *hash* ou função resumo 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*, descrita por $H(m) = h$, onde m é a mensagem em claro, $H()$ é a função *hash* e h é o valor *hash* resultante, 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 $H(m)$ 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

[19] 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.

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*;

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 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.

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 1.13 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].

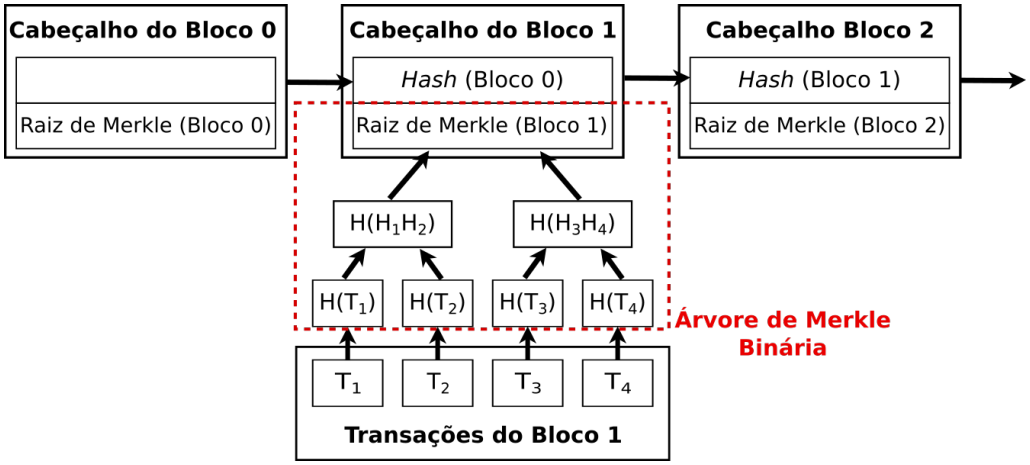

Figura 1.13. Estrutura de uma árvore de Merkle binária utilizada em correntes de blocos. O uso de árvores de Merkle permite armazenar apenas a raiz da árvore sem prejuízo à verificação das transações.

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. 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 2018] Alvarenga, I. D. (2018). Securing Configuration, Management and Migration of Virtual Network Functions using Blockchain. Master Science Thesis, Universidade Federal do Rio de Janeiro, Brasil.

[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., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S. W., and Yellick, J. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 30:1–30:15, New York, NY, USA. 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)*.

[Arnott et al. 2016] Arnott, M., Owen, P., Buckland, E., and Ranken, M. (2016). IoT global forecast analysis 2015-2025. Technical report, Machina Research. `https://machinaresearch.com/login/?next=/report/iot-global-forecast-analysis-2015-25/`. Acessado em 09 de abril de 2019.

[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 9 de abril de 2019.

[Bessani et al. 2014] Bessani, A., Sousa, J., and Alchieri, E. E. P. (2014). State Machine Replication for the Masses with BFT-SMART. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362.

[Bhamare et al. 2016] Bhamare, D., Jain, R., Samaka, M., and Erbad, A. (2016). A survey on service function chaining. *Journal of Network and Computer Applications*, 75:138–155.

[Bouras et al. 2017] Bouras, C., Kollia, A., and Papazois, A. (2017). SDN & NFV in 5G: Advancements and challenges. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 107–111.

[Bozic et al. 2017] Bozic, N., Pujolle, G., and Secci, S. (2017). Securing Virtual Machine Orchestration with Blockchains. In *2017 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, The University of Guelph.

[Cachin and Vukolić 2017] Cachin, C. and Vukolić, M. (2017). Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*.

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

[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.

[Dai 1998] Dai, W. (1998). B-money. `http://www.weidai.com/bmoney.txt`. Acessado em 9 de abril de 2019.

[Decker et al. 2016] Decker, C., Seidel, J., and Wattenhofer, R. (2016). Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13. ACM.

[Digiconomist 2019] Digiconomist (2019). Bitcoin Energy Consumption Index. `https://digiconomist.net/bitcoin-energy-consumption`. Acessado em 9 de abril de 2019.

[Dinh et al. 2017] Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017). Blockbench: A framework for analyzing private blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1085–1100. ACM.

[Dolev 1982] Dolev, D. (1982). The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14 – 30.

[Dolev and Yao 1983] Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208.

[Douceur 2002] Douceur, J. R. (2002). The Sybil Attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK. Springer-Verlag.

[Duan et al. 2018] Duan, S., Reiter, M. K., and Zhang, H. (2018). BEAT: Asynchronous BFT made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2028–2041. ACM.

[Dwork et al. 1988] Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323.

[ETSI 2014] ETSI (2014). ETSI GS NFV-MAN 001: Network Functions Virtualisation; Management and Orchestration.

[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 2018] Eyal, I. and Sirer, E. G. (2018). Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102.

[Finney 2005] Finney, H. (2005). RPOW: Reusable proofs of work. `http://nakamotoinstitute.org/finney/rpow/theory.html`. Acessado em 9 de abril 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.

[Halpern and Pignataro 2015] Halpern, J. and Pignataro, C. (2015). Service Function Chaining (SFC) architecture. RFC 7665, RFC Editor. `http://www.rfc-editor.org/rfc/rfc7665.txt`. Acessado em 9 de abril de 2019.

[Han et al. 2015] Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97.

[Han et al. 2018] Han, R., Gramoli, V., and Xu, X. (2018). Evaluating blockchains for iot. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

[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.

[Huh et al. 2017] Huh, S., Cho, S., and Kim, S. (2017). Managing IoT devices using blockchain platform. In *Advanced Communication Technology (ICACT), 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 9 de abril 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 9 de abril 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 *Annual International Cryptology Conference*, pages 357–388. Springer.

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

[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.

[Li and Chen 2015] Li, Y. and Chen, M. (2015). Software-Defined Network Function Virtualization: A Survey. *IEEE Access*, 3:2542–2553.

[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.

[Medhat et al. 2017] Medhat, A. M., Taleb, T., Elmangoush, A., Carella, G. A., Covaci, S., and Magedanz, T. (2017). Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges. *IEEE Communications Magazine*, 55(2):216–223.

[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.

[Mijumbi et al. 2016] Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

[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.

[Mingxiao et al. 2017] Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., and Qijun, C. (2017). A review on consensus algorithm of blockchain. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2567–2572. IEEE.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. `http://bitcoin.org/bitcoin.pdf`. Acessado em 9 de abril de 2019.

[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 *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA. USENIX Association.

[Pattaranantakul et al. 2016] Pattaranantakul, M., He, R., Meddahi, A., and Zhang, Z. (2016). SecMANO: Towards Network Functions Virtualization (NFV) Based Security MANagement and Orchestration. In *IEEE Trustcom/BigDataSE/ISPA*, pages 598–605.

[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 09 de Abril de 2019.

[Popov 2016] Popov, S. (2016). The Tangle. `https://iota.org/IOTAWhitepaper.pdf`. Acessado em 9 de abril de 2019.

[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 9 de abril de 2019.

[Raman 1998] Raman, L. (1998). OSI systems and network management. *IEEE Communications Magazine*, 36(3):46–53.

[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. 2018] Rebello, G. A. F., Alvarenga, I. D., Sanz, I. J., and Duarte, O. C. M. B. (2018). SINFONIA: Gerenciamento Seguro de Funções Virtualizadas de Rede através de Corrente de Blocos. In *I Workshop em Blockchain: Teoria, Tecnologias e Aplicações (WBlockchain - SBRC 2018)*, Campos do Jordão, SP, Brasil. SBC.

[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., Guimarães, L. C. B., de Souza, L. A. C., Alvarenga, I. D., and Duarte, O. C. M. B. (2019b). 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).

[Santos and Schiper 2012] Santos, N. and Schiper, A. (2012). Tuning Paxos for High-throughput with Batching and Pipelining. In *Proceedings of the 13th International Conference on Distributed Computing and Networking*, ICDCN'12, pages 153–167, Berlin, Heidelberg. Springer-Verlag.

[Schneider 1993] Schneider, F. B. (1993). chapter Replication Management Using the State-machine Approach. In Mullender, S., editor, *Distributed Systems (2Nd Ed.)*, pages 169–197. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

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

[Sekar et al. 2012] Sekar, V., Egi, N., Ratnasamy, S., Reiter, M. K., and Shi, G. (2012). Design and Implementation of a Consolidated Middlebox Architecture. In *9th Symposium on Networked Systems Design and Implementation (NSDI)*, pages 323–336, San Jose, CA. USENIX.

[Smolensk 2018] Smolensk, M. (2018). Lightstreams White Paper. `https://s3.amazonaws.com/lightstreams/lightstreams_whitepaper.pdf.Acessadoem9deabrilde2019.`

[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, Statista. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Acessado em 9 de abril de 2019.

[Storj Labs 2018] Storj Labs, I. (2018). Storj: A Decentralized Cloud Storage Network Framework. `https://storj.io/storjv3.pdf`. Acessado em 9 de abril de 2019.

[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.

[Tseng 2017] Tseng, L. (2017). Bitcoin's Consistency Property. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 219–220.

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

[Visa 2019] Visa (2019). About VisaNet. `https://usa.visa.com/about-visa/visanet.html`. Acessado em 9 de Abril de 2019.

[Vukolić 2016] Vukolić, M. (2016). *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*, pages 112–125. Springer International Publishing, Cham.

[W3C 2016] W3C (2016). Blockchain Community Group. `https://www.w3.org/community/blockchain`. Acessado em 9 de abril de 2019.

[Wiki 2019] Wiki, B. (2019). Bitcoin Scalability. `https://en.bitcoin.it/wiki/Scalability`. Acessado em 9 de abril 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 9 de abril de 2019.

[Wood 2014] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. `http://gavwood.com/paper.pdf`. Acessado em 9 de abril de 2019.

[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.

[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.