Capítulo 1

Introdução aos Blockchains: Teoria e Prática

João Otávio Chervinski, Felipe Melchior, Rafael Fernandes, Guilherme Neri, Lucas Antunes, Diego Kreutz, Rodrigo Mansilha (UNIPAMPA)

O lançamento de criptomoedas como a Bitcoin[1], Monero[2] e Ripple[3] deu origem a uma revolução nos sistemas de pagamento no mundo todo. A Bitcoin é o primeiro e mais conhecido caso disruptivo de sucesso. Criptomoedas como a Bitcoin funcionam de maneira descentralizada, isto é, sem a necessidade de uma autoridade reguladora, permitindo que qualquer um participe da rede e efetue transações.

Em sistemas financeiros convencionais, quando pagamentos são efetuados através de uma empresa de cartões de crédito, o usuário estabelece uma relação de confiança com a empresa e delega a ela a responsabilidade de validar os seus dados e efetuar a transação corretamente. Diferentemente dos sistemas tradicionais, uma das principais inovações da Bitcoin foi a criação de um sistema descentralizado no qual participantes são capazes de enviar pagamentos uns aos outros sem a necessidade de estabelecer confiança prévia. Em outras palavras, criptomoedas como a Bitcoin possuem um grande apelo pelo fato de eliminarem a necessidade de uma autoridade intermediária. Isto é realizado através da tecnologia de *blockchain*.

Um *blockchain* é uma estrutura de dados distribuída, formada por uma série de blocos de informação encadeados. Cada participante da rede pode obter uma cópia completa dos dados e compartilhá-la com outros participantes. Numa rede de *blockchain*, os usuários trabalham de maneira colaborativa para validar transações, utilizando criptografia para garantir a sua segurança e verificabilidade. Mecanismos criptográficos também são utilizados para garantir a ordem dos blocos e evitar a sua adulteração, visto que só deve existir uma sequência válida de blocos. Entretanto, apesar dos *blockchains* serem utilizados principalmente em criptomoedas, já existem propostas e exemplos práticos de aplicação dessa tecnologia em áreas e temas como sistemas financeiros, registro e proteção de propriedade intelectual, sistemas baseados em reputação, cadeia de suprimentos, setor de energia, setor de saúde, e-gov, marketing, sistemas de votação eletrônica, aplicações industriais, sistemas inteligen-

[1] `https://bitcoin.org`
[2] `https://www.getmonero.org`
[3] `https://ripple.com`

tes, realidade aumentada e jogos [Zheng et al. 2018, Hoy 2017, Olnes et al. 2017, Chen et al. 2018, Marinoff 2018, Dai et al. 2019, Al-Jaroodi and Mohamed 2019, Abbas and Sung-Bong 2019, Min et al. 2019, Yang et al. 2019, Xie et al. 2019].

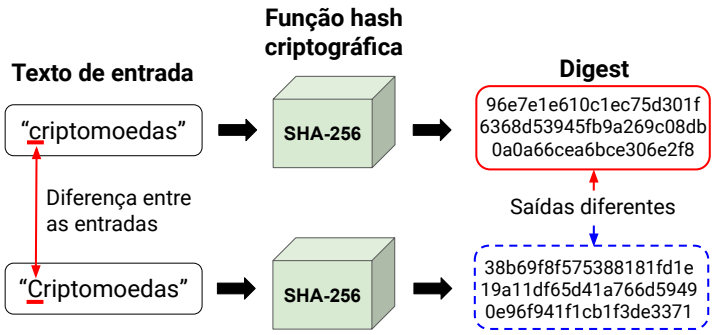
Este minicurso tem como objetivo apresentar os principais conceitos inerentes às tecnologias de *blockchain* e criptomoedas, contribuindo para a difusão do conhecimento desta área que está no seu auge de pesquisa e desenvolvimento. No decorrer do minicurso são abordadas também questões relacionadas à segurança (e.g., integridade de dados) em criptomoedas como a Bitcoin. Por fim, vale ressaltar que atualmente há muitas oportunidades, tanto na academia (exemplo: novas conferências específicas, criadas em 2018 e 2019) quanto no mercado (exemplo: muitas empresas investindo e apostando nessas novas tecnologias), relacionadas diretamente aos temas discutidos aqui.

O restante do minicurso está dividido como segue. Na Seção 1.1 são apresentados e discutidos os detalhes da tecnologia *blockchain*. A Seção 1.2 discute as criptomoedas e métodos de mineração e criação de novas moedas. O funcionamento da criptomoeda Bitcoin é detalhado na Seção 1.3. Nas Seções 1.4 e 1.5 são apresentadas o sistema de simulação de *blockchains* UniBlock e as considerações finais do minicurso, respectivamente.

1.1. Blockchain

O *blockchain* é um registro de informações distribuído formado por uma cadeia de blocos de dados, conectados uns aos outros por um sistema que utiliza funções *hash* criptográficas. A Figura 1.1 apresenta o resultado da aplicação da função *hash* criptográfica SHA-256 (Secure Hash Algorithm-2) [NIST 2018] em duas entradas distintas. Como pode ser observado, a saída da função é completamente diferente para as duas entradas aparentemente idênticas. A utilização desse tipo de função é essencial para garantir a segurança e a integridade dos dados presentes no *blockchain*.

Figura 1.1. Aplicação de uma função *hash* criptográfica.



Funções *hash* tradicionais transformam dados de entrada de tamanho arbitrário em uma saída de tamanho fixo, chamada de *digest* ou *hash*. Funções *hash* criptográficas são um tipo especial de funções *hash* que devem possuir as seguintes propriedades:

(p_1) Um mesmo dado de entrada da função deve sempre retornar a mesma saída.

(p_2) Computar um *digest* para uma determinada entrada deve ser computacionalmente fácil, i.e., a operação pode ser realizada em menos de um segundo.

(p_3) Descobrir quais dados foram utilizados como entrada da função, analisando somente o *digest*, deve ser computacionalmente muito difícil. Uma tentativa por força bruta para descobrir os dados de entrada deve necessitar de vários anos considerando as capacidades atuais de processamento.

(p_4) Encontrar duas entradas que gerem o mesmo *digest* deve ser computacionalmente muito difícil, necessitando de vários anos de processamento.

(p_5) Uma pequena mudança na entrada deve alterar o *digest* resultante de tal forma que não seja possível encontrar alguma co-relação entre as saídas geradas pelo dado original e pelo dado alterado.

Este tipo de função é utilizado para ajudar a garantir a integridade de informações, já que qualquer mudança nos dados de entrada (e.g., um único bit, um único byte) altera o *digest* resultante, como pode ser observado na Figura 1.1. Ao aplicar uma função *hash* criptográfica em um arquivo e enviar o *digest* para a pessoa que irá recebê-lo, o receptor poderá aplicar a função novamente sobre o arquivo e verificar se o resultado é igual ao *digest* recebido. Dessa forma, ela garante que os dados não foram modificados, desde que o *digest* tenha sido recebido de forma segura.

Por exemplo, suponha que João trabalhe no setor financeiro de uma empresa e que tenha sido encarregado de realizar uma transferência de dinheiro da conta da empresa para a conta de algumas empresas parceiras. João recebeu de Maria o arquivo contendo os dados das contas bancárias por email. Um usuário malicioso, realizando um ataque de interceptação de dados, pode alterar o documento contido no email antes que ele seja recebido por João, adicionando novas contas bancárias na lista, por exemplo. Para certificar-se de que João receberá o arquivo com as mesmas informações enviadas originalmente, Maria computa o *digest* do documento anexado no email. Considere que a empresa utiliza um canal seguro para comunicação auxiliar, para o envio de pequenas quantidades de dados como os *digests*. Maria envia o *digest* do arquivo para João utilizando o canal de comunicação seguro. Ao fazer o *download* do arquivo com os dados das contas, João precisa certificar-se de que nada foi modificado. Para isso, ele computa o *digest* do arquivo recebido e compara-o com o *digest* enviado por Maria. Se os *digests* forem iguais, o documento não foi modificado.

No *blockchain*, a conexão lógica entre os blocos de dados é estabelecida e garantida através de *digests* de uma função *hash* criptográfica. Cada bloco na cadeia aponta para um bloco anterior identificado por um *digest* único (*digest* do bloco). Isso fornece uma propriedade interessante para o registro de transações: quando um bloco é adicionado ao final da cadeia de informações, torna-se uma tarefa muito difícil alterá-lo. Suponha a existência de um *blockchain* para uma aplicação bancária fictícia. Cada bloco da aplicação armazena a informação de uma transação entre duas contas. Um exemplo de um conjunto de dados armazenados em cada bloco de um *blockchain* fictício voltado para o armazenamento de transações financeiras é:

(d_1) *Versão*: Versão do sistema utilizada.

(d_2) *Timestamp*: Representação da data e hora de criação do bloco.

(d_3) *Nonce*: Número auxiliar utilizado para realização do cálculo da função *hash* criptográfica.

(d_4) *Digest* do bloco anterior: Resultado da aplicação de uma função *hash* criptográfica sobre os dados do bloco anterior.

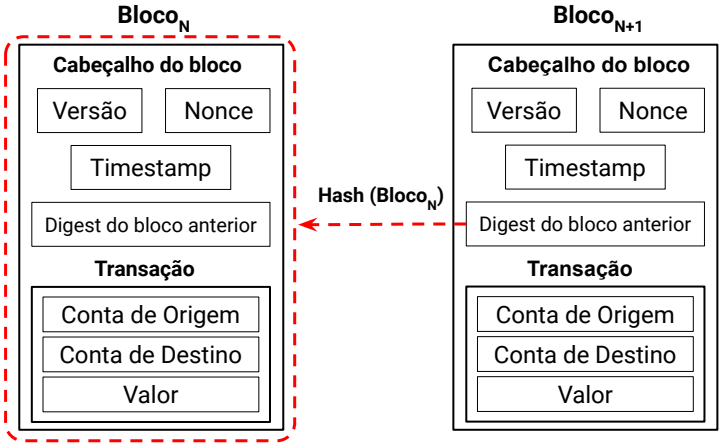
(d_5) *Conta de origem*: A conta de onde o dinheiro será retirado.

(d_6) *Conta de destino*: A conta que receberá o dinheiro.

(d_7) *Valor da transação*: Quantia de dinheiro a ser transferida.

A Figura 1.2 ilustra a conexão entre uma cadeia de blocos. Considerando a estrutura de bloco apresentada, a informação que garante a ordem correta dos blocos é o ponteiro para o *digest* do bloco anterior na cadeia. A utilização de uma função *hash* criptográfica nos dados de cada bloco gera um *digest* único, permitindo estabelecer um vínculo forte e único entre os dados. Isso garante que exista uma única sequência válida de blocos.

Figura 1.2. Representação dos blocos em um *blockchain*.



Imagine que um atacante deseja modificar o conteúdo de um bloco para fins maliciosos, como direcionar o valor de uma transação para si mesmo. Para isso, será necessário que os novos dados do bloco gerem um *digest* igual ao anterior; caso contrário, a ligação entre a cadeia de blocos será desfeita. Se a cadeia de blocos for desfeita, o *blockchain* ficará em um estado inconsistente.

Como as funções *hash* criptográficas garantem que a probabilidade de gerar o mesmo *digest* a partir de dados de entrada diferentes é extremamente baixa, seria mais fácil alterar também o campo que aponta para o *digest* do bloco anterior nos blocos seguintes. Bastaria que o atacante mudasse os campos nos blocos posteriores, calculasse seus *digests* e repetisse o processo alterando os ponteiros até o final do *blockchain*. No entanto, um dos sistemas utilizado para controlar a criação de novos blocos, chamado de Prova de Trabalho, do inglês *Proof-of-Work* (PoW), evita que isso aconteça. Através desse sistema, não basta apenas calcular o *digest* do bloco para que ele seja adicionado à cadeia. O resultado da função *hash* deve obedecer a uma restrição que requer um grande esforço computacional para ser atendida. A restrição adotada na prática pela criptomoeda Bitcoin é encontrar um bloco cujo *digest* resultante possua os primeiros n bits iguais a zero, onde n depende da dificuldade de mineração determinada pelo sistema. Para variar a saída da função *hash* e encontrar um bloco que atenda a restrição é necessário alterar

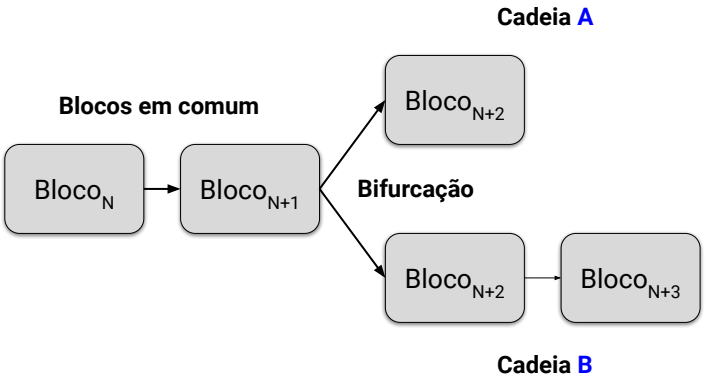
os dados de entrada, para esta finalidade há em cada bloco um campo chamado de *nonce*, que é alterado até que o resultado desejado seja obtido.

Um usuário que deseja criar um *digest* válido altera os dados do campo *nonce* até que a execução da função *hash* gere o resultado desejado. Como não é possível prever o resultado da aplicação de uma função *hash* criptográfica, para cumprir o desafio, quem deseja criar um bloco válido deve tentar valores diferentes no campo *nonce* até que o *digest* resultante do bloco atenda às exigências do sistema. Após descoberto o *nonce* que torna o bloco válido, calcular novamente o *digest* do bloco torna-se trivial. Dois blocos diferentes podem possuir o mesmo dado no campo *nonce*, porém, os dados de transações não podem ser iguais. Isso faz com que *digests* iguais não possam ser gerados a partir de blocos diferentes.

A descentralização do *blockchain* também dificulta a execução do ataque de modificação de blocos, pois alterações na cadeia de blocos implicam em mudar todas as outras cópias do *blockchain*, armazenadas por outros usuários. Para que um atacante possa controlar a criação de novos blocos, ele deve possuir mais de 50% do poder computacional de toda a rede. Somente assim seria possível criar blocos válidos com uma velocidade maior do que o resto dos usuários. Além do PoW, foram propostos outros esquemas de criação de blocos, como o *Proof-of-Stake* (PoS), o *Proof-of-Activity* (PoA) e o *Proof-of-Publication* (PoP) [Tschorsch and Scheuermann 2016].

Eventualmente, pode ocorrer uma bifurcação no *blockchain* quando novos blocos são adicionados, causada pelo tempo necessário para a propagação dos blocos na rede. A Figura 1.3 ilustra o estado da cadeia de blocos quando ocorre uma bifurcação. O problema ocorre quando um usuário *A* cria um um bloco válido ao mesmo tempo em que um usuário *B* cria outro bloco válido contendo dados diferentes. Como é necessária uma quantia de tempo para que a informação seja propagada aos outros participantes da rede, aqueles que recebem primeiro o bloco de *A*, o colocam no fim de suas cadeias, enquanto que aqueles que recebem primeiro o bloco de *B*, colocam um bloco no final de suas cadeias diferente dos primeiros. Isso causa uma divergência momentânea no *blockchain*. A partir desse momento existem duas cadeias cuja única diferença são os últimos blocos. Não é possível saber qual das cadeias é a correta.

Figura 1.3. Bifurcação na cadeia de blocos.



Para resolver o problema da bifurcação na cadeia de blocos, alguns sistemas em-

pregam uma estratégia chamada de "a regra da cadeia mais longa". Com o passar do tempo, naturalmente outros mineradores irão adicionar novos blocos ao final de suas próprias cópias do *blockchain* e irão propagá-los na rede. O sistema irá selecionar a cadeia com o maior número de blocos e a tornará definitiva. As cadeias restantes serão descartadas e as transações contidas em seus blocos voltarão para o conjunto de transações que estão aguardando para serem confirmadas. Devido à esse tipo de ocorrência, em várias criptomoedas é recomendado que os usuários aguardem até que mais blocos sejam adicionados após o bloco onde sua própria transação foi validada. Isso ajuda a garantir que a transação não seja desfeita.

A tecnologia de *blockchain* está se popularizando e sua aplicação como uma estrutura de armazenamento de dados vem sendo investigada em uma grande gama de sistemas em diversas áreas [IEEE 2017]. *Blockchains* podem (potencialmente) substituir plataformas digitais em diferentes setores, além da área de finanças. Porém, o investimento em pesquisa e desenvolvimento ainda é embrionário e necessário para acelerar a criação e comercialização de soluções baseadas nessa tecnologia.

1.2. Criptomoedas

A adoção de criptomoedas como forma de pagamento vem crescendo rapidamente devido aos benefícios que elas oferecem em relação às formas de pagamento tradicionais [medfar87 2018]. Um usuário pode efetuar um pagamento para um receptor em qualquer lugar do mundo a qualquer momento, sem a necessidade de uma instituição intermediária, o que leva a menores taxas de transações, maior controle, e mais privacidade. É importante notar que muitos dos benefícios oferecidos pelas criptomoedas em relação às instituições financeiras tradicionais, como descentralização e maior privacidade, são possibilitados pela utilização da tecnologia de *blockchains*. Entretanto, apesar dos benefícios oferecidos por esse tipo de moeda, existem desvantagens como a impossibilidade de reverter transações e de obter suporte caso ocorram erros no sistema. A volatilidade dos preços da moedas é outro fator negativo. No caso da Bitcoin, segundo estatísticas de mercado, o preço pode variar mais de 10% em poucas horas [Adkisson 2018].

1.2.1. Mineração

Para obter criptomoedas, um usuário pode realizar uma compra através de serviços especializados em vendas de criptomoedas, chamados de *cryptocurrency exchanges*. Algumas moedas, como a Bitcoin e a Monero, oferecem aos usuários a possibilidade de obtê-las diretamente através do sistema, participando de um processo comumente chamado de mineração. É importante notar que existem criptomoedas como a Ripple [Schwartz et al. 2014] e a IOTA [Popov 2014] que não possuem um sistema através do qual os usuários possam obter moedas ao criar blocos de transações, ou seja, não podem ser mineradas. Neste tutorial, iremos focar a discussão em processos de mineração similares aos que ocorrem em criptomoedas como a Bitcoin e a Monero.

O processo de mineração consiste em participar da criação de blocos válidos através do esquema de PoW, PoS ou outro similar, conforme determinado pelo respectivo sistema. Esse processo é essencial para garantir o funcionamento do *blockchain* e da moeda digital, pois é através dele que as transações são confirmadas e adicionadas ao final

da cadeia de blocos.

Mineradores são participantes da rede que utilizam sua capacidade computacional para tentar computar *digests* válidos para os blocos e adicioná-los ao final do *blockchain*. Quando um usuário efetua uma transação, seus dados são compartilhados na rede para que mineradores possam validá-la. Para cada transação efetuada, os usuários devem incluir uma taxa como pagamento para recompensar os mineradores. Transações que pagam taxas maiores são geralmente escolhidas primeiro e são incluídas em blocos mais rapidamente. Cada minerador escolhe as transações (disponíveis na rede) que irão fazer parte do bloco. Em seguida, inicia o processo de encontrar um *digest* que atenda às restrições do sistema.

Devido ao esforço computacional necessário para a criação de blocos válidos pelo esquema de PoW, é necessário um incentivo para que os participantes da rede, ou mineradores, emprestem o seu poder de processamento para ajudar no funcionamento do sistema. Esse incentivo é dado através de uma recompensa em criptomoedas para o participante da rede que conseguir validar primeiro um bloco de transações. Quando um minerador valida um bloco, sua informação é disseminada na rede para que os outros mineradores atualizem suas cópias do *blockchain* e escolham novas transações para validar.

A primeira transação de cada bloco, chamada de *coinbase transaction*, é um tipo especial de transação cuja finalidade é enviar o valor da recompensa para o usuário que efetuou a validação do bloco. O sistema de recompensas é geralmente desenvolvido de maneira que, com o passar do tempo, a recompensa por bloco diminua. Isto é necessário para controlar a quantidade de novas moedas criadas devido ao aumento no preço da moeda e outros fatores econômicos. Na Bitcoin, essa diminuição ocorre a cada 210.000 blocos minerados, quando a recompensa é reduzida pela metade.

A diminuição da recompensa estende a vida do sistema ao impedir que todo o suprimento de moedas seja emitido em um curto período de tempo, o que acabaria com a motivação para a criação de novos blocos válidos. A redução da recompensa também contribui para a valorização da moeda, que passa a valer mais quando a procura aumenta e a oferta diminui. Em um momento no futuro, a validação de novos blocos não irá gerar mais recompensas e o pagamento pela criação de blocos válidos será feito somente através das taxas de transações, pagas pelos usuários. Atualmente, em 2019, a quantia recompensada por bloco na Bitcoin é de 12,5 unidades da moeda, chamada de BTC.

A Tabela 1.1 mostra a mudança na recompensa por bloco válido criado ao longo do tempo. O período estimado para que ocorra a criação de 210.000 novos blocos e ocorra uma diminuição no valor da recompensa por bloco é de 4 anos. Na prática este período pode ser maior ou menor, apesar do aumento no número de mineradores ao longo do tempo. Esta oscilação ocorre porque o sistema eleva automaticamente a dificuldade de criação de novos blocos quando a velocidade da rede aumenta. Esta estratégia é empregada para manter o tempo entre a criação de novos blocos por volta de dez minutos. Isto também evita a emissão de muitas moedas novas em um curto período de tempo. Quando a dificuldade de criar blocos aumenta, o retorno pela criação de blocos diminui dado o esforço necessário, causando uma redução no número de mineradores. A redução no número de mineradores diminui a capacidade de criação de blocos da rede, que ajusta a dificuldade de acordo. Essas variações causam mudanças no intervalo de tempo entre a

diminuição da recompensa por bloco.

Tabela 1.1. Recompensa pela validação de blocos na Bitcoin.

Nº de Blocos	Recompensa por bloco	Ano
0	50 BTC	2009
210.000	25 BTC	2012
420.000	12,5 BTC	2016
630.000	6,25 BTC	2020 (estimado)

É comum que mineradores organizem-se em grupos que trabalham em conjunto para criarem blocos válidos, chamados de *mining pools*. Quando um usuário cria um bloco válido, a recompensa é dividida entre todos os participantes do grupo. Este método diminui o valor da recompensa individual de cada minerador, porém garante um fluxo mais estável de renda para todos. A participação em *mining pools* é interessante pelo fato de aumentar a probabilidade de retorno financeiro uma vez que é difícil um único usuário, sozinho, competir com os demais e as *mining pools*.

1.2.2. Hardware especializado para mineração

A mineração de blocos pode ser uma atividade lucrativa se um usuário conseguir vencer a corrida usando recursos computacionais a um custo menor do que a recompensa. Considerando a cotação atual da criptomoeda Bitcoin (43.520,00 reais em 05 de setembro de 2019), a recompensa de 12,5 BTC pela criação de um bloco é equivalente a 544.000,00 reais.

Com o objetivo de aumentar o lucro proveniente da mineração de criptomoedas, algumas empresas desenvolveram hardwares específicos para computar PoW da Bitcoin, chamados de circuitos integrados de aplicação específica, do inglês *Application Specific Integrated Circuit*s (sASICs). Esses equipamentos são projetados especificamente para computar *digests* de blocos em uma taxa muito superior àquela alcançável em hardware comum. Alguns dos ASICs mais poderosos, destinados ao sistema Bitcoin, como o *ANTMINER S9 Hydro* da fabricante Bitmain, possuem uma capacidade de processamento de até 18.000.000.000 (18 trilhões) de *hashes* por segundo, enquanto um processador Intel Core i7-3930k consegue computar apenas cerca de 98.000 *hashes* por segundo [Cointopper 2018].

O uso de ASICs permite que usuários monopolizem a criação de novos blocos, especialmente quando vários utilizadores desses dispositivos cooperam em *mining pools*. Algumas *Graphics Processing Unit*s (sGPUs) também são utilizadas por mineradores para computar *hashes* por serem mais rápidas do que processadores comuns, porém, sua capacidade também é muito inferior aos *hardwares* especializados. A popularização de ASICs impede que usuários obtenham lucro através do processo de PoW do Bitcoin a menos que invistam na aquisição de equipamentos especializados [Jefferys 2018].

O sistema Monero utiliza um algoritmo resistente à ASIC, chamado de CryptoNight, no seu processo de PoW. Este algoritmo é usado para tornar o processo de

mineração mais igualitário do que em outras criptomoedas. O algoritmo CryptoNight pertence a uma classe de funções denominada *memory-bound*. Nessa classe de função, o tempo necessário para resolver um problema computacional depende principalmente da quantidade e da velocidade da memória disponível para armazenar os dados utilizados durante a sua resolução. A utilização do algoritmo CryptoNight combate o uso de ASICs, que tornaram-se praticamente essenciais para usuários que desejam participar do processo de PoW da Bitcoin.

1.2.3. Mineração através de navegadores

A idéia de inserir códigos de mineração de criptomoedas em páginas da *Web* surgiu logo após o lançamento da Bitcoin [Eskandari et al. 2018]. Pouco tempo após a proliferação da idéia, várias aplicações de mineração desenvolvidas com a linguagem de programação JavaScript tornaram-se populares como: JSMiner[4], MineCrunch[5] e Tidbit[6]. O processo de mineração utilizando navegadores foi divulgado como uma alternativa à exibição de propagandas para a monetização de conteúdos em páginas da *Web*. Ao visitar uma página que contém uma aplicação de mineração, os recursos computacionais do usuário são utilizados para computar *digests* de blocos de transações, uma etapa essencial na criação de blocos e obtenção de recompensas em sistemas de criptomoedas.

Algumas páginas *Web* defendem o uso benigno de mineradores baseados em código JavaScript para a geração de lucro e manutenção de sua infraestrutura [Saad et al. 2018], solicitando que os usuários emprestem seu poder computacional em troca de acesso ao conteúdo da página. Entretanto, o uso dessas aplicações por usuários maliciosos difundiu-se rapidamente devido à facilidade de execução de ataques, necessitando apenas que as vítimas possuam uma conexão com a Internet e visitem uma página da *Web* infectada com o código do minerador desenvolvido em JavaScript. Após a difusão inicial da idéia e o surgimento de diversas aplicações de mineração via navegadores, desenvolvedores e páginas da *Web* que compactuavam com o uso desse tipo de aplicações passaram a enfrentar problemas judiciais devido ao uso não autorizado dos recursos computacionais dos usuários [Blattberg 2014].

No ano de 2017, anos após a primeira onda de aplicações de mineração codificadas em JavaScript, as atividades de mineração em navegadores aumentaram expressivamente devido ao lançamento de um novo serviço de mineração de criptomoedas em plataformas *Web*, chamado de Coinhive [Rauchberger et al. 2018]. O código de mineração do serviço Coinhive chegou a estar presente em cerca de 32.000 *websites* distintos em 2017 [Krebs 2018]. Desta vez, o foco da mineração deixou de ser a Bitcoin e voltou-se para a criptomoeda Monero. Um dos principais fatores que contribuiram para a escolha da Monero foi o seu algoritmo de PoW, CryptoNight, projetado para permitir que computadores com processadores de propósito geral sejam adequados para participar do processo de criação de blocos. Ao permitir que os processadores presentes em computadores comuns sejam capazes de calcular *hashes* de blocos de maneira mais eficiente do que *hardwares* especializados, o algoritmo CryptoNight torna viável a mineração de Monero através de códigos embutidos em páginas *Web*.

[4] `https://github.com/jwhitehorn/jsMiner`
[5] `https://www.technicpack.net/modpack/minecrunch.814457/changelog`
[6] `https://en.wikipedia.org/wiki/Tidbit`

O serviço Coinhive despertou novamente o interesse de atacantes. A mineração de criptomoedas através de navegadores sem o consentimento dos usuários (*in-browser cryptojacking*) tornou-se novamente uma forma de ataque proeminente. Ataques de *cryptojacking* consistem na utilização dos recursos computacionais de uma ou mais vítimas, sem o seu consentimento, para a realização do processamento necessário à criação de blocos válidos em criptomoedas. No ano de 2017, o aumento na detecção de ataques de *in-browser cryptojacking* foi de 8.500% [Mathur 2018]. Em junho do ano de 2018, a plataforma Coinhive foi responsável por 1,18% dos blocos minerados no sistema Monero [Rüth et al. 2018].

Estima-se que 10 milhões de usuários sejam afetados por ataques de *in-browser cryptojacking* a cada mês [Hong et al. 2018]. Páginas da *Web* nas quais os usuários permanecem por longos períodos de tempo como plataformas de *streaming* de vídeo, são alvos ideais para a injeção de scripts de *cryptojacking*, visto que o lucro obtido através de mineração é proporcional ao tempo que os usuários permanecem em uma página infectada.

A execução de um ataque de *in-browser cryptojacking* geralmente depende de dois componentes distintos: um código controlador e um código de mineração. A Figura 1.4 ilustra o processo de comunicação necessário entre a máquina do usuário e os servidores do atacante para iniciar ao processo de mineração. A primeira etapa da comunicação ocorre quando um usuário visita uma página que executa um código de mineração. Ao comunicar-se pela primeira vez com a página, o navegador do usuário recebe e executa um código Javascript que inspeciona os recursos computacionais disponíveis em sua máquina. O script também verifica se o navegador da vítima suporta a execução de código WebAssembly (Wasm).

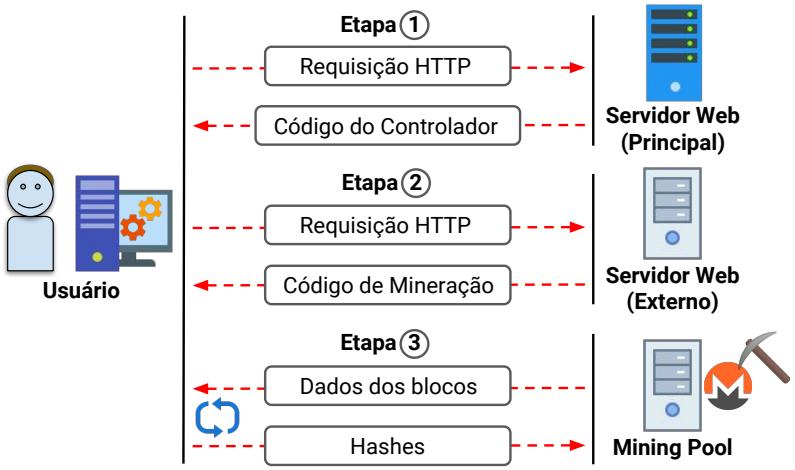
O formato de instruções Wasm é otimizado para permitir a execução de código em navegadores com um desempenho próximo ao de uma aplicação sendo executada nativamente na máquina[7]. Na próxima etapa, o script comunica-se com um servidor externo definido pelo atacante, isto é, um servidor diferente do qual o usuário acessou. Se o navegador suportar a execução de código Wasm, o script do controlador faz o download de um código Wasm que é compilado na máquina da vítima. Caso contrário, é realizado o download de instruções `asm.js`, um tipo de código JavaScript focado em desempenho.

Na última etapa, o código de mineração cria *threads* (processos leves) na máquina da vítima de acordo com a quantidade de recursos disponíveis, cria uma conexão com um serviço de *mining pool* e requisita tarefas de processamento. Ao terminar as tarefas recebidas, a máquina envia para o servidor da *mining pool* os *digests* computados e repete a última etapa até o encerramento da conexão.

Embora sejam amplamente utilizadas, estima-se que plataformas de mineração como Coinhive não gerem um retorno monetário tão expressivo quanto a exibição de propagandas em páginas da *Web* [Saad et al. 2018]. A menos que as páginas utilizadoras de mineração em navegadores atraiam muitos usuários e ofereçam conteúdos que os mantenham no *site* por longos períodos de tempo, a utilização desse tipo de serviço não é recomendado.

[7] `https://webassembly.org/`

Figura 1.4. Diagrama ilustrando as etapas um ataque de *in-browser cryptojacking*.



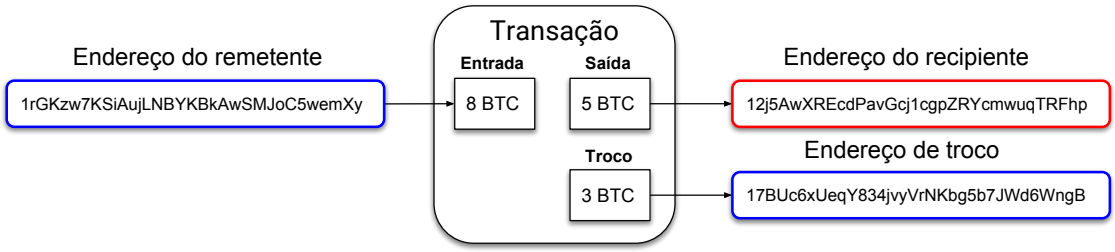
1.3. A criptomoeda Bitcoin

Introduzida através de um *white paper* em uma lista de correio eletrônico sobre criptografia em 2008 e lançada em 2009, Bitcoin foi a primeira criptomoeda de sucesso e utilizada em larga escala [Nakamoto 2008]. Anos após seu lançamento, ainda permanece sendo a mais utilizada, possuindo um valor aproximado de mercado de US$ 188 bilhões[8] em 05 de setembro de 2019. A Bitcoin baseia-se em material de pesquisas anteriores, como o esquema de PoW para controlar a criação de blocos válidos no *blockchain*, esquemas de assinaturas digitais para garantir que os usuários que utilizam moedas realmente as possuem e técnicas de *timestamping* que marcam a data e a hora da realização das operações. A principal contribuição da Bitcoin foi eliminar a necessidade de uma autoridade central que regule a emissão de moedas e a confirmação de transações. Isso foi possível pela forma descentralizada pela qual o sistema funciona, utilizando uma rede ponto-a-ponto onde usuários participam do processo de validação e verificação da autenticidade das transações. A descentralização também é fruto da maneira como os dados estão armazenados. Todas as transações ocorridas estão armazenadas em um *blockchain* público, isto é, cada participante da rede pode optar por obter uma cópia desse registro. Usuários que não desejam realizar o download dos dados do *blockchain* da criptomoeda podem acessá-los através de clientes leves, do inglês *Light-Clients*. Clientes leves são programas que permitem aos usuários acessar dados através de uma conexão com um nó remoto confiável que mantém uma cópia do *blockchain*.

Para enviar e receber transações em Bitcoin, um usuário necessita de um par de chaves criptográficas composto por uma chave pública e uma chave privada. No caso da Bitcoin, a chave pública é utilizada como endereço de envio e recebimento de pagamentos. Um usuário pode divulgar a sua chave pública e outras pessoas podem enviar Bitcoins para esse endereço. A chave privada é utilizada para comprovar que um usuário é dono da chave pública que a acompanha, podendo assim utilizar os fundos recebidos. A chave privada não deve ser compartilhada e deve ser armazenada em segurança para

[8] `https://coinmarketcap.com/`

Figura 1.5. Transação de Bitcoins contendo uma única entrada.

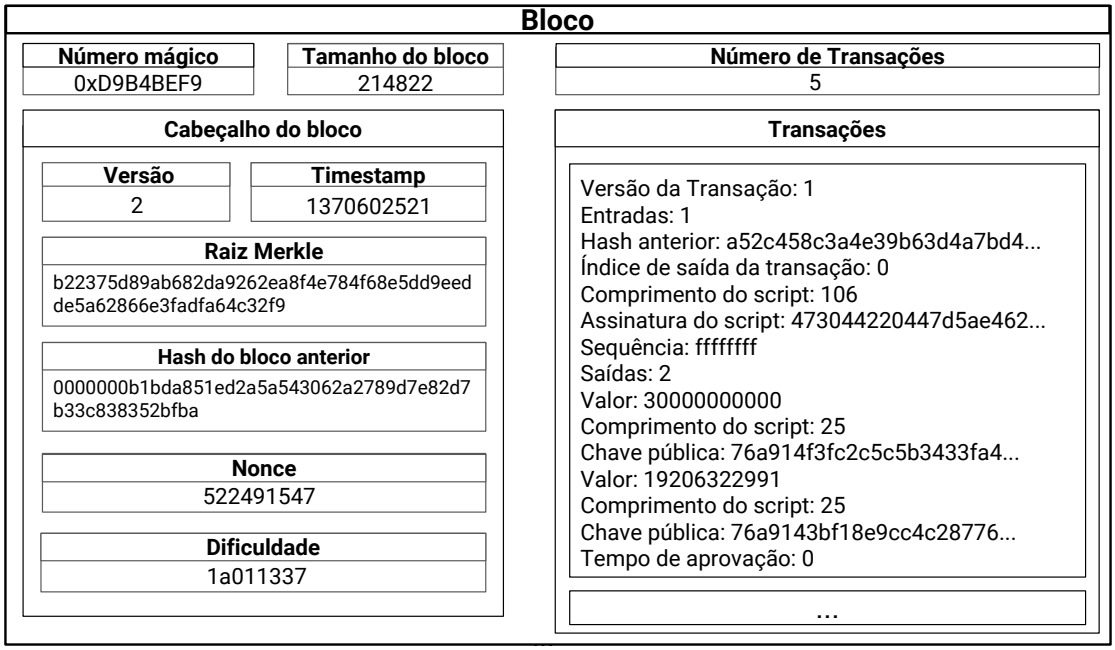


que ninguém além do proprietário possa acessá-la. O endereço de um usuário é derivado de sua chave pública, ao aplicar o algoritmo de *hashing* SHA-256 e depois o algoritmo RIPEMD-160, adicionar números para controle de erro e controle de versões e, por fim, codificá-lo em BASE58 [Tschorsch and Scheuermann 2016]. O processo de derivação da chave é realizado para fornecer segurança adicional, ajudando a ocultar a verdadeira chave pública. Os usuários também podem optar pela utilização de sua chave pública original como endereço.

Em uma transação, existem endereços de entrada e endereços de saída. A Figura 1.5 ilustra o procedimento de uma transação de Bitcoins contendo uma única entrada. O endereço do remetente é a chave pública que corresponde ao endereço da carteira de Bitcoins do emissor do pagamento. O endereço do recipiente é o endereço da carteira do receptor do pagamento. O endereço de troco deve ser definido pelo emissor do pagamento para que ele receba o troco da operação, caso a chave de entrada utilizada exceda o valor do pagamento.

Só podem ser utilizadas como entradas de transações as chaves que foram geradas como saída em uma transação anterior. Endereços de saída de transações são chaves recebidas pelos usuários que recebem as criptomoedas. O saldo de um usuário da Bitcoin consiste na soma dos valores de todas as saídas de transações que ele já recebeu e ainda não utilizou. Antes de serem utilizadas, as chaves que contêm criptomoedas recebem a denominação de "saída de transação não-utilizada", do inglês *Unspent Transaction Output* (UTXO). Sempre que uma chave de entrada é utilizada em uma transação, todo o seu conteúdo em Bitcoins deve ser gasto, ou seja, não é possível usar somente parte da quantia armazenada em um endereço. Após uma UTXO ser utilizada, seu estado muda para "chave de saída utilizada", do inglês *Spent Transaction Output* (STXO), para indicar que a quantia armazenada nesta chave já foi gasta. Para permitir que os remetentes mantenham o dinheiro que sobra do pagamento, existe a idéia de troco, onde o pagamento é feito para o recipiente e o restante é enviado para um endereço de escolha do remetente. O endereço de troco pode ser o mesmo endereço utilizado como entrada, mas isso é desencorajado porque quanto mais um endereço é utilizado, mais fácil torna-se o processo de rastrear informações do usuário. É recomendado que os usuários utilizem uma carteira de Bitcoins, um tipo de programa que auxilia no gerenciamento das chaves e endereços ao criar automaticamente novos endereços para o recebimento de troco. Uma carteira é capaz de gerenciar diferentes endereços de um mesmo usuário.

Figura 1.6. Estrutura de um bloco do *blockchain* do sistema Bitcoin.



Um único bloco é capaz de armazenar milhares de transações, desde que os dados de todas elas somados não ultrapassem o tamanho de 1 MB. A Figura 1.6 resume os dados contidos um bloco do sistema Bitcoin. Os blocos criados para armazenar as transações no *blockchain* possuem os seguintes campos:

(c_1) *Número mágico*: Valor utilizado para identificar o tipo de estrutura contida nos dados. Neste caso, um bloco. Este valor é específico do protocolo Bitcoin.

(c_2) *Tamanho do bloco*: Especifica o tamanho em *bytes* dos dados contidos no bloco.

(c_3) *Cabeçalho do bloco*: Contém dados que identificam o bloco atual.

(c_4) *Versão*: Especifica a versão do sistema no momento da criação do bloco.

(c_5) *Timestamp*: Representa o momento no tempo em que o bloco foi criado.

(c_6) *Raiz Merkle*: Um tipo de *hash* utilizado para verificar a validade das transações contidas nos blocos sem a necessidade de verificar todas as informações das transações. Para realizar a verificação é necessário o cabeçalho dos blocos e uma estrutura chamada de Árvore de Merkle.

(c_7) *Nonce*: Campo cujo valor deve ser modificado até que o *digest* resultante do bloco atenda as exigências do sistema.

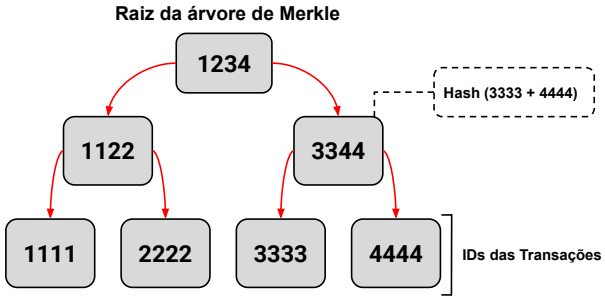
(c_8) *Dificuldade*: Especifica o número de *bits* 0 necessários à esquerda do *digest* do bloco para que ele atenda às exigências do sistema.

(c_9) *Número de transações*: Contém o número de transações presentes no bloco atual.

(c_{10}) *Transações*: Contém os dados de cada uma das transações contidas no bloco.

É interessante ressaltar a importância do campo que armazena a raiz da Árvore de Merkle. A Figura 1.7 ilustra a estrutura de uma Árvore de Merkle. Os valores contidos nos nós da árvore foram simplificados para fins didáticos. Em uma estrutura real,

Figura 1.7. Representação de uma Árvore de Merkle.



as informações armazenadas dentro dos nós de uma Árvore de Merkle são os *digests* gerados pela função SHA-256. A raiz de Merkle funciona como um identificador para as transações que estão contidas em um bloco.

Imagine que para cada bloco criado seja computado um resumo (ou *digest*) dos identificadores de todas as transações nele contidas. Para verificar se todas as transações contidas no bloco foram incluídas no *digest* (para verificar a integridade dos dados), seria necessário reunir os identificadores de todas as transações e computar o *digest* novamente. Em uma Árvore de Merkle, cada nó da estrutura armazena o *digest* dos dados contidos em seus nós filhos. Os nós folha da árvore armazenam os dados originais, no caso da Bitcoin, os identificadores das transações contidas no bloco. Com essa estrutura é possível verificar a integridade das transações presentes em um bloco através dos *digests* que resumem os dados, acelerando o processo.

Considerando a estrutura apresentada na Figura 1.7, suponha que um usuário foi informado de que recebeu Bitcoins através da transação com o identificador 4444, em um bloco cuja raiz de Merkle é 1234. O receptor deseja verificar se esta transação está de fato presente no bloco. O usuário já conhece o valor da raiz da árvore e do identificador de sua transação. Para realizar a verificação, serão necessários também os valores 3333 e 1122. Primeiro, é computado o *digest* dos valores 3333 e 4444 e é gerado o valor 3344, que resume os dados das folhas do lado direito da árvore. O valor que resume os dados das folhas do lado esquerdo, 1122, já é conhecido. Na sequência é gerado um *digest* a partir dos valores que resumem os dois lados da árvore, 1122 e 3344, gerando a raiz da Árvore de Merkle, que é igual a 1234 e resume ambos os lados da árvore. Se a raiz de Merkle resultante é igual ao valor contido dentro do bloco no *blockchain*, isto significa que a transação 4444 está contida nas transações do bloco examinado.

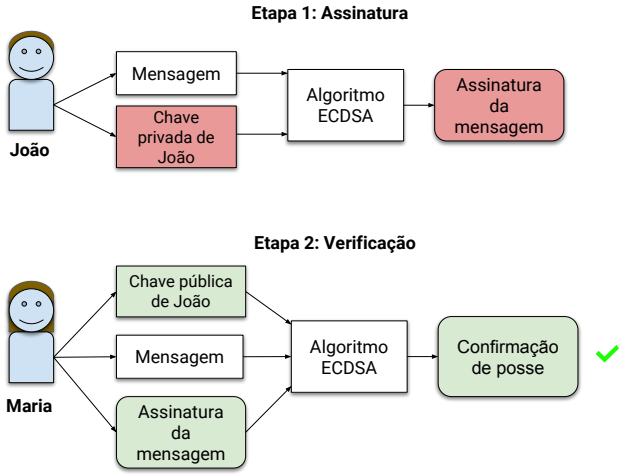
Em blocos com poucas transações, a diferença entre o número de dados necessários para realizar a verificação e o número de dados no bloco é pequena. Porém, conforme o número de transações aumenta para centenas ou milhares, o número de operações com uma Árvore de Merkle torna-se muito menor do que o número de transações necessárias para verificar o *digest*, tornando evidente o benefício da utilização de Árvores de Merkle.

Como o *blockchain* da Bitcoin é um registro transparente, o histórico de transações de todos os usuários está disponível abertamente. Através da análise dos endereços e do

fluxo de transações é possível efetuar ataques que correlacionam os pseudônimos com as identidades dos usuários [Meiklejohn et al. 2013]. Para reduzir o impacto de ataques que efetuam a análise das transações, é sugerida a utilização de uma nova chave e endereço para cada transação [Nakamoto 2008]. A quantidade de Bitcoins associada a cada usuário não é armazenada explicitamente nos registros. O saldo de um endereço pode ser verificado ao checar todo o seu histórico de transações, calculando quantas Bitcoins foram recebidas e quantas foram enviadas a partir do endereço. Por isso, sempre que um usuário instala pela primeira vez uma carteira de Bitcoins em seu sistema, é necessário que ele verifique todas as transações já ocorridas. A verificação é realizada para checar se os pagadores possuem de fato os valores sendo gastos.

Para garantir a segurança das transações na Bitcoin, um sistema de assinaturas baseado no algoritmo ECDSA é utilizado. A Figura 1.8 ilustra o processo de criação de uma assinatura digital. João precisa provar que possui um endereço para enviar dinheiro através dele. Para isso, ele cria uma mensagem contendo os dados da transação que deseja realizar. O segundo passo é criar uma assinatura digital, que servirá para provar que João é o dono do endereço do qual as moedas estão sendo enviadas. Utilizando a sua chave privada, João gera a assinatura digital da mensagem e a envia para a rede juntamente com a mensagem e sua chave pública.

Figura 1.8. Processo de assinatura de uma transação.



Para verificar a validade da transação, Maria realiza uma operação matemática utilizando a chave pública e assinatura enviadas junto com a mensagem. O resultado da operação irá confirmar se a chave pública recebida corresponde à chave privada utilizada na geração da assinatura digital, sem revelar qualquer informação sobre a chave privada. Maria saberá que a chave pública enviada junto com a mensagem, que é o mesmo endereço de onde estão sendo enviadas moedas, pertence à pessoa que tem chave privada correspondente. João é então identificado como o dono do endereço.

Após a criação de uma transação, o remetente dissemina na rede uma mensagem avisando que possui uma nova transação. Os participantes interessados nos dados enviam um pedido explícito ao remetente. Os mineradores acumulam transações e as organizam

em blocos antes de iniciarem o processo de tentativa de criação de um bloco válido. A dificuldade de criação dos blocos é regulada pelo sistema e é alterada com base no tempo que foi necessário para a criação dos últimos 2.016 blocos. O ajuste é realizado com a intenção de manter o tempo necessário para adicionar um bloco a cada dez minutos, para que o tempo de confirmação das transações seja razoável. Levando em consideração o tempo ideal de criação de um bloco, 10 minutos, 2.016 blocos devem ser criados em exatamente duas semanas. Se o tempo necessário para a criação dos últimos 2.016 blocos exceder duas semanas, a dificuldade da criação dos blocos é reduzida e se o tempo for inferior a duas semanas, a dificuldade é elevada. O sistema ajusta a dificuldade da criação de blocos válidos de acordo com a Equação 1.

$$D = D_{anterior} \times \frac{T}{2016 \times 10min} \tag{1}$$

onde:

D = Dificuldade da resolução do problema de criação de um bloco válido.

T = Tempo ocorrido desde a última mudança de dificuldade em minutos.

Ao efetuar a criação de um bloco válido, o minerador dissemina a informação do bloco para que os outros participantes saibam que as transações contidas naquele bloco foram confirmadas por ele. Uma aplicação de carteira Bitcoin realiza uma varredura na cadeia de blocos para verificar as transações destinadas ao seu endereço público e saber quantas moedas o usuário possui. Devido à transparência das informações contidas no *blockchain*, qualquer um com acesso aos dados é capaz de descobrir o saldo de um endereço qualquer, diminuindo a privacidade dos usuários.

Apesar de apresentar alguns problemas de privacidade e de possuir uma capacidade limitada de processar transações devido ao esquema de PoW, a Bitcoin continua sendo amplamente utilizada. Seu sucesso contribui para a criação de novas criptomoedas que buscam solucionar problemas existentes nos sistemas atuais, como a demora das confirmações e a rastreabilidade das transações.

1.4. UniBlock

Para demonstrar na prática o funcionamento de uma estrutura de *blockchain*, de maneira didática, foi desenvolvido um sistema denominado UniBlock. Este sistema permite aos seus usuários executar as suas próprias instâncias de entidades cruciais de uma rede *blockchain*, como os mineradores (*miners*) e negociantes (*traders*). Estas entidades participam do processo de criação de transações e mineração de blocos, conforme ilustrado na Figura 1.9.

O desenvolvimento do UniBlock foi baseado na estrutura de *blockchain* do tutorial "*Learn Blockchains by Building One*"[9]. O tutorial destina-se aos usuários iniciantes, que querem conhecer os princípios básicos da implementação de uma *blockchain*. Entretanto, a implementação da *blockchain* não apresenta uma separação clara entre mineradores e

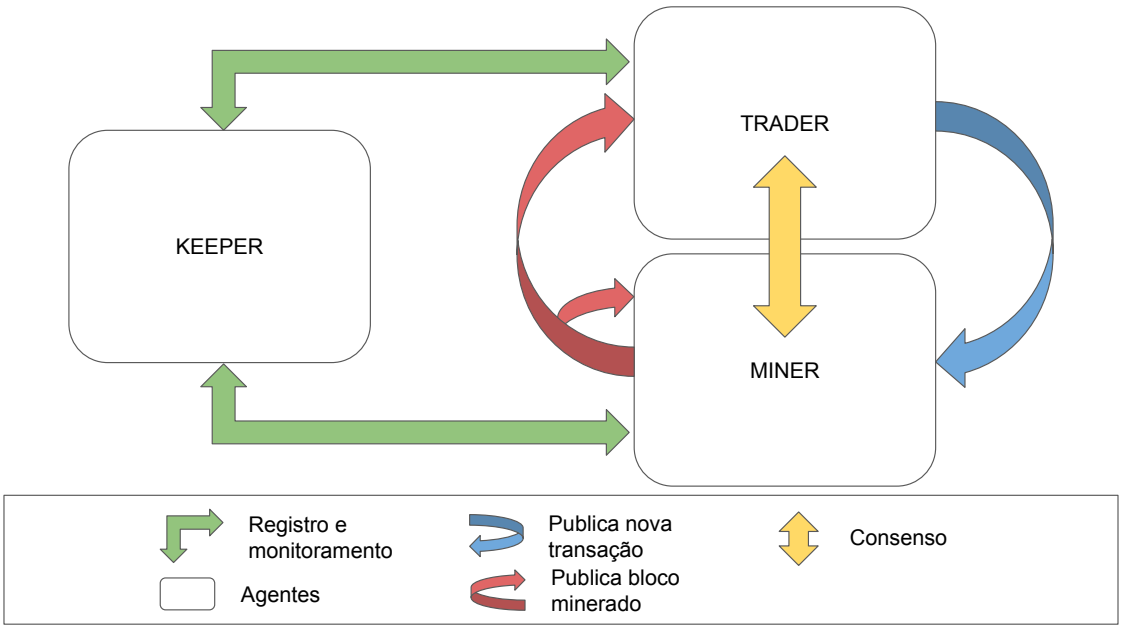
[9] `https://github.com/dvf/blockchain`

negociantes, afetando o entendimento e a compreensão do sistema. Suprir esta deficiência do código apresentado no tutorial foi um dos principais motivadores para a criação do UniBlock[10].

1.4.1. Arquitetura e Funcionamento do UniBlock

A arquitetura do UniBlock pode ser visualizada na Figura 1.9. Como pode ser observado, há três agentes essenciais, o *Keeper*, o *Trader* e o *Miner*. O *Keeper* é o serviço de coordenação das entidades distribuídas da rede da *blockchain*, cuja finalidade e funcionalidades, apesar de simplificadas, são similares as do Apache ZooKeeper[11]. Os *Traders* são responsáveis por gerar as transações que irão alimentar a cadeia de blocos. Por fim, os *Miners* são responsável por minerar os blocos alimentados preenchidos com transações enviadas pelos *Traders*. Vale ressaltar que há somente um único *Keeper* ativo na rede, enquanto que o número de *Miners* e *Traders* pode variar de acordo com o cenário de testes que o usuário pretende criar.

Figura 1.9. Representação da arquitetura do UniBlock.



A rede do UniBlock é iniciada obrigatoriamente com um *Keeper*, que irá manter atualizada a lista de entidades ativas na rede. Em seguida, *Miners* e *Traders* podem ser criados. Cada novo agente irá registrar-se ao *Keeper*. Além disso, cada agente da rede deverá enviar, periodicamente, um *heartbeat* ao *Keeper*, informando que ainda está ativo na rede. Caso o *heartbeat* não seja recebido, o *Keeper* remove o agente da rede. Neste caso, o *Keeper* irá informar todos os agentes ativos sobre a remoção do agente cujo *heartbeat* não foi recebido. Na Figura 1.9, este processo é representado pelas setas verdes de registro e monitoramento.

[10] `https://github.com/homdreen/UniBlock`
[11] https://zookeeper.apache.org/

Os dados utilizados nas transações são alimentados nos *Traders*. Assim que uma nova transação é criada, o *Trader* publica a nova transação aos *Miners* ativos na rede (seta azul da figura). Os *Miners* agrupam as transações e mineram os blocos que serão publicados na rede (setas vermelhas da figura). Finalmente, para ser inserido definitivamente na *chain*, o novo bloco precisa ser aprovado pela maioria (i.e., mais de 50% do agentes ativos), utilizando um protocolo de consenso entre os *Traders* e *Miners* da rede (seta amarela na figura).

1.4.2. Utilização Prática do UniBlock

Todos os agentes do UniBlock são inicializados a partir do mesmo código Python (`main.py`). Primeiro, é necessário inicializar um *Keeper*, passando como parâmetro a opção `--keeper`, como ilustrado a seguir. É necessário passar como parâmetro também o IP e a porta onde o *Keeper* estará aguardando as conexões dos *Traders* e *Miners* da rede.

```
$ python3 main.py -ki <keeperIP> -kp <keeperPorta> --keeper
```

Para adicionar um novo minerador no sistema, é necessário fornecer o IP e a porta do *Keeper*, além de incluir o parâmetro `--miner`, que identifica o agente como sendo um *Miner*. O minerador possui uma interface que permite acessar informações em tempo real, como os usuários ativos no sistema, a carteira atual e os dados da cadeia de blocos. Os comandos da interface estão disponíveis ao usuário através do comando `help`.

```
$ python3 main.py -ki <keeperIP> -kp <keeperPorta> --miner
```

A inicialização de um *trader* é similar à de um minerador, incluindo o IP e porta do *Keeper* e o parâmetro `--trader`, que o identificará como sendo um *Trader*. Assim como o negociador, o minerador oferece um conjunto de comandos, que podem ser visualizados através do comando `help`, para acessar informações em tempo real. No *Trader*, o usuário do sistema pode listar usuários ativos, criar uma nova transação ou listar os dados do *blockchain*.

```
$ python3 main.py -ki <keeperIP> -kp <keeperPorta> --trader
```

1.5. Considerações Finais

Este minicurso apresentou uma introdução às tecnologias de *blockchain* e criptomoedas, desde a teoria necessária para entender essas tecnologias até as estruturas e métodos utilizados. As discussões foram acompanhadas de ilustrações e exemplos com o objetivo de auxiliar a compreensão do leitor. Temas como os métodos de segurança e consenso foram abordados, tendo em vista que estes garantem o correto funcionamento da *blockchain*, um sistema descentralizado, sem a necessidade de um intermediador para que as transações aconteçam. Também foram discutidos os detalhes de funcionamento da criptomoeda Bitcoin, que é a moeda digital mais utilizada atualmente.

No minicurso foi apresentado também o UniBlock, um sistema de *blockchain* com fins didáticos. De uma forma simples e bem estruturada, o UniBlock apresenta os

conceitos, a implementação e a prática com uma rede de negociadores (*Traders*) e mineradores (*Miners*). Com o UniBlock, os usuários podem rapidamente criar, operacionalizar e acompanhar os detalhes de funcionamento de uma *blockchain*.

Referências

[Abbas and Sung-Bong 2019] Abbas, Q. E. and Sung-Bong, J. (2019). A Survey of Blockchain and Its Applications. In *ICAIIC*, pages 001–003.

[Adkisson 2018] Adkisson, J. (2018). Why bitcoin is so volatile. http://tiny.cc/8fp35y.

[Al-Jaroodi and Mohamed 2019] Al-Jaroodi, J. and Mohamed, N. (2019). Blockchain in Industries: A Survey. *IEEE Access*, 7:36500–36515.

[Blattberg 2014] Blattberg, E. (2014). New Jersey slaps MIT Bitcoin hackers with subpoena — and they're fighting back. https://tinyurl.com/y3owapf4.

[Chen et al. 2018] Chen, W., Xu, Z., Shi, S., Zhao, Y., and Zhao, J. (2018). A survey of blockchain applications in different domains. In *Proceedings of the ICBTA*, pages 17–21, New York, NY, USA. ACM. http://doi.acm.org/10.1145/3301403.3301407.

[Cointopper 2018] Cointopper (2018). Difference between asic, gpu and cpu mining. http://tiny.cc/ljp35y.

[Dai et al. 2019] Dai, H., Zheng, Z., and Zhang, Y. (2019). Blockchain for internet of things: A survey. *CoRR*, abs/1906.00245.

[Eskandari et al. 2018] Eskandari, S., Leoutsarakos, A., Mursch, T., and Clark, J. (2018). A first look at browser-based cryptojacking. *arXiv preprint arXiv:1803.02887*. https://arxiv.org/abs/1803.02887.

[Hong et al. 2018] Hong, G., Yang, Z., Yang, S., Zhang, L., Nan, Y., Zhang, Z., Yang, M., Zhang, Y., Qian, Z., and Duan, H. (2018). How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1701–1713. ACM. https://doi.org/10.1145/3243734.3243840.

[Hoy 2017] Hoy, M. B. (2017). An introduction to the blockchain and its implications for libraries and medicine. *Medical reference services quarterly*, 36(3):273–279.

[IEEE 2017] IEEE (2017). Special Report on Blockchain World. *IEEE Spectrum*, 10. http://bit.do/spectrum-blockchain.

[Jefferys 2018] Jefferys, K. (2018). The Problem With ASICs. http://tiny.cc/kqu35y.

[Krebs 2018] Krebs, B. (2018). Who and what is coinhive? https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/.

[Marinoff 2018] Marinoff, N. (2018). South korea is trialing blockchain voting — here's what that means. http://tiny.cc/swp35y.

[Mathur 2018] Mathur, N. (2018). Cybersecurity: Cryptojacking attacks exploded by 8,500% in 2017, says report. https://tinyurl.com/y84alobt.

[medfar87 2018] medfar87 (2018). Cryptocurrency Growth & Adoption Statistics. http://tiny.cc/oxp35y.

[Meiklejohn et al. 2013] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S. (2013). A fistful of bitcoins: characterizing payments among men with no names. In *ACM IMC*, pages 127–140. ACM.

[Min et al. 2019] Min, T., Wang, H., Guo, Y., and Cai, W. (2019). Blockchain Games: A Survey. *CoRR*, abs/1906.05558.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

[NIST 2018] NIST (2018). Hash functions. https://csrc.nist.gov/projects/hash-functions.

[Olnes et al. 2017] Olnes, S., Ubacht, J., and Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3):355 – 364.

[Popov 2014] Popov, S. (2014). The tangle. http://www.tangleblog.com/wp-content/uploads/2016/11/IOTA_Whitepaper.pdf.

[Rauchberger et al. 2018] Rauchberger, J., Schrittwieser, S., Dam, T., Luh, R., Buhov, D., Pötzelsberger, G., and Kim, H. (2018). The other side of the coin: A framework for detecting and analyzing web-based cryptocurrency mining campaigns. In *Proceedings of the 13th Int. Conf. on Availability, Reliability and Security*, page 18. ACM.

[Rüth et al. 2018] Rüth, J., Zimmermann, T., Wolsing, K., and Hohlfeld, O. (2018). Digging into Browser-based Crypto Mining. In *ACM IMC*, pages 70–76. ACM.

[Saad et al. 2018] Saad, M., Khormali, A., and Mohaisen, A. (2018). End-to-End Analysis of In-Browser Cryptojacking. *arXiv preprint arXiv:1809.02152*. https://arxiv.org/abs/1809.02152.

[Schwartz et al. 2014] Schwartz, D., Youngs, N., Britto, A., et al. (2014). The Ripple protocol consensus algorithm. https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[Tschorsch and Scheuermann 2016] Tschorsch, F. and Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123.

[Xie et al. 2019] Xie, J., Tang, H., Huang, T., Yu, F. R., Xie, R., Liu, J., and Liu, Y. (2019). A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. *IEEE Communications Surveys Tutorials*, 21(3):2794–2830.

[Yang et al. 2019] Yang, R., Yu, F. R., Si, P., Yang, Z., and Zhang, Y. (2019). Integrated Blockchain and Edge Computing Systems: A Survey, Some Research Issues and Challenges. *IEEE Communications Surveys Tutorials*, 21(2):1508–1532.

[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.