UNIVERSIDADE FEDERAL FLUMINENSE

VANESSA ROCHA LEANDRO CHICARINO

Uma heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

VANESSA ROCHA LEANDRO CHICARINO

Uma heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação.

Orientador:
CÉLIO VINÍCIUS NEVES DE ALBUQUERQUE

Co-orientador:
ANTÔNIO AUGUSTO DE ARAGÃO ROCHA

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

C532h Chicarino, Vanessa Rocha Leandro
Uma heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain / Vanessa Rocha Leandro Chicarino ; Célio Vinícius Neves de Albuquerque, orientador ; Antonio Augusto de Aragão Rocha, coorientador. Niterói, 2018.
108 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2018.

DOI: http://dx.doi.org/10.22409/PGC.2018.m.09805711706

1. Redes de comunicação de computadores. 2. Produção intelectual.I. Albuquerque, Célio Vinícius Neves de, orientador. II. Rocha, Antonio Augusto de Aragão, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

VANESSA ROCHA LEANDRO CHICARINO

Uma heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação.

Aprovada em abril de 2019.

BANCA EXAMINADORA

Prof. Célio Vinícius Neves de Albuquerque, UFF

Prof. Antonio Augusto de Aragão Rocha, UFF

Prof. José Viterbo Filho, UFF

Prof. Diogo Menezes Ferrazani Mattos, UFF

Prof. Pedro Braconnot Velloso, UFRJ

Niterói

2019

A Deus, autor da minha vida. Seu fôlego de vida em mim me foi sustento e me deu forças durante toda esta longa caminhada.

Agradecimentos

Agradeço primeiramente a Deus, que permitiu que tudo isso acontecesse, ao longo de minha vida, e não apenas nestes anos como mestranda, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Aos meus orientadores Célio e Guto, pela orientação, competência, profissionalismo e dedicação. Obrigada por acreditar em mim e por entenderei os momentos em que estive ausente e por me ajudarem no meu retorno a pesquisa. Tenho certeza que não chegaria neste ponto sem o apoio vocês.

Aos meus amigos do Instituto de Computação e do Laboratório MidiaCom, onde passamos muitas horas de pesquisa e estudo. Obrigada pelo convívio, amizade e apoio. Em especial à amiga Marister, pelo constante apoio e incentivos e pelos precisos suportes administrativos, muito obrigada pelo carinho e cuidado.

Ao meu amigo e parceiro de trabalho Emanuel, obrigada pelos grandes ensinamentos, apoio, pelos trabalhos e disciplinas realizados em conjunto e pela constante preocupação, sempre compartilhando todo o aprendizado obtido comigo. Ao meu Diretor CMG Italo e Vice Diretor CF Welp pelo incentivo e apoio que foram fundamentais para a conclusão da minha pesquisa e aos meus companheiros d trabalhos, especialmente CT Renato e 1T Souto, agradeço a amizade e o carinho que sempre me disponibilizaram. Ao meu Diretor CMG Italo e Vice Diretor CF Welp pelo incentivo e apoio que foram fundamentais para a conclusão da minha pesquisa e aos meus companheiros d trabalhos, especialmente CT Renato e 1T Souto, agradeço a amizade e o carinho que sempre me disponibilizaram.

Aos meus pais Wanderley e Maria Rosangêla que dignamente me apresentaram a importância da família e ao caminho da honestidade e persistência, obrigada pelas orações, pelas conversas de ânimo, pelo imenso carinho e amor comigo e princpalmente com a minha filha nas muitas horas em que tive que estar ausente. Não chegaria aqui sem o esforço e abdicação de vocês.

Aos meus filhos, minha princesa Isabella que participou desde o início do Mestrado, m apoiando ns momentos d dificuldades, mesmo com tão pouca idade soube compreender os

momentos em que a mamãe não pode lhe dar a devida atenção. E ao mais novo membro da família Matheus que daqui a pouquinho está chegando.

Ao meu amado marido Luiz Gabriel, que de forma especial e carinhosa me deu força e coragem, me apoiando nos momentos de dificuldades. Obrigada pelo apoio incondicional em todos os momentos, inclusive nas noites de sono perdidas em que você esteve ao meu lado apenas para eu não ficar sozinha, jamais esquecerei esse lindo gesto de amor, carinho e companheirismo.

Aos membros da banca examinadora, Prof Viterbo, Prof Diogo e Prof. Pedro, que tão gentilmente aceitaram participar e colaborar com esta dissertação.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigada.

"O que sabemos é uma gota; o que ignoramos é um oceano. Mas o que seria o oceano se não infinitas gotas?"

Isaac Newton

Resumo

O conceito de Blockchain surgiu em 2008 como uma estrutura de rede distribuida, capaz de garantir segurança às transações realizadas utilizando a moeda digital Bitcoin, sem a necessidade de intermediários para validá-las. Apesar de seu início estar ligado às criptomoedas, o seu uso se diversificou. Existem vários projetos utilizando a tecnologia Blockchain, como validação de documentos, votação eletrônico e a tokenização de bens não perecíveis. Com o seu crescente uso, surge a preocupação com possíveis ataques que possam ameaçar a integridade da cadeia e comprometer a segurança dos dados. Um dos ataques mais conhecidos é chamado na literatura como *"Selfish Miner"*, ou Minerador Egoísta, onde alguns nós maliciosos podem desviar seu comportamento do padrão ao não divulgarem imediatamente os seus blocos recém minerados, mantendo uma cadeia de blocos privada, com o objetivo de conseguir suplantar a cadeia honesta no momento da publicação, o que pode resultar em uma parcela desproporcional de recompensas para esses nós. Outro ataque é denominado *Stalker Miner*, ou Minerador Perseguidor, uma variação da estratégia de mineração egoísta que tem como objetivo impedir a publicação de um bloco ou transação específica de um nó, não necessariamente com fins econômicos. O objetivo desta dissertação é apresentar uma heurística para detectar a presença de ataque de mineração egoísta (e suas variações) na Blockchain que usa o algoritmo *Proof-of-Work* (PoW) ou Prova-de-Trabalho, com base no aumento da taxa de forks, com uma determinada altura, que é apresentada quando a rede está sob influência de tais ataques. O algoritmo de detecção foi capaz de identificar com precisão 99,98% das tentativas do ataque de mineração egoísta e 99,85% dos casos de ataque do perseguidor. A avaliação do algoritmo foi realizada através da quantidade de Falsos Posistivos e Verdadeiros Positivos identiicada após a identificação da ocorrência de ataque.

Palavras-chave: Blockchain; *Selfish-miner*; *Stalker*; algoritmo de detecção; *Bitcoin*; criptomoedas, Consenso Distribuído.

Abstract

Blockchain's concept emerged in 2008 as a distributed network structure, capable of guaranteeing security for transactions made using the Bitcoin digital currency, without the need for intermediaries to validate them. Although its beginning was linked to the cryptocurrencies, its use has diversified. There are several projects using Blockchain technology, such as document validation, electronic voting and the tokenization of non-perishable goods. With its increasing use, concerns arise about possible attacks that could threaten the integrity of the chain and compromise data security. One of the most well-known attacks is called in the literature as Selfish Miner, where some malicious nodes can deflect their behavior from the pattern by not immediately disclosing their newly mined blocks, keeping a blockchain private, with the goal of succeeding in overcoming the honest chain at the time of publication, which may result in a disproportionate share of rewards for those nodes. Another attack is called Stalker Miner, a variation of the selfish mining strategy that aims to prevent the publication of a specific block or transaction of a node, not necessarily for economic purposes. The objective of this dissertation is to present a heuristic to detect the presence of selfish mining attack (and its variations) in Blockchain using the Proof-of-Work (PoW) algorithm, based on the increase of the forks, with a certain height, which is displayed when the network is under the influence of such attacks. The detection algorithm was able to accurately identify 99,98% of the selfish mining attack attempts and 99,85% of the pursuer's attack cases. The evaluation of the algorithm was performed through the number of False Positive and True Positive identified after the identification of the occurrence of the attack.

Palavras-chave: Blockchain; *Selfish-miner*; *Stalker*; detection algorithm; *Bitcoin*; cryptocurrencies, Distributed Consensus.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADEPT	:	Telemetria Autônoma Par-a-Par descentralizada (*Automated Decentralized P2P Telemetry*);
BIP	:	Proposta de melhoria de Bitcoin (*Bitcoin Improvement Proposal*);
DLP	:	Problema do Logaritmo Discreto (*Discrete Logarithm Problem*);
DoS	:	Ataque de Negação de Serviço (*Denial Of Service, em inglês*);
DSA	:	Algoritmo de Assinatura Digital (*Digital Signature Algorithm*);
ECC	:	Criptografía de Curvas Elípticas (*Elliptic Curve Cryptography*);
ECDLP	:	Problema do Logaritmo Discreto em Curvas Elípticas (*Elliptic Curve Discrete Logarithm Problem*);
ECDSA	:	Algoritmo de assinatura digital com curvas elípticas (*Elliptic Curve Digital Signature Algorithm*);
IFP	:	Problema de Fatoração de Inteiros (*Integer Factorization Problem*);
MPC	:	Computação Segura entre Múltiplos Participantes (*Multi-Party Computing*);
NAT	:	Tradução de Endereço de Rede (*Network Address Translation*);
NIST	:	Instituto Nacional de Padrões e Tecnologia (*National Institute of Standards and Technology*);
PBFT	:	Algoritmo de Tolerância a Falhas Bizantinas (*Practical Byzantine Fault Tolerance*);
PKI	:	Infra-Estrutura Pública de Chaves (*Public Key Infrastructure*);
PoeT	:	Prova do Tempo Decorrido (*Proof of Elapsed Time*);
PoS	:	Prova de Posse (*Proof of Stake*);
PoW	:	Prova de Trabalho (*Proof-of-Work*);
P2P	:	Rede para a par (*Peer-to-Peer*);
RPC	:	Chamada remota de procedimento (*Remote Procedure Call*);
RSA	:	Rivest-Shamir-Adleman;
SHA	:	Algoritmo de Dispersão Seguro (*Secure Hash Algorithm*);
UTXO	:	Saída de Transação não Gasta (*Unspent Transaction Output*);
VSP	:	Verificação Simplificada de Pagamento;

Sumário

Capítulo 1

Introdução

1.1 Visão Geral

Blockchain ficou conhecida como a principal inovação tecnológica introduzida pela moeda digital Bitcoin apresentada em 2008 por Satoshi Nakamoto [43] que apresentou a cadeia de blocos como mecanismo para garantir irretratabilidade, auditabilidade, e imutabilidade a fim de prover segurança a transações eletrônicas, servindo como um grande livro razão distribuído. Esse sistema proposto por Nakamoto elimina a necessidade de instituições como bancos ou cartórios, chamados de "terceiros de confiança", pois todos os registros são, além de públicos, mantidos de maneira descentralizada por diversos participantes da rede.

Alcançar o consenso em um sistema distribuído é um desafio. Os algoritmos de consenso devem ser resilientes a falhas de nós, particionamento da rede, atrasos de mensagens, mensagens que chegam fora de ordem e corrompidas. Eles também têm que lidar com nós egoístas e deliberadamente maliciosos. Vários algoritmos tem sido propostos para resolver isso, cada um realizando o conjunto de suposições necessárias em termos de sincronia, transmissões de mensagens, falhas, nós maliciosos, desempenho e segurança das mensagens trocadas [6, 10, 49, 65]. Para uma rede Blockchain, alcançar consenso garante que todos os nós na rede concordem com um estado global consistente da cadeia de blocos [13].

Como o mecanismo de consenso, utilizado na rede Blockchain do Bitcoin, chamado de prova de trabalho (do inglês *Proof-of-Work* - PoW), depende de se ter uma maioria de mineradores, ou seja, mais que 51% de nós agindo honestamente por interesse próprio, ele se torna vulnerável a ataques de mineradores ou cooperativa de mineradores que tentarem

usar o seu poder de *hashing*[1] com finalidades desonestas ou destrutivas. Portanto, se um minerador ou um grupo de mineradores conseguir atingir uma cota significativa de poder de mineração, eles podem atacar o mecanismo de consenso, de maneira que comprometam a segurança e a disponibilidade da rede, esse ataque é conhecido como ataque de 51% [3].

Mineradores maliciosos também podem desviar seu comportamento do padrão ao não divulgarem imediatamente os seus blocos recém minerados, mantendo secretamente uma cadeia onde somente ele produz blocos, adotando uma heurística própria para a divulgação desses blocos. Neste ponto, todos os mineradores, incluindo o atacante, estariam explorando o bloco n. Ao fazer isso, se o atacante puder produzir o próximo bloco, ele ganhará uma vantagem sobre outros mineradores, mesmo sem ter maior poder computacional, já que ele pode iniciar o processo de mineração do bloco $n + 1$ antes de todos os outros mineradores, em sua cadeia privada. Como ele começou a minerar antes, há uma probabilidade de minerar o bloco $n + 1$ antes dos outros, gerando uma bifurcação na cadeia onte o atacante terá um *fork* com altura maior que a cadeia principal. À medida que os outros nós se comportarem honestamente, eles adotarão esse *fork* como a nova cadeia principal e o atacante alcançará seu objetivo, recebendo as recompensas pelos blocos minerados.

Estes ataques ao mecanismo de consenso são chamados na literatura de Mineração Egoísta (do inglês *Selfish Mining*) [21, 45], e sob certas condições, podem resultar em uma parcela desproporcional de recompensas a atacantes que se desviam do comportamento honesto. Uma variação da estratégia usada pelo minerador egoísta, com o objetivo de impedir a publicação de um bloco ou transação específica é chamada de Perseguidor (do inglês *Stalker[32]*), esse tipo de ataque pode gerar uma certa negação de serviço a um determinado minerador.

1.2 Objetivo

O objetivo deste trabalho é apresentar uma heurística para a detecção dos ataques de Mineração Egoísta e sua variação, o ataque do Perseguidor. Ambos ataques utilizam a Prova-de-Trabalho que é o mecanismo de consenso da rede Blockchain do Bitcoin e

[1] O Hashing refere-se ao processo de geração de uma saída de tamanho fixo a partir de uma entrada de tamanho variável. Isto é feito através do uso de fórmulas matemáticas conhecidas como funções hash. A blockchain da Bitcoin tem várias operações que envolvem hashing, a maioria delas no processo de mineração. O poder de hashing significa a capacidade de lidar com enormes quantidades de informação, sendo possível executar um arquivo grande ou conjunto de dados através de uma função hash e, em seguida, usar seu output para rapidamente verificar a precisão e integridade dos dados.

de outras criptomoedas . A heuristica foi construída através da observação na alteração da quantidade e no estudo da variação da altura dos *forks* que são apresentados sobre a evolução da cadeia de blocos. Os resultados das simulações realizadas indicam que a presença do comportamento de mineração egoísta e / ou perseguidora, faz com que a cadeia apresente uma taxa de forks maior, com uma média de altura maior ou igual a 2.

1.3 Trabalhos Relacionados

Desde a sua criação em 2009, a Blockchain da Bitcoin alimentou o surgimento de novas redes de criptomoedas como Ethereum, Litecoin e Dogecoin e várias aplicações inovadoras, como por exemplo, os contratos inteligentes, foram projetados para tirar proveito da Blockchain. Embora as Blockchain que utilizam a Prova de Trabalho atualmente representar mais de 90% da capitalização de mercado total das criptomoedas as questões de segurança não receberam muita atenção na literatura. O próprio Bitcoin usa o mecanismo de consenso de Prova de Trabalho como um sistema de segurança, com o objetivo de reduzir a possibilidade de ataques de gasto duplo[2], entretanto alguns destes ataques já foram registrados na rede como: o ataque de gasto duplo ocorrido na rede Bitcoin Gold[64], o ataque de Selfish Miner sofrido pela Monacoin[24] e o ataque de 51% na rede Ethereum Classic[53], onde alguns especialistas afirmaram que também se tratou de um ataque de Selfish Miner a maioria concentra-se na detectção de ataques de gasto duplo, porém um ataque de gasto duplo pode acontecer na presença de um ataque de Mineração Egoísta. Como trata-se de um ataque abordado recentemente, não existem muitas referências sobre sua detecção e até o momento da pesquisa não foi encontrado nenhum trabalho que abordasse a detecção de mineração egoísta considerando a alteração no padrão das ocorrências dos *forks* na cadeia de blocos.

Em [23] apresentou uma nova estrutura quantitativa para analisar as implicações de segurança e desempenho de diversos parâmetros de consenso e rede das Blockchains que utilizam como mecanismo de consenso a Prova-de-Tradbalho. Com base nessa estrutura, foram planejadas várias estratégias adversárias ideais para extração de gasto duplo e de mineração egoísta, levando em consideração as restrições do mundo real, como propagação de rede, tamanhos de blocos diferentes, intervalos de geração de blocos, mecanismo de propagação de informações e o impacto gerado por ataques de eclipse. Esse modelo permite simular as cadeias existentes com o mecanismo de PoW, bem como alterar as

[2]O cliente bitcoin clássico mostrará uma transação como "n/unconfirmed"até que a transação tenha 6 blocos de profundidade.

variáveis propostas que são instanciadas com diferentes parâmetros e comparar objetivamente a análise do custo-benefício entre seu desempenho e provisões de segurança. Os resultados permitem os usuárioss levem em conta as sugestões de segurança ao aceitar transações e avaliar seus respectivos riscos de gasto duplo e também ajuda os mineiros a quantificar a resiliência de uma Blockchain baseada em contra mineração egoísta, com base na taxa de blocos descartados, porém não implenta um modelo que possibilite a identificar esses ataques na rede.

[21] introduziu o ataque de Selfish Miner, mostrando que um minerador egoísta pode aumentar sua receita de mineração relativa ao não publicar diretamente seus blocos. A idéia-chave por trás dessa estratégia, é que um *pool* de mineradores mantenha seus blocos descobertos em sigilo, intencionalmente bifurcando a cadeia. Os nós honestos continuam a minerar na cadeia pública, enquanto o *pool* explora sua própria cadeia privada. Se o *pool* descobrir mais blocos, ele desenvolverá um ramo mais longo que a cadeia pública, os mineradores revelam os blocos de sua cadeia privada ao público, fazendo com que essa seja a cadeia com mais poder computacional, sendo ela a nova cadeia pública. Essa estratégia leva os mineradores honestos que seguem o protocolo Bitcoin a desperdiçar recursos em enigmas de criptografia de mineração que acabam não servindo para nada. A pesquisa demonstra que, enquanto partes honestas e egoístas desperdiçam alguns recursos, os mineradores honestos desperdiçam proporcionalmente mais e as recompensas do *pool* de mineradores desonestos excedem sua participação do poder de mineração da rede, conferindo-lhe uma vantagem competitiva e incentivando os mineradores honestos a se unirem ao grupo de mineradores egoístas. Mostramos ainda que o protocolo de mineração Bitcoin nunca será seguro contra ataques de um grupo de mineração egoísta que comanda mais de $1/3$ do poder de mineração total da rede, um valor substancialmente menor que os 50% atualmente assumido e difícil de alcançar na prática. O trabalho ainda sugere uma modificação prática no protocolo Bitcoin que protege contra pools de mineração egoístas que controlam menos de 25% dos recursos.

[32] apresentou uma nova estratégia do ataque do minerador egoísta, chamado de *Stalker mining*, cujo objetivo é impedir que um determinado nó publique um bloco, ou que uma determinada transação, deste nó seja incluída na cadeia. Seus objetivo não inclui ganhos financeiros. Os resultados de simulações demonstraram que a eficácia deste ataque é proporcional a razão de poder de mineração entre o atacante e o alvo. Além disso, foi verificado um efeito colateral deste ataque, onde com o aumento do poder do atacante, observa-se a perda de blocos de nós honestos da rede.

1.4 Organização da Dissertação

No Capítulo 2 é apresentado um referencial teórico sobre o funcionamento do Bitcoin, que foi o primeiro sistema a utilizar a Blockchain de forma comercial, utilizando como mecanismo de consenso a Prova-de-Trabalho.

O Capítulo 3 inicia-se definindo Blockchain, seus tipos e os mecanismo de consenso. É dada uma atenção especial ao mecanismo de prova de trabalho, que é utilizado neste trabalho.

No Capítulo 4 são descritos os principais ataques encontrados na literatura: ataque 51%, *Selfish Miner* ou Minerador egoísta, minerador teimoso, *Eclipse* e o *Stalker* ou Perseguidor.

O Capítulo 5 introduz uma heurística de detecção para o ataque de Mineração egoísta e para o ataque da Mineração Perseguidora. Os resultados alcançados são apresentados por meio de simulação usando o simulador de eventos discretos, NS-3. Também é apresentada uma análise da eficácia da heurística apresentada com base na taxa de Falsos Positivos e Verdadeiros Posistivos.

Por fim, o Capítulo 6 conclui a dissertação com algumas considerações, trabalhos futuros e contribuições.

Capítulo 2

Referencial Teórico

2.1 Bitcoin

O *Bitcoin* é um conjunto de conceitos e tecnologias que formam a base de um ecossistema de dinheiro eletrônico. Inicialmente apresentada em 2008 por um programador ou grupo de programadores sob o pseudônimo *Satoshi Nakamoto* que escreveu um artigo [6] contendo seus princípios de funcionamento. É considerada a primeira moeda digital mundial descentralizada, constituindo um método econômico alternativo. O mecanismo Bitcoin é um sistema distribuído Ponto-a-Ponto (ou P2P, do termo em inglês *peer-to-peer*) e foi proposto para resolver o problema da possibilidade de gasto duplo, pois ele permite transações financeiras sem intermediários. Bitcoin o faz distribuindo o imprescindível registro histórico a todos os usuários do sistema via uma rede P2P. Todas as transações que ocorrem na economia Bitcoin são registradas em uma espécie de livro-razão[1] público e distribuído chamado de Blockchain (tecnologia que armazena o registro público das transações) contendo o histórico de todas as transações realizadas. Novas transações são verificadas na Blockchain de modo a assegurar que os mesmos bitcoins[2] não tenham sido previamente gastos, eliminando assim o problema do gasto duplo [59]. A rede global peer-to-peer, composta de milhares de usuários (nós) da rede, torna-se o próprio intermediário, tornando as transações públicas, imutáveis e distribuídas, permitindo assim que qualquer nó da rede possa auditá-las.

A Figura 2.1 apresenta uma visão geral dos principais elementos que compõem a rede

[1] Livro-razão é nome dado pelos profissionais de contabilidade ao agrupamento dos registros contábeis de uma empresa que usa o método das partidas dobradas. Nele é possível visualizar todas as transações ocorridas em dado período de operação de uma empresa.

[2] Quando refere-se ao sistema, à rede ou ao projeto Bitcoin, usa-se sempre inicial maiúscula. No entanto, quando a referência é às unidades monetárias bitcoins, utiliza-se a palavra em caixa baixa.

Bitcoin. Os usuários ou nós da rede são classificados de acordo com as funções que podem desempenhar, são elas: Nó Completo; Nó Carteira; Minerador ou Cooperativa de Mineradores. Essas funções serão explicadas na próxima secão. Esses nós possuem carteiras que armazenam as moedas e realizam as transações. Essas transações são propagadas pela rede e verificadas por meio de um uso inteligente da criptografia de chave pública. Tal mecanismo exige que a cada usuário sejam atribuídas duas "chaves", uma privada, que é mantida em segredo, como uma senha, e outra pública, que pode ser compartilhada com todos.

O Bitcoin é uma rede peer-to-peer, não há uma autoridade central encarregada nem de criar unidades monetárias nem de verificar as transações. Essa rede depende dos usuários que proveem a força computacional para realizar os registros e as verificações das transações. Esses usuários são chamados de "mineradores", porque são recompensados pelo seu trabalho com bitcoins recém-criados. Os mineradores, agrupam diversas transações em uma estrutura de dados chamada de bloco. A operação de produzir um bloco implica em validar as transações, realizar a prova de trabalho e conectar o bloco produzido ao anterior. O primeiro minerador que conseguir produzir um bloco válido o enviará aos outros nós da rede. Cada nó completo da rede, ao receber este bloco, verificará se ele é válido e o adicionará a sua cadeia.

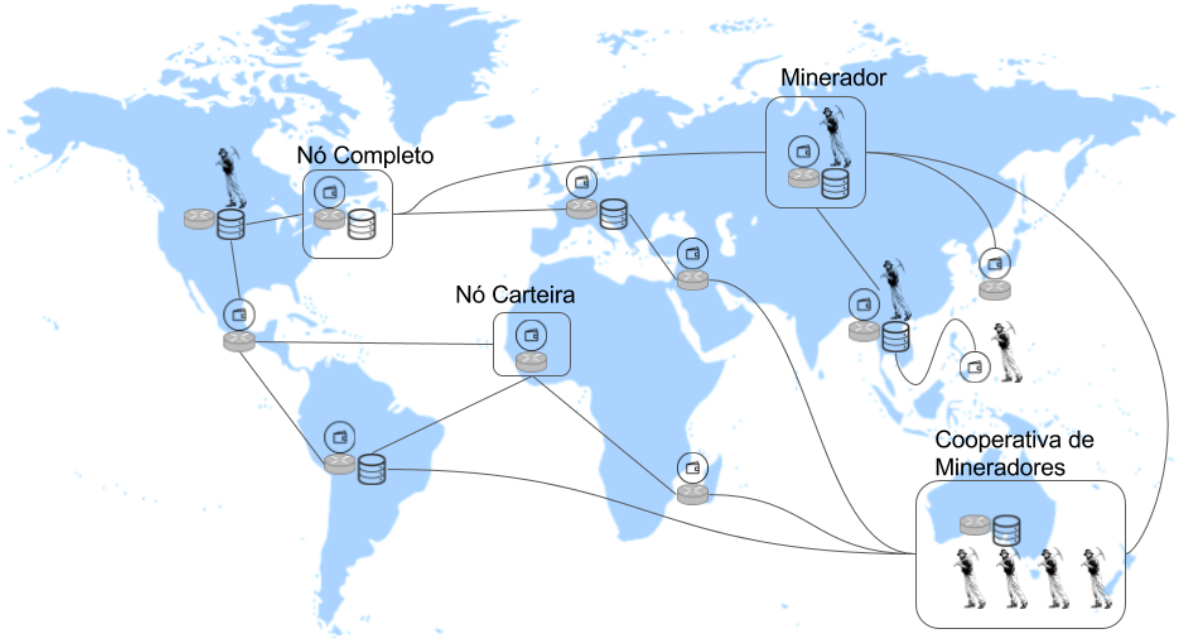

Figura 2.1: Visão geral da rede Bitcoin

O funcionamento do Bitcoin e suas partes principais consistem em:

- **Rede P2P** - a Rede Bitcoin;
- **Chaves, Endereços e Carteira** - a Criptografia no Bitcoin;
- **Transações** - um sistema descentralizado de verificação de transações;
- **Mineração** - uma emissão de moeda descentralizada, matemática e determinística; e
- **Blockchain** - um registro público de transações.

2.2 A Rede P2P

As Redes P2P são inerentemente resilientes, descentralizadas e abertas. A rede Bitcoin tem uma arquitetura P2P onde todos os participantes da rede são nós que funcionam tanto como servidores quanto como clientes sem hierarquia especial entre um nó ou outro. Todos são considerados hierarquicamente iguais e devem servir a rede com certos recursos, não existindo um nó centralizador. Este modelo aberto e descentralizado faz com que a rede não tenha uma barreira de entrada arbitrariamente escolhida. Todos os nós se interconectam na forma de uma rede sobreposta à internet, em uma rede mesh com uma topologia "plana".

Os nós na rede trocam mensagens contendo transações, blocos e endereços (IP) de outros nós entre si. Ao conectar à rede o cliente bitcoin deste nó realiza o processo chamado *bootstrap*[3] para achar e se conectar a outros nós na rede, e começa a baixar blocos para a cópia local da blockchain de nós mais adiantados na rede. Ao mesmo tempo o seu cliente bitcoin já pode começar a prover dados para os nós conectados e ajudar a rede enquanto se atualiza com o consenso da rede. Quando uma transação é criada e enviada para a rede por um nó, o seu cliente envia esta transação para alguns nós, estes nós enviam para outros e assim, sucessivamente, até que um minerador valide a transação e a inclua num bloco, e, então, transmita este bloco aos nós de forma sucessiva até que o cliente do nó que enviou a transação receba o novo bloco e verifique que a transação foi confirmada.

[3]Na computação, bootstrapping refere-se ao processo em que um sistema simples ativa outro sistema mais complexo para servir ao mesmo propósito.

2.2.1 Tipos de Nós

Embora os nós na rede P2P do Bitcoin sejam iguais, eles podem assumir diferentes papéis dependendo da funcionalidade que eles estejam suportando. Existem quatro características/funções básicas que um nó pode ter na rede: roteamento, o banco de dados da blockchain, mineração e serviços de carteira. Nem todos os nós apresentam todas as quatro características/funções, podendo apresentar apenas uma, mas todos os nós incluem a função de roteamento para participar na rede e podem incluir outras funcionalidades. Todos os nós validam e propagam as transações e blocos, e descobrem e mantém conexões com os pontos. Os principais nós são:

- **Nó completo:** Um nó completo possui todas as quatro funções, mantém uma cópia completa e atualizada da blockchain podendo verificar de maneira autônoma e autoritária qualquer transação sem referência externa.

- **Nó simplificado:** Busca somente um meio para realizar pagamentos. Possui apenas a carteira e o roteamento. Desta forma, ele pode se conectar a rede e realizar transações somente com dispositivos simples como um celular, sem a necessidade de armazenar toda a cadeia de blocos. Esses nós não possuem uma visão completa da rede e necessitam de ajuda de outros nós para fazer checagens de rotina, por exemplo, ao receber um pagamento um nó quer saber se o valor recebido é válido. O protocolo então específica que os nós completos podem realizar essas checagens e responder aos nós simples.

- **Minerador:** Contém a função de mineração, a base de dados completa e o roteamento. Os nós de mineração competem para criar novos blocos ao utilizarem hardware especializado para resolver os algoritmos de prova-de-trabalho.

- ***Pool* de Mineração:** Os mineradores que paricipam de uma *Pool* dividem o trabalho de buscar uma solução para um bloco candidato, cada um deles ganhando "cotas"pela sua contribuição com a mineração. O *Pool* de mineração define um alvo de dificuldade menor para ganhar uma cota, tipicamente mais do que 1.000 vezes mais fácil que a dificuldade da rede bitcoin. Quando alguém do pool minera um bloco com sucesso, a recompensa é destinada ao *Pool* e então é redistribuída para todos os mineradores, de acordo com o número de cotas que cada um deles contribuiu para o esforço coletivo. Portanto, os mineradores do *Pool* compartilham o esforço necessário para minerar um bloco, e, então, comparilham as recompensas

da mineração. Internamente utilizam um protocolo próprio da *Pool* para distribuir o trabalho entre os mineradores, mas para rede aparecem como um único minerador.

2.2.2 Descoberta da Rede

Quando um novo nó é ligado, ele deve descobrir outros nós Bitoins na rede para que possa participar. Para iniciar esse processo, um novo nó deve descobrir pelo menos um nó existente na rede para conectar-se a ele. A localização geográfica dos outros nós é irrelevante, pois a topologia da rede Bitcoin não é definida geograficamente. Logo, qualquer nó Bitcoin existente pode ser selecionado aleatoriamente. Para se conectar a um ponto conhecido, os nós estabelecem uma conexão TCP, geralmente na porta 8333 (a porta geralmente conhecida como uma das usadas pelo bitcoin), ou a uma porta alternativa caso tenha sido fornecida.

Para estabelecer a conexão inicial o nó realiza um *HandShake* com uma mensagem *version*, que contém informações de identificação. Assim que a conexão é estabelecida, o nó envia uma mensagem GETADDR solicitando uma lista de endereços IP conhecidos. De posse desta lista, inicia o processo novamente para outros nós, desta lista, a fim de se tornar bem conectado. Após a primeira vez que o nó é conectado, ele armazena em disco uma lista com todos os nós com os quais estabeleceu conexão recentemente, assim das próximas vezes que se ligar a rede pode não necessitar do auxílio dos *Seeders*.

Além das mensagens *version e ADDR* o protocolo especifica mensagens para troca de dados, que são as mensagens para difusão das transações e dos blocos. A Figura 2.2 sintetiza as principais mensagens utilizadas.

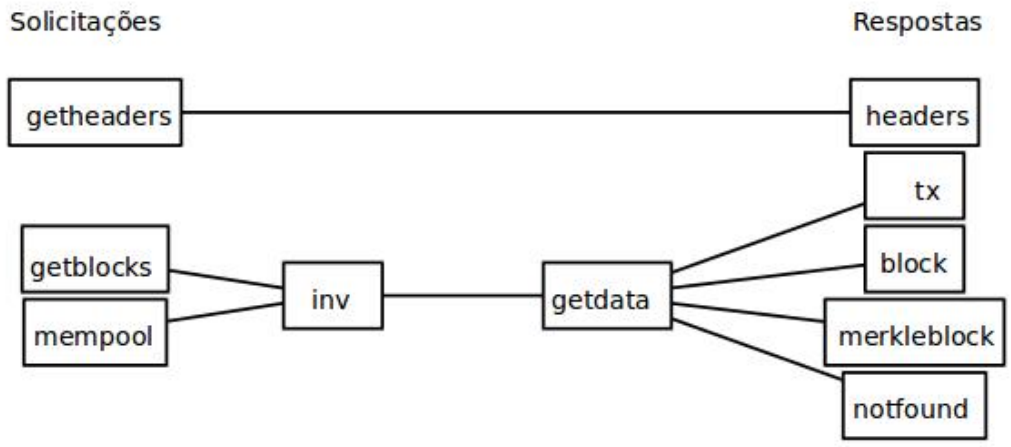

Figura 2.2: Mensagens do protocolo Bitcoin

A primeira coisa que um nó completo fará assim que se conectar aos outros pontos da rede é tentar construir uma blockchain completa. Se ele for um nó recém criado que não tenha nenhuma parte da Blockchain, ele só terá um bloco, o bloco gênese. A primeira mensagem enviada ao se conectar ao outro nó inclui o *bestHeight* que informa até que bloco ele tem conhecimento. Assim que ele receber a mensagem de versão do nó que está conectado contendo o *bestHeight* dele, ele saberá se está precisando sincronizar a sua cópia local da blockchain com o resto do consenso da rede ou se é o nó em que está conectado que está mais atrás. Os nós pareados irão trocar uma mensagem *getblocks*, a qual contém o *hash* (impressão digital) do bloco mais alto de suas blockchains locais. O nó que tiver a blockchain mais longa (normalmente, a com mais trabalho), já sabendo quantos blocos o nó com a blockchain mais curta precisa, identificará os 500 primeiros blocos que o nó está precisando para alcançar o resto da rede e enviará uma mensagem *inv* com os hashes dos blocos para que, então, o nó que precisa se atualizar envie uma mensagem *getdata* pedindo a informação completa de cada bloco para que ele possa baixar os blocos e verificar um por um de forma independente. Este nó, agora, faz parte da rede P2P Bitcoin e contribui recebendo, verificando e transmitindo transações, blocos e outras informações ao resto da rede.

Toda vez que um nó conectado à rede cria uma transação, ele envia uma mensagem *inv* informando a nova transação para todos os seus nós conectados. A mensagem *inv* é uma mensagem de inventário que apenas notifica outros nós sobre um novo objeto descoberto ou envia dados sobre algo que está sendo requisitado. O nó que recebe uma mensagem *inv* sabe se tem ou não um objeto porque esta mensagem inclui o hash do objeto. Os outros nós que receberem a mensagem *inv* e não conhecerem este novo objeto, por exemplo, uma nova transação, enviarão uma mensagem *getdata* incluindo o hash do objeto e pedindo a informação completa. Ao receberem a transação, poderão verificar a validade dela por si próprios e, em caso da transação ser válida, repetirão o mesmo processo com os outros nós propagando a transação sucessivamente pela rede.

Enquanto estas transações recebidas e validadas pelos nós não são incluidas por algum minerador à blockchain, elas, geralmente, permanecem em um espaço de memória volátil de cada nó completo da rede. A maioria dos nós conectados à rede implementam a *pool* chamada *mempool* e, alguns outros, implementam *pools* separadas como as *orphan pools* para as chamadas transações orfãs e as *UTXO pools* que pode ser implementada em uma memória local persistente dependendo do tipo do software utilizado pelo nó.

- **mempool:** guarda todas as transações recebidas e validadas pelo nó que não te-

nham sido escritas na blockchain por algum minerador. Quanto mais transações sendo propagadas pela rede, maior a *mempool* ficará até que ela vá sendo gradualmente esvaziada com as transações sendo aceitas pelos mineradores. Muitas implementações de carteiras usam a *mempool* para calcular as taxas de transação ideais para que a transação seja aceita o mais rápido quanto possível, criando uma sugestão de taxa flutuante.

- **orphan pool:** é uma *pool* separada da *mempool* existente em algumas versões de nós completos na rede que guardam as transações que referenciem e dependam de uma transação anterior a elas, chamadas de *parent transactions* - transações pai/mãe - que ainda não tenham sido vistas por este nó.

- **UTXO pool:** diferente da *mempool*, é uma *pool* de transações presente em algumas implementações que guarda em memória volátil ou persistente local, milhões de entradas de outputs de transação criadas e nunca gastas com UTXOs que podem datar, em alguns casos, de transações criadas em 2009. E, enquanto a *mempool* e a *orphan pool* apenas contém transações não confirmadas, esta *pool* apenas contém transações confirmadas na rede.

Alguns nós da rede são nós simples, que possuem apenas funções de roteamento e carteira. Esses nós não possuem uma visão completa da rede e necessitam de ajuda de outros nós para fazer checagens de rotina, por exemplo, ao receber um pagamento um nó quer saber se o valor recebido é válido. O protocolo então específica que os nós completos podem realizar essas checagens e responder aos nós simples. Para isso eles proveem um serviço de RPC(*Remote Procedure Call*) de modo que aqueles nós que são limitados possam realizar consultas sobre a rede e realizar operações típicas de carteira. Para receber os cabeçalhos dos blocos, os nós simples usam uma mensagem *getheaders* ao invés de *getblocks*. O nó que responder irá enviar até 2.000 cabeçalhos de blocos usando uma mensagem *headers* única. O processo é o mesmo que o utilizado para um nó completo receber os blocos completos. Quaisquer transações de interesse são recebidas usando uma requisição *getdata*. Em resposta, os pontos geram uma mensagem *tx* contendo as transações.

A verificação simplificada de pagamento (VSP) executada por um nó simples verifica as transações através de referências às suas profundidades na blockchain, ao invés de sua altura. Enquanto um nó com a blockchain completa irá construir uma cadeia completamente verificada de milhares de blocos e transações que siga pela blockchain (retrospectivamente no tempo) até o bloco gênese, um nó simples irá verificar a cadeia

de todos os blocos (mas não todas as transações) e irá ligar essa cadeia à transação de interesse. Por exemplo, ao examinar uma transação no bloco 300.000, um nó completo segue todos os 300.000 blocos até o bloco gênese e constrói um banco de dados completo de UTXO, estabelecendo a validade da transação ao confirmar que a UTXO ainda não foi gasta. Um nó simples não pode validar se a UTXO ainda não foi gasta. Ao invés disso, o nó simples irá estabelecer uma conexão entre a transação e o bloco que a contém, usando um caminho de Merkle. Então, o nó simples aguarda até ver os seis blocos do 300.0001 até o 300.006 empilhados sobre o topo do bloco contendo a transação e verifica-a ao estabelecer sua profundidade sob os blocos 300.006 a 300.001. O fato de que outros nós na rede aceitaram o bloco 300.000 e então fizeram o trabalho necessário para produzir mais seis blocos no seu topo é a prova, através de proxy, que a transação não foi um gasto duplo.

Um nó simples não pode ser convencido de que uma transação existe em um bloco quando a transação de fato não existe. O nó simples estebelece que uma transação existe em um bloco ao solicitar um caminho de merkle como prova e ao validar a prova de trabalho na cadeia de blocos. Entretanto, a existência de uma transação pode ser "escondida"de um nó simples. Um nó simples pode provar definitivamente que uma transação existe, mas não pode verificar que uma transação, como um gasto duplo de uma mesma UTXO, não existe porque ele não tem um registro de todas as transações. Essa vulnerabilidade pode ser usada em um ataque de negação de serviço (DOS) ou para um ataque de gasto duplo contra os nós simples Para defender-se contra isso, um nó simples precisa se conectar aleatoriamente a vários nós, para aumentar a probabilidade de que ele esteja em contato com pelo menos um nó honesto. Essa necessidade de se conectar aleatoriamente significa que os nós simples também são vulneráveis a ataques de particionamento de rede ou ataques Sybil, onde eles são conectados a nós ou redes *fakes* e não tem acesso a nós honestos ou à rede Bitcoin verdadeira.

Como os nós simples precisam baixar transações específicas para verificá-las seletivamente, eles também criam um risco de privacidade. Ao contrário dos nós com a blockchain completa, que baixam todas as transações no interior de cada bloco, as solicitações por dados específicos que são feitas pelos nós simples podem inadvertidamente revelar os endereços em suas carteiras. Por exemplo, um terceiro monitorando uma rede poderia monitorar todas as transações solicitadas por uma carteira em um nó simples e usá-las para associar endereços bitcoins com o usuário dessa carteira, destruindo a privacidade do usuário. Para contornar esse problema uma funcionalidade chamada filtros bloom foi

desenvolvida para resolver os riscos de privacidade dos nós simples. Os filtros bloom[4][8][9] permitem que os nós simples recebam um conjunto de transações sem que revelem precisamente em quais endereços eles estão interessados, através de um mecanismo de filtros que utiliza probabilidades ao invés de padrões fixos. Em resposta a uma mensagem *getdata* vindo do nó simples, os outros nós da rede irão enviar uma mensagem *merkleblock* que contém somente os cabeçalhos de bloco para os blocos correspondentes ao filtro e um caminho merkle para cada transação correspondente. O ponto também enviará mensagens *tx* contendo as transações que correspondem ao filtro.

2.3 Chaves, Endereços e Carteira

2.3.1 Chaves Privada e Pública

Toda a posse de recursos e transações na rede são feitas utilizando-se o conceito de chaves e assinaturas digitais. As chaves usadas são geradas aplicando o conceito de criptografia de chaves públicas. São geradas um par de chaves, uma pública que pode ser compartilhada e uma secreta que somente o dono tem acesso. Toda transação requer uma assinatura para ser considerada válida e para provar a posse do recurso dispendido.

Resumos criptográficos, ou hash, são funções matemáticas que geram um resumo, uma espécie de impressão digital dos dados de entrada. Quando aplicadas a um determinado conjunto de dados ela irá gerar como saída um valor, que a princípio, é único. Um dos usos mais frequentes para o hash é verificar a integridade de arquivos. Por exemplo, ao assinar digitalmente um documento, a pessoa que o recebe, além de verificar a chave usada para a assinatura, também compara o hash fornecido pelo emissor do documento com o calculado na hora do recebimento. Caso o documento sofra alguma alteração, os resumos serão diferentes. O tamanho da saída do hash depende do algoritmo usado, mas o importante é que ela seja sempre do mesmo tamanho, não importando o tamanho da entrada. Exemplos de algoritmos de hash são o SHA-256 e o RIPEMD160, ambos usados pelo Bitcoin. Os algoritmos de hash devem possuir algumas características:

- **Unidirecionalidade:** Deve ser computacionalmente muito difícil encontrar a entrada a partir do resumo.

[4]É uma estrutura de dados probabilística com eficiência de espaço, concebida por Burton Howard Bloom em 1970, usada para testar se um elemento é membro de um conjunto. Correspondências de falso positivo são possíveis, mas falsos negativos não são.Uma consulta retorna "possivelmente no conjunto"ou "definitivamente não no conjunto". Os elementos podem ser adicionados ao conjunto, mas não removidos, quanto mais elementos forem adicionados ao conjunto, maior a probabilidade de falsos positivos.

- **Compressão:** É desejável que o tamanho do resumo represente uma fração pequena dos dados.
- **Facilidade de cálculo:** Não deve ser custoso calcular o valor do resumo.
- **Difusão:** Para dificultar a engenharia reversa do algoritmo ao mudar um bit dos dados de entrada o valor do resumo deve ser alterado de uma quantidade de bits próxima a 50%.
- **Colisão:** Deverá ser computacionalmente difícil encontrar dois valores de entrada que gerem o mesmo resumo.

Todos os sistemas de criptografia usam uma transformação de uma mensagem clara em uma ilegível. Para realizar esta transformação são usadas algumas transformações matemáticas sobre a mensagem clara juntamente com uma chave. Após essas transformações é obtido um texto cifrado que poderá ser lido apenas por quem possuir a chave para decifrar. Um dos sistemas de criptografia mais simples foi o utilizado por Júlio Cesar, que consistia em substituir as letras do texto por outras posteriores espaçadas da mesma chave. Por exemplo, escrever SBC com chave 13 resulta em FOP. Este tipo de criptografia também pode ser classificado como Criptografia de Chave Privada ou Simétrica, onde uma única chave é usada para criptografar e de-criptografar o segredo.

Outro tipo de criptografia é a Assimétrica ou de Chave Pública. Onde usa-se um par de chaves, uma pública e outra privada, a primeira para criptografar e a segunda para de-criptografar e vice-versa. Isso é possível graças ao uso de algumas funções matemáticas que possuem a propriedade de serem irreversíveis, significando que elas são fáceis de calcular em uma direção, e inviáveis de serem calculadas na direção oposta. As mais usadas são a fatoração em números primos (IFP - *Integer Factorization Problem*), curvas elíptica s(ECDLP - *Elliptic Curve Discrete Logarithm Problem*) ou logaritmos discretos (DLP - *Discrete Logarithm Problem*). A eficiência de um sistema de criptografia pode ser medida considerando:

- **Carga Computacional:** Mede a eficiência com que os algoritmos podem implementar as transformações com as chaves públicas e privadas.
- **Tamanho da Chave:** O NIST indica o uso de pares de chave (pública, privada) com tamanhos, em bits, para cada tipo de implementação: RSA (1088,2048), DSA (1026,160), e ECC O Bitcoin adota o *Elliptic Curve Digital Signature Algorithm* (ECDSA)[34] para realizar assinaturas. É uma versão baseada em curvas elípticas.

Assume-se que a dificuldade de cálculo do logaritmo discreto não permita que terceiros assinem um documento sem que tenha conhecimento da chave privada de uma pessoa. Pensando de modo inverso, se é impossível forjar a assinatu

(161,160). O ECC apresenta grande vantagem nesse aspecto.

- **Tamanho de Banda:** Corresponde a quantidade de bits necessária para transmitir uma mensagem, após codificar ou assinar.

[31] comparou o ECC com os RSA e chegou a conclusão de que para um mesmo nível e segurança a ECC possui uma menor carga computacional, menor tamanho de chave e menor tamanho de banda. Por esses motivos o Bitcoin adotou o sistema de curvas elípticas definida em um padrão chamado *secp256k1*, estabelecido pelo Instituto Nacional de Padronização e Tecnologia (NIST). Para maiores informações sobre curvas elípticas é recomendado [25].

Uma assinatura digital é a cifragem do hash de um documento usando uma chave privada, tal que a chave pública, da mesma pessoa que assinou, seja usada para provar que foi ela quem assinou aquele documento.

O Bitcoin adota o *Elliptic Curve Digital Signature Algorithm* (ECDSA)[34] para realizar assinaturas. É uma versão baseada em curvas elípticas. Assume-se que a dificuldade de cálculo do logaritmo discreto não permita que terceiros assinem um documento sem que tenha conhecimento da chave privada de uma pessoa. Pensando de modo inverso, se é impossível forjar a assinatura, então uma assinatura válida não pode ser refutada pelo dono da chave. Normalmente, o processo de assinar um documento é realizado sobre seu resumo criptográfico. Uma vantagem de se usar estas funções é que elas sempre geram como saída uma pequena quantidade de bits de mesmo tamanho. A assinatura deve ser capaz de prover integridade, não repúdio e autenticidade.

A chave privada é obtida gerando um número aleatório de 256 bits e a chave pública é obtida ao efetuar a multiplicação da chave privada por um ponto na curva conhecido como "ponto gerador". Ele é sempre o mesmo para todos os usuários do Bitcoin e é definido na especificação secp256k1. O resultado da multiplicação da chave privada pelo ponto gerador é um ponto na curva, este ponto é a chave pública, por exemplo: Começando com uma chave privada na forma de um número k aleatoriamente gerado, o multiplicamos por um ponto predeterminado na curva, chamado de o ponto gerador para produzir outro ponto em outro lugar da curva, que será a chave pública K correspondente. Os nós armazenam somente as suas chaves privadas, pois ele pode a qualquer momento gerar a

pública correspondente.

Existem dois tipos de preocupações com ECDSA, uma de ordem política que refere-se a confiabilidade das curvas produzidas pelo NIST sendo questionada após serem feitas revelações de que a NSA voluntariamente insere backdoors em software, componentes de hardware e padrões publicados e outra de ordem técnica referente a dificuldade de implementar corretamente o padrão, sua lentidão e falhas de projeto reduzem a segurança em implementações incorretas do gerador de número aleatório[39][61].

Em dezembro de 2010, um grupo autodenominado fail0verflow anunciou a recuperação da chave privada do ECDSA usada pela Sony para assinar software para o console de jogos PlayStation 3. No entanto, esse ataque só funcionou porque a Sony não implementou corretamente o algoritmo, porque k era estático em vez de aleatório[29]. Em agosto de 2013, foi revelado que os erros em algumas implementações da classe Java SecureRandom, por vezes, gerava colisões entre os valores de k. Isso permitiu que hackers recuperassem chaves privadas, dando-lhes o mesmo controle sobre transações de bitcoin que os proprietários legítimos das chaves tinham, usando a mesma falha que foi usada para revelar a chave de assinatura do PS3 em algumas implementações em aplicativos para Android, que usam Java e dependem de ECDSA para autenticar transações[14]. Este problema pode ser evitado por meio da geração determinística de$k$, como descrito pela RFC 6979[50].

2.3.2 Endereços

A partir deste ponto, o nó já possui um par de chaves ele pode gerar o endereço. O endereço, não confundir com endereço IP, é um número obtido usando a chave pública do nó. Ele é usado para informar ao sistema quem é o dono daquela transação, pois somente quem possuir a chave privada que gerou aquele endereço poderá usar o que estiver na transação, seja um montante monetário ou um dado. Para gerar o endereço o nó deve realizar uma operação de duplo hash, primeiro o *Secure Hash Algorithm* (SHA-256) e depois o *RACE Integrity Primitives Evaluation Message Digest* (RIPEMD 160): $Endereco = RIPEMD160(SHA256(ChavePublica))$, após deve converter o resultado para Base58, ver Figura 2.3.

Um endereço bitcoin é uma string de dígitos e caracteres que pode ser compartilhada com qualquer pessoa que queira lhe enviar dinheiro. Os endereços produzidos a partir de chaves públicas consistem em uma string de números e letras, iniciando com o dígito "1".

Os endereços bitcoin são quase sempre apresentados aos usuários em uma codificação

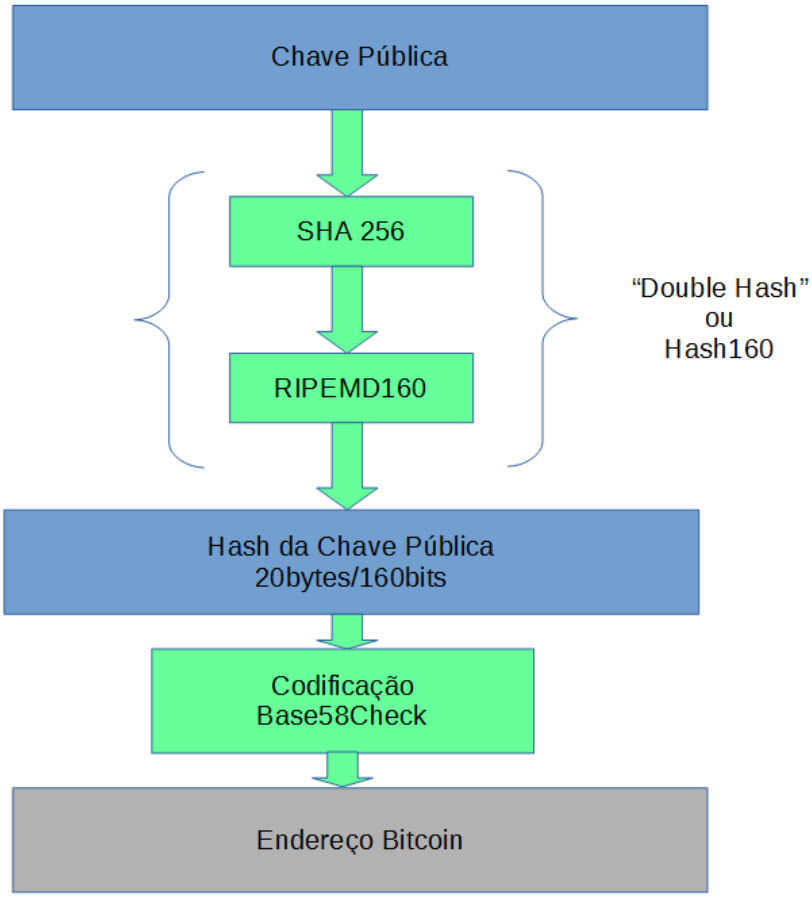

Figura 2.3: Chave pública para endereço bitcoin, adaptado de [3]

chamada "Base58Check", que usa 58 caracteres (um sistema numérico de Base58) e um checksum para ajudar a legibilidade humana, evitar ambiguidade, e proteger contra erros em transcrição e digitação de endereços. O checksum é composto de quatro bytes adicionais que são acrescidos no final dos dados sendo codificados. O checksum é derivado do hash dos dados codificados e, portanto, poder ser usado para detectar e prevenir erros de transcrição e digitação. Quando o software decodificador se depara com um código Base58Check, ele calcula o checkum dos dados e o compara com o checksum incluído no código, se os dois não corresponderem, tem-se uma indicação de que o erro foi introduzido e o dado Base58Check é inválido, isso previne que um endereço bitcoin digitado errado seja aceito pelo software da carteira como um destino válido, um erro que, do contrário, resultaria na perda de fundos.

Para converter dados no formato Base58Check, primeiro adicionamos um prefixo aos dados, chamados de "byte de versão", que serve para identificar facilmente o tipo de dado codificado, no caso de um endereço bitcoin, o prefixo é zero (0x00 em hexa). Em seguida,é computado o checksum "duplo-SHA", o que significa aplicar o algoritmo de hash SHA256

duas vezes no resultado anterior (prefixo e dados): $checksum = SHA256(SHA256(prefixo + dados))$, após, pega-se apenas os primeiros quatro bytes do hash de 32 bytes resultante, esses quatro byte servem como código de verificação de erro, ou checksum. O checksum é concatenado ao final. Portanto o resultado é composto de três itens: um prefixo, os dados, e um checksum, conforme Figura 2.4.

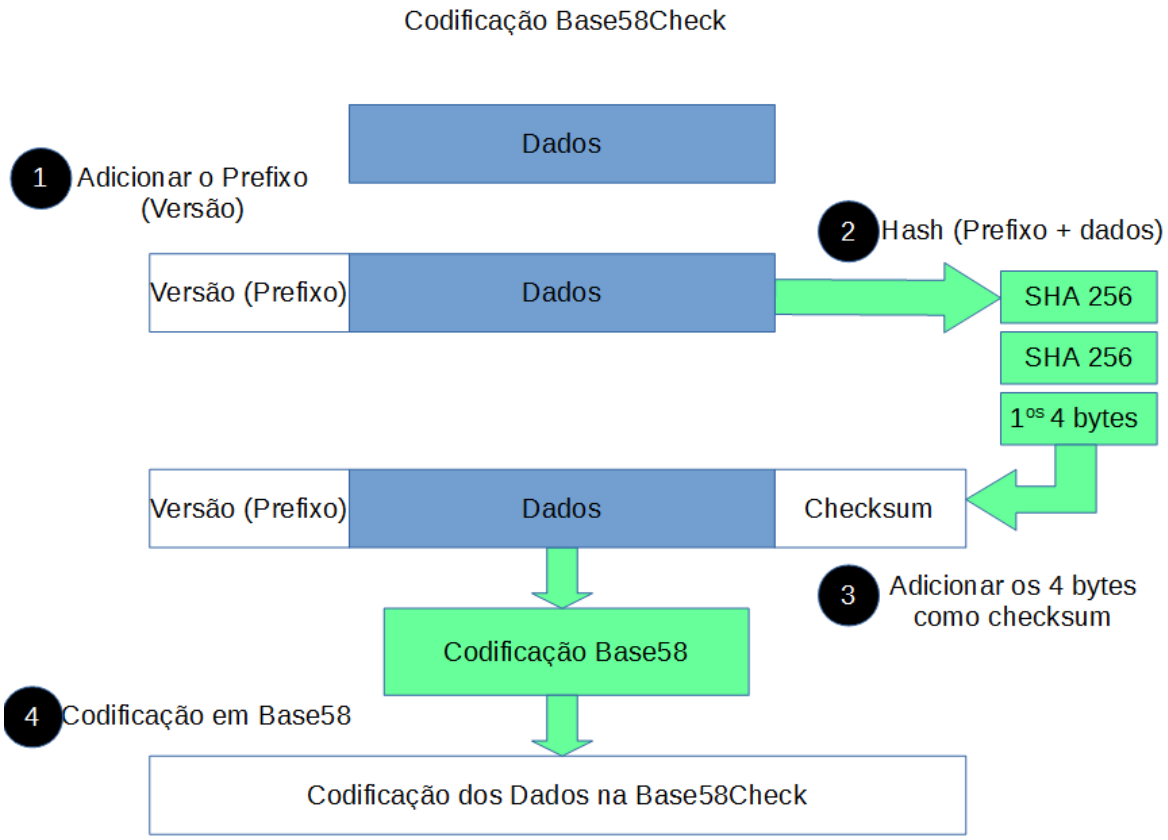

Figura 2.4: Codificação Base58Check, adaptado de [3]

2.3.3 Carteira

Os usuários do Bitcoin possuem chaves que permitem provar a posse de transações. Essas chaves precisam ser armazenadas, e geralmente são armazenadas em uma carteira digital. A carteira tem a função de gerar e armazenar as chaves dos usuários. As carteiras na verdade são chaveiros que contém pares de chaves privadas e públicas e/ou disponibilizam funcionalidades para administração destas chaves, como: backup, envio de transações, monitoramento de recebimento de transações, UTXOs, geração de endereços, etc. As carteiras podem guardar outras informações sobre as transações para melhor usabilidade como anotações sobre as transações, e outras utilidades como assinatura de

textos ou arquivos com suas chaves privadas. Os usuários assinam as transações com as chaves, provando que eles são donos dos outputs da transação, ou seja, provando que são donos de seus bitcoins. Os bitcoins são armazenados na blockchain na forma de outputs de transações.

De acordo com o método de geração de chaves, existem quatro tipos de carteira: *Single-Address Wallets*; *Nondeterministic (Random) Address Wallets*; *Deterministic Address Wallets* e *Hierachical Deterministic (HD) Wallets*:

- **Single-Address Wallets:** são carteiras que utilizam um único endereço para recebimento de transações e trocos, envio, etc. Não são um bom tipo de carteira para privacidade e segurança, e são cada vez menos utilizadas substituídas por métodos superiores para a maioria dos casos de uso.

- **Nondeterministic (Random) Address Wallets):** carteiras não-determinísticas carteiras tem um armazenamento de tamanho fixo de endereços gerados aleatoriamente. As chaves privadas são geradas de forma aleatória, tem seus endereços correspondentes gerados e ficam armazenadas. Este esquema tem alguns problemas para a segurança dos bitcoins caso backups não sejam feitos regularmente.

- **Deterministic Address Wallets:** carteiras determinísticas geram chaves privadas derivadas a partir de uma *seed* (semente) em comum. Para a recuperação dos bitcoin neste tipo de carteira, só precisa estar de posse da *seed* inicialmente gerada. Neste tipo de carteira, diferente das não-determinísticas, há a garantia de que todas as chaves privadas serão geradas na mesma ordem não importando o dispositivo em que ela estiver rodando. Isto facilita a segurança e usabilidade já que não é mais necessário repetir o processo de backup da carteira original de tempos em tempos.

- **Hierachical Deterministic (HD) Wallets:** carteiras hierárquicas determinísticas como definidas pelas BIP0032 e BIP0044[68] são um tipo avançado de carteira determinística em que as chaves também são geradas a partir de uma seed e todas são geradas em uma ordem não aleatória. A diferença aparece na feature que este tipo de carteira apresenta ao gerar suas chaves formando uma estrutura de árvore com cada folha na árvore tendo a possibilidade de gerar as chaves filhas e não as acima delas. Isto possibilita uma organização estrutural superior da carteira com facilidades de organização, divisão de carteiras e, inclusive, a possibilidade de ter uma carteira dividida por suas folhas, de acordo com a estrutura organizacional sem comprometer as chaves privadas mestras.

De acordo com o método de segurança das chaves, existem quatro tipos de carteira: *Hot Wallets*; *Cold Wallets*; *Paper Wallets* e *Hardware Wallets*:

- **Hot Wallets:** são todas as carteiras que, em algum momento, tem as suas chaves privadas expostas em um ambiente com conexão disponível à Internet. Estas são, provavelmente, as carteiras mais usadas e que apresentam o menor nível de segurança para o usuário. Pelo menos, toda vez em que uma nova transação precisa ser assinada, o usuário digita uma senha ou PIN para desencriptar a chave privada e assinar a transação em um ambiente com conexão à Internet, fazendo com que a chave fique exposta mesmo que por alguns momentos.

- **Cold Wallets:** são todas as carteiras rodando em um ambiente sem conexão à Internet. A chave privada nunca é exposta em um computador com acesso à internet, normalmente são criadas e usadas para acumular bitcoins e, quando usadas, descartadas e trocadas por novas chaves com menos chance de terem sido comprometidas. Também são usadas como carteira em ambiente totalmente offline apenas para assinar transações e ter as transações levadas de alguma outra forma ao conhecimento da rede bitcoin. O ideal para a segurança é que as chaves sejam sempre renovadas quando uma transação for feita.

- **Paper Wallets:** podem ser consideradas um tipo de *cold storage*, já que são carteiras com as chaves privadas impressas em papel para proteção física da chave privada. A efetividade da segurança deste método depende de como as chaves foram geradas, em que ambiente elas foram geradas (ex.: um computador sem conexão) e como elas são armazenadas com segurança contra roubo ou destruição. Um método interessante para este tipo de carteira é o M-de-N que permite que se tenha um número N de papéis e precise apenas de M destes papéis para gastar os bitcoins. Desta forma além da proteção das chaves não estarem conectadas, poderá existir a proteção contra um único ponto de falha já que, caso um papel seja destruído ou roubado, os bitcoins ainda estarão seguros para serem enviados para uma nova carteira segura.

- **Hardware Wallets:** também podem ser consideradas um tipo de *cold storage*. O interessante desta carteira é a possibilidade de ter um nível de segurança razoável em comparação aos outros métodos ao mesmo tempo que se pode ter uma carteira boa para ser utilizada com mais frequência no dia-a-dia. Técnicamente, este tipo de carteira se vale de uma separação de hardware, sendo um dispositivo seguro, desconectado de qualquer rede e que não confia no computador a que ela é conectada.

Tudo que ela faz é assinar transações com as chaves privadas armazenadas nela e verificar a validade de endereços de recebimento para proteção contra certos tipos de ataque *man-in-the-middle*.

2.4 Transação

Transações são a parte mais importante do sistema bitcoin. Todo o resto é desenhado para garantir que as transações possam ser criadas, propagadas na rede, validadas, e finalmente adicionadas ao livro-razão de transações (a Blockchain). Transações são estruturas de dados que codificam a transferência de valor entre participantes do sistema Bitcoin. Cada transação é uma entrada pública na Blockchain do Bitcoin, o livro-razão de dupla entrada global [3].

O ciclo de vida de uma transação é todo o processo desde a criação de uma transação até a inclusão dela na Blockchain por um minerador. De forma resumida, uma transação bem-sucedida é criada, assinada com a chave privada correspondente para "destrancar"os *outputs* da transação anterior referenciados nos *inputs* da transação atual, enviada à rede, verificada e validada por todos os nós na rede P2P Bitcoin, e propagada por eles até chegar ao minerador que incluirá a transação na Blockchain. Ao ser registrada na Blockchain e confirmada por um número subsequente suficiente de blocos (confirmações), atualmente são aceitos 6 blocos, a transação se torna parte permanente do registro do Bitcoin e é aceita como válida por todos os participantes.

Conforme discutido na Seção 2.2, a rede Bitcoin é uma rede P2P, as mensagens, incluindo transações e blocos, são propagadas de cada nó a todos os pares aos quais ele está conectado, por um processo chamado *flooding* (inundação).

O mecanismo de validação da transação do Bitcoin depende de dois tipos de scripts para validar as transações: um script de travamento e um de destravamento. A linguagem de script da transação Bitcoin contém muitos operadores, mas é deliberadamente limitada de uma maneira importante, não existem loops ou capacidades de controle de fluxo complexo, além do controle de fluxo condicional. Isso garante que a linguagem não é Turing completa, o que significa que os scripts tem uma complexidade limitada e tempos de execução previsíveis, evitando que o mecanismo de validação da transação seja usado como uma vulnerabilidade. Uma transação recém validada e injetada em qualquer nó da rede será enviada a todos os nós conectados a ele (vizinhos), cada um dos quais enviará a transação a todos os seus vizinhos, e assim por diante. Dessa maneira, dentro de alguns

segundos uma transação válida será propagada numa onda exponencialmente expandida através da rede até que todos os nós da rede a recebam[5].

Uma transação contém vários campos, como demonstrado na Tabela 2.1.

Tabela 2.1: Estrutura de uma Transação

Campo	Tamanho	Descrição
version	4 bytes	Identifica as regras que a transação segue
Input Counter	1-9 bytes	Identifica quantos inputs a transação tem
Inputs	Tamanho variável	O(s) input(s) da transação
Output Counter	1-9 bytes	Identifica quantos outputs a transação tem
Outputs	Tamanho variável	O(s) output(s) da transação
Locktime	4 bytes	Um timestamp UNIX ou um número de bloco a partir de quando/qual a transação poderá ser destrancada

Os *inputs* e *outputs* são os elementos fundamentais de uma transação. As transações são ligadas umas às outras por estes dois elementos. Os *inputs* de uma transação são, simplemente, referências aos *outputs* de uma transação anterior. Estes *outputs* prontos para serem usados por uma nova transação são chamados de UTXO - output de transação não gasta (do inglês *Unspent Transaction Output*) que são valores vinculados a uma determinada conta. O modo como as transações funcionam, faz com que elas formem uma sucessiva corrente de inputs e outputs trancando e destrancando valores na rede. Tendo como única exceção as chamadas transações *coinbase* que são as transações criadas pelos mineradores ao incluirem um novo bloco na blockchain e recolherem o prêmio pelo trabalho. A estrutura de dados das transações não tem um campo para taxas. Ao invés

[5]Os nós armazenam transações aguardando para entrar em um bloco em seu conjunto de memórias (ou *mempool*) após recebê-las. O *pool* de memória de um nó contém todas as transações de confirmação igual a zero em toda a rede que esse nó específico conhece. Quando os nós mineradores constroem novos blocos, eles preenchem seus blocos candidatos com transações tiradas do *pool* de memórias do nó local. É um equívoco comum que "o *mempool*"seja uma coisa única em toda a rede, quando, na realidade, é um conjunto de transações diferente (embora provavelmente semelhante) para cada nó. Exatamente quais transações entram no pool de memórias de um nó e quanto tempo elas permanecem lá, é uma política que varia de acordo com o nó. O comportamento de *mempool* do Bitcoin Core 0.16 é complexo, com muitas variáveis configuráveis pelo usuário, mas alguns destaques dos padrões são: As transações somente entram no pool de memórias se forem válidas e tiverem uma taxa acima de 1 satoshi/byte. As transações excluídas do *pool* de memórias têm seu hash salvo na memória para que não sejam baixadas repetidamente; O *mempool* é salvo nas reinicializações do nó, nas versões antigas, o mempool era salvo apenas na memória e seria apagado se o nó fosse reiniciado; O *mempool* aumentará para um tamanho máximo de 300 MB, se fosse maior do que isso, as transações de menor taxa por byte já existentes serão despejadas; As transações que estão no *pool* de memórias há 336 horas sem entrar em um bloco são despejadas; As transações no *mempool* podem ser substituídas por versões com taxas mais altas se as versões antigas sinalizarem substituição por taxa e a nova versão aumentar a taxa em pelo menos 1 satoshi/byte.[6]. Atualmente o tempo médio para uma transação ser aceita em um bloco minerado e adicionada a Blockchain é de 14 minutos de acordo com o gráfico do Tempo médio de confirmação em [7].

disso, infere-se que as taxas são a diferença entre a soma dos inputs e a soma dos outputs. Qualquer valor em excesso que permanece após todos os outputs terem sido deduzidos de todos os inputs é a taxa que é coletada pelos mineradores. A taxa é independente do valor em bitcoins da transação, elas são calculadas de acordo com o tamanho da transação em kilobytes,

Os UTXO são indivisíveis e precisam ser gastos de uma vez. Assim caso um usuário tenha uma conta com 20 Bitcoins e queira transferir 1 a outro usuário ele deverá realizar uma transação gastando os 20 Bitcoins, produzindo duas saídas (outputs) 1 para o usuário de destino e 19 como troco para ele próprio. Os UTXO consumidos por uma transação são chamados de entradas *(inputs)* de transação, e os UTXOs criados por uma transação são chamados de saídas *(outputs)*. de transação. Dessa maneira, partes de valor do Bitcoin movem adiante de dono para dono em uma cadeia de transações que consome e cria UTXOs. As transações consomem UTXOs quando os desbloqueiam com a assinatura do dono atual, e criam UTXOs quando os vinculam ao endereço Bitcoin do novo dono.

Os *outputs* são formados pelo valor da transação e pelo *script* de travamento, que define a condição necessária para que aquele valor possa ser usado novamente. Normalmente o *script* de travamento associa aquela UTXO a uma chave pública - endereço Bitcoin de destino - transferindo assim, a posse dessa quantia para o novo dono. Os UTXOs são monitorados por todos os nós completos de Bitcoin como um conjunto de dados chamado de conjunto UTXO ou *pool UTXO*, que é armazenado em um banco de dados. As novas transações consomem (gastam) um ou mais desses outputs que estão no conjunto UTXO.

Os *inputs* são referências aos UTXOs (outputs não gastos) contendo a resposta à condição necessária para gastar os UTXOs. Também necessitam de um *script* para destravar o uso do valor associado aquele UTXO, e somente o usuário que possui a chave privada correta conseguirá destravar o UTXO. Desta forma é possível que o sistema seja público, e ao mesmo tempo permita que somente os donos dos UTXOs possam usá-los.

A linguagem de script das transações bitcoin é uma linguagem de execução smples, que requer processamento mínimo. Os cinco tipos padrões de scripts de transação são:

- pagamento-para-hash-de-chave-pública *(pay-to-public-key-hash, P2PKH)* - A vasta maioria das transações processadas na rede bitcoin são transações P2PKH. Elas contém um script de travamento que onera o output com um hash de chave pública, mais comumente conhecido como um endereço bitcoin. As transações que pagam um endereço bitcoin contém scripts P2PKH. Um output travado por um script P2PKH

pode ser destravado (gasto) ao se apresenar a chave pública e a assinatura digital criada pela chave privada correspondente.

- chave-pública *(public-key)* - Com esse tipo de script, a própria chave pública é armazenada no script de travamento. Atualmente o pagamento-para-chave-pública é visto com maior frequência em transações *coinbase*, geradas por softwares de mineração mais antigos que não foram atualizados para utilizarem o P2PKH.

- múltiplas-assinaturas *(multi-signature)* - Scripts de múltiplas assinaturas definem uma condição onde N chaves públicas são registradas nos script e pelo menos M dessas devem fornecer assinaturas para liberar a oneração. Isso também é conhecido como um esquema M-de-N, onde N é o número total de chaves e M é o número de assinaturas necessárias para a validação.

- pagamento-para-hash-de-script *(pay-to-script-hash, P2SH)* - Foi desenvolvido para fazer a utilização de scripts complexos de uma maneira tão fácil quanto um pagamento para um endereço Bitcoin. Com pagamentos P2SH, o script de travamento complexo é substituído pela sua impressão digital, que é um hash criptográfico. Quando uma transação tentanto gastar o UTXO é apresentada mais tarde, ela deve conter o script que corresponde ao hash, além do script de destravamento. Em termos simples, P2SH significa "pagar para um script que corresponde a esse hash, um script que será apresentado mais tarde quando esse output for gasto". Nas transações P2SH, o script de travamento que é substituído por um hash é chamado de script de resgate *("redeem script")*, pois ele é apresentado ao sistema na hora do resgate ao invés de ser apresentado como um script de travamento.

- output de dados *(data output, OP_RETURN)* - A Blockchain, ou seja, todos os registros do Bitcoin, que são distribuídos e contém data e hora, possui usos potenciais muito além de pagamentos. Muitos desenvolvedores tentaram usar a linguagem de script de transação para tirar vantagem da segurança e da resiliência do sistema para aplicações como serviços de cartórios digitais, certificados de ações e contratos smart. As primeiras tentativas de se usar a linguagem de script do Bitcoin para essas finalidades envolveram criar outputs de transação que registravam dados na Blockchain; por exemplo, para registrar uma impressão digital de um arquivo de uma maneira que qualquer um pudesse estabelecer uma prova-de-existência daquele arquivo em uma data específica através de uma referência àquela transação. Outputs OP_RETURN são registrados na Blockchain, então eles consomem espaço em disco e contribuem para o aumento do tamanho da Blockchain, mas eles não são

armazenados no conjunto UTXO e portanto eles não ficam enchendo a *pool* de memória UTXO e não são um fardo para os nós completos, pois não exigem memórias RAM mais caras. O OP_RETURN foi inicialmente proposto com um limite de 80 bytes, mas o limite foi reduzido para 40 bytes quando a funcionalidade foi lançada. Em fevereiro de 2015, na versão 0.10 do Bitcoin Core, o limite foi novamente elevado para 80 bytes. Os nós podem decidir transmitir ou minerar OP_RETURN, ou apenas transmitir e minerar OP_RETURN com o tamanho menor que 80 bytes de dados.

Quando várias transações são transmitidas na rede, elas nem sempre chegam na mesma ordem. Às vezes, a transação filha pode chegar antes da pai. Nesse caso, os nós que enxergarem a filha primeiro irão ver que ela tem uma referência a uma transação pai que ainda não é conhecida. Ao invés de rejeitar a filha, eles a colocam em uma *pool* temporária para aguardar a chegada de sua transação pai e propagá-la para todos os outros nós. A *pool* de transações sem transações-pais é conhecida como a *pool* de transações órfãs. Assim que uma transação pai chega, quaisquer órfãs que tenham uma referência ao UTXO criado pela pai são liberadas, revalidadas recursivamente, e então toda a corrente de transações pode ser incluída em um *pool* de transações, pronto para ser minerado em um bloco. Existe um limite para o número de transações órfãs armazenadas na memória, para prevenir um ataque de negação de serviço (ataque DoS) contra os nós do Bitcoin.

Capítulo 3

Blockchain

A Blockchain[1] foi proposta por Satoshi Nakamoto [43], como uma abordagem inteligente a um problema inerente a sistemas descentralizados: confiança. Sua ideia inicial foi permitir que operações financeiras sejam realizadas de forma rápida, sem a necessidade de intermediários e com baixas taxas. É uma estrutura de dados ordenada composta por blocos de transações criptograficamente ligados por uma referência direta ao bloco anterior, servindo como livro-razão público no Bitcoin. A propriedade mais interessante da Blockchain para o consenso de um sistema descentralizado é este elo criptográfico que liga os seus blocos entre si, que é criado pelo esforço computacional do algoritmo de Prova de Trabalho (PoW, do termo em inglês *Proof-of-Work*) que será visto mais adiante.

Esta tecnologia tem como pilar três conceitos: Livro Razão Aberto (*Open Ledger*), descentralização e consenso. O primeiro conceito usado é o livro razão aberto, que é uma estrutura de dados que registra todas as transações. Um *ledger* é um livro razão digital que tem a propriedade de imutabilidade dos registros. Normalmente estes livros são mantidos por entidades centralizadoras e somente ela pode alterá-lo. Já o livro razão aberto pode ser escriturado por qualquer usuário. A descentralização permite que haja várias cópias do livro na rede, eliminando assim o ponto único de falha. Em virtude da característica pública e da descentralização, é necessário um mecanismo de consenso para sincronização das diversas cópias do livro e validação das transações escrituradas. Assim, o terceiro e último conceito usado é o consenso, onde alguns participantes especiais, denominados mineradores ou validadores[2], concorrem entre si para realizar três tarefas: validar as transações e incluí-las em um bloco, ligá-lo com o anterior e provar que foi o primeiro

[1] Quando refere-se a tecnologia, a rede ou ao sistema Blockchain, usa-se sempre inicial maiúscula. No entanto, quando a referência é ao banco de dados ou a cadeia de blocos (do inglês blockchain, utiliza-se a palavra em caixa baixa.

[2] Neste trabalho usaremos o termo *minerador* para nos referirmos a estes participantes.

minerador a calcular uma chave especial que faz o elo com o último bloco anterior da cadeia. Para encontrar esta chave, é necessário resolver um problema que, normalmente, requer grande poder computacional e uma quantidade de tempo significativa. O primeiro minerador que conseguir realizar estas três tarefas informará aos demais nós da rede que a chave foi encontrada e que o bloco de transações é válido. Assim, caso a solução esteja correta, todos os usuários atualizarão o seu próprio livro com as nova transações contidas neste bloco e a rede alcança um novo consenso.

As transações podem ser adicionadas a qualquer momento, mas apenas ao final do livro. Além disso, é garantido que as transações anteriores não podem ser removidas ou reordenadas e que todos os usuários honestos tenham uma visão consistente do livro razão. Para manter as transações de forma consistente e inalteradas, é utilizado como mecanismo de consenso a Prova de Trabalho. Embora existam outros mecanismos de consenso, como o bizantino, que possam ser usados para alcançar uma sequência de transações imutáveis e ordenadas, o algoritmo de prova-de-trabalho foi proposto por Nakamoto para ser o mecanismo de consenso da rede Bitcoin.

3.1 Bloco

A Blockchain é frequentemente visualizada como um empilhamento vertical, com os blocos empilhados uns sobre os outros e com o primeiro bloco servindo como fundação que suporta a pilha. A visualização dos blocos empilhados uns sobre os outros resulta no uso de termos como "altura"para se referir à distância em relação ao primeiro bloco, e "topo"ou "ponta"para se referir ao bloco adicionado mais recentemente.

Cada bloco contido na Blockchain é identificado no cabeçalho do bloco por um hash, que é gerado utilizando-se o algoritmo criptográfico de hash SHA256. Cada bloco também contém uma referência ao bloco anterior, conhecido como o bloco pai, através do campo "hash do bloco anterior"(previous blockhash) que existe no cabeçalho do bloco. Ou seja, cada bloco contém o hash de seu bloco-pai no interior de seu próprio cabeçalho. A sequência de hashes ligando cada bloco ao seus pais cria uma corrente que pode ser seguida retrogradamente até o primeiro bloco que já foi criado, que é conhecido como o bloco gênese.

Embora um bloco tenha apenas um bloco "pai", ele pode temporariamente ter múltiplos blocos "filhos". Cada um dos blocos "filhos"refere-se ao mesmo bloco "pai"e contém o mesmo hash (o hash do bloco"pai") no campo "hash do bloco anterior". Múltiplos

blocos filhos surgem quando há uma "bifurcação"ou "fork"da Blockchain, uma situação temporária que ocorre quando diferentes blocos são descobertos quase que simultaneamente por diferentes mineradores. No final, somente um bloco filho se tornará parte da Blockchain e a "bifurcação"deixará de existir. Apesar de cada bloco poder ter mais que um filho, cada bloco pode ter somente um pai. Isso ocorre porque um bloco possui apenas um único campo de "hash do bloco anterior", que é uma referência ao seu bloco pai único.

O campo "hash do bloco anterior"está dentro do cabeçalho do bloco e portanto afeta o hash do bloco atual. A identidade de um filho muda se a identidade de seu pai mudar. Quando o pai é modificado de qualquer maneira, o hash do pai muda. O hash modificado do pai exige uma mudança no apontador (hash do bloco anterior) do filho. Isso por sua vez faz com que o hash do filho mude, o que requer uma mudança no apontador do neto, o que por sua vez muda o hash do neto, e assim por diante. Esse efeito dominó garante que, assim que um bloco tenha muitas gerações o sucedendo, ele não pode ser modificado sem que haja um novo cálculo forçado de todos os blocos subsequentes. Como um novo cálculo exigiria um processamento computacional enorme, a existência de uma longa corrente de blocos faz com que a história profunda da Blockchain seja imutável, o que é uma característica chave para a segurança do Bitcoin.

De uma forma geral, o bloco é composto por um cabeçalho contendo metadados sobre o bloco e uma lista de transações que constituem a maior parte de seu tamanho, conforme demonstrado na Tabela 3.1. O cabeçalho do bloco tem 80 bytes, enquanto uma transação em média tem pelo menos 250 bytes e um bloco em média contém mais de 500 transações.

Tabela 3.1: Estrutura de um bloco

Campo	Tamanho	Descrição
Tamanho do Bloco	4 bytes	tamanho do bloco (block size) em bytes a partir deste campo
Cabeçalho do Bloco	80 bytes	o cabeçalho do bloco (block header) contendo metadados
Contador de Transações	1-9 bytes	número de transações neste bloco
Transações	Variável	as transações deste bloco

Por sua vez, o cabeçalho do bloco consiste de três conjuntos de metadados de bloco. Primeiro, existe uma referência ao hash do bloco anterior, que conecta esse bloco ao bloco anterior na blockchain. O segundo conjunto de metadados, a dificuldade, a data e hora (timestamp) e o nonce, está relacionado à competição da mineração. A terceira parte dos metadados é a raiz da árvore de merkle, uma estrutura de dados usada para resumir de maneira eficiente todas as transações contidas no bloco, conforme demonstrado na

tabela 3.2. Além deles, também é preciso entender dois metadados: altura do bloco e hash do cabeçalho, que são armazenados de forma a identificar o bloco e sua posição na cadeia.

Tabela 3.2: Estrutura do cabeçalho de um bloco

Campo	Tamanho	Descrição
Versão	4 bytes	número de versão do bloco indica que regras este bloco segue
Hash do Cabeçalho do Bloco Anterior	32 bytes	hash do cabeçalho do bloco anterior (parent block) a este na blockchain
Raiz de Merkle	32 bytes	hash da raiz de Merkle das transações deste bloco
Data e Hora (Timestamp)	4 bytes	a hora aproximada em que este bloco foi criado (emsegundos, usando Unix Epoch)
Dificuldade Alvo	4 bytes	dificuldade alvo do algoritmo de proof-of-work para este bloco
Nonce	4 bytes	um contador usado para o algoritmo de prova-de-trabalho

Cada bloco tem um identificador único que é criado a partir do hash de seu cabeçalho que serve tanto como identificador único deste objeto na rede quanto como prova de toda informação - incluindo as transações - contida nele. Para esta identificacão, necessita apenas do cabeçalho de cada bloco já que a Raiz de Merkle serve como prova de todas as transações incluidas no bloco da qual ela faz parte. Esta característica faz com que a Blockchain possa ser visualizada como uma corrente de blocos com os hashes do bloco anterior como elo criptográfico entre cada bloco. A Figura 3.1 exemplifica a cadeia de blocos de forma simplificada:

- **Altura:** Os blocos são incluídos na cadeia de forma linear em ordem cronológica, cada novo bloco recebe um número de ordem, a diferença entre o número do último bloco e o primeiro é chamada de altura. Este valor é um metadado e não faz parte do bloco. Não é um bom valor para identificar um bloco, pois pode haver, momentaneamente, dois ou mais blocos com a mesma altura.

- **Hash do cabeçalho:** É o principal identificador do bloco, é obtido ao realizar uma operação de resumo criptográfico sobre o cabeçalho do bloco. Ele Também não faz parte da estrutura de dados do bloco e também não é enviado pela rede junto com o bloco. Ele é computado isoladamente por cada nó completo no ato do recebimento de um novo bloco. É armazenado em um banco de dados separado como parte dos metadados do bloco. Ao contrário da altura, o hash do cabeçalho pode ser usado para identificar um bloco de forma inequívoca.

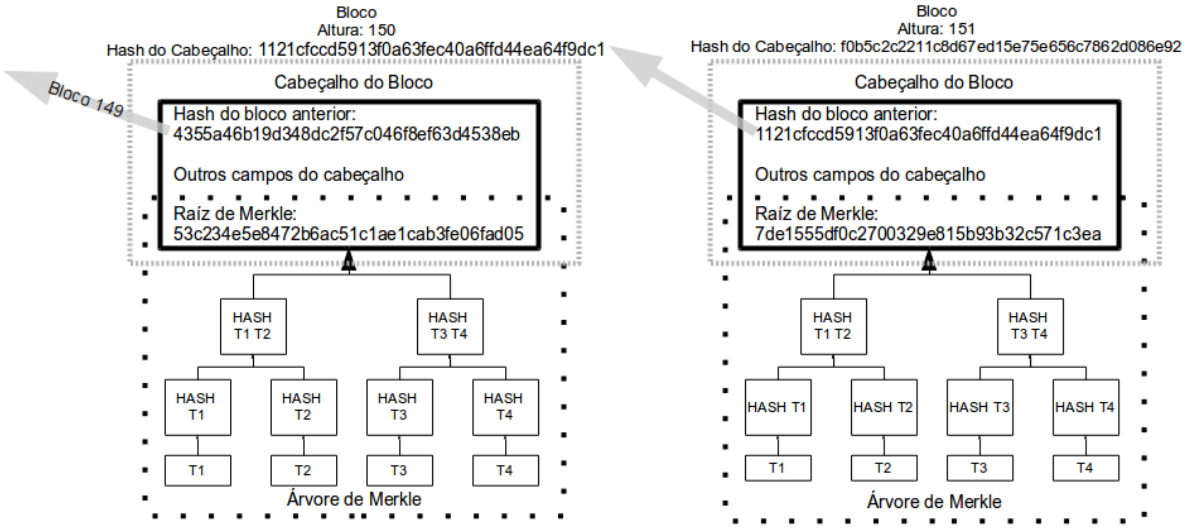

Figura 3.1: Estrutura Simplificada dos Blocos

- **Hash do bloco anterior:** Este campo é incluído no cabeçalho de modo a possibilitar que um novo bloco seja ligado ao anterior. Como vimos na Figura 3.1, o bloco 151 possui, em seu cabeçalho, o hash do bloco 150. Tão logo o bloco 152 seja recebido, por um nó completo, ele irá verificar este campo e definirá que o bloco 152 é filho do bloco 151.

- **Nonce:** Este campo é usado como uma variável. Serve para modificar a saída da função hash do cabeçalho. O nonce e a dificuldade são usados para provar que um minerador realizou a prova de trabalho e encontrou um cabeçalho que atenda aos critérios estabelecidos para essa prova.

- **Dificuldade:** A dificuldade nada mais é que uma colisão parcial de hash. Um algoritmo de hash gera sempre um mesmo resumo para uma determinada entrada. O Bitcoin usa o SHA-256 como algoritmo de hash, assim um valor de dificuldade deve ser também expresso com 256 bits. Quando a dificuldade é configurada para 1 bit (zero) basta achar um hash que inicie com um zero e qualquer valor para os outros 255 bits, ou seja, existem 2^{255} valores possíveis que são considerados válidos. Caso seja utilizada uma dificuldade com 2 bits a quantidade de valores possíveis será reduzida para 2^{254}. Com 10 bits serão 2^{246}, e assim por diante. É possível observar que há uma diminuição do espaço de possíveis valores que satisfaçam a colisão, implicando em maior dificuldade em achar um resumo válido. Assim, será necessário maior poder computacional ou mais tempo de mineração.

  A taxa pela qual novos blocos são incluídos na cadeia é definida pelos desenvolvedo-

res de cada projeto de Blockchain. Na rede Bitcoin foi estabelecido um alvo de 10 minutos. Assim, a dificuldade é ajustada por todos os nós completos e mineradores para que, em média, a cada 10 minutos um novo bloco seja incluído na cadeia. É esperado que novos mineradores se juntem a rede e novos equipamentos mais poderosos sejam lançados. Com isso, em média, o tempo de inclusão de novos blocos tende a diminuir. Para evitar que novos blocos sejam incluídos a intervalos menores que 10 minutos a dificuldade precisa ser ajustada, aumentando a quantidade de bits para a colisão, tornando mais demorado achar um novo hash. A cada 2016 blocos os nós recalculam a nova dificuldade. Como consequência, o tempo de inclusão de novos blocos irá se ajustar até ficar próximo ao alvo de 10 minutos. Para calcular a nova dificuldade os nós mineradores realizam a seguinte operação matemática:

$$Nova\ Dificuldade = Dificuldade\ Antiga * (\frac{Tempo\ 2016\ Blocos}{20160})$$

O reajuste do alvo da dificuldade ocorre automaticamente e em cada nó completo de maneira independente. A cada 2.016 blocos, todos os nós reajustam o alvo da dificuldade de prova de trabalho. A equação para reajustar o alvo mede o tempo que levou para encontrar os últimos 2.016 blocos e o compara com o tempo esperado de 20.160 minutos (duas semanas, baseando-se em um tempo desejado de 10 minutos para gerar um bloco). A razão entre os períodos de tempo atual e o desejado é calculada, e um ajuste correspondente (para cima ou para baixo) é feito à dificuldade. Se a rede estiver encontrando blocos mais rápido do que a cada 10 minutos, a dificuldade irá aumentar. E se a descoberta de blocos estiver mais lenta que o esperado, a dificuldade irá diminuir.

- **Raiz da árvore de merkle:** Uma árvore de Merkle [40], ou árvore de hash binário, é definida como uma árvore binária completa. Onde os dados são armazenados nas folhas e cada nó interior da árvore possui como valor o resultado do hash dos valores de seus filhos, em caso do bloco não ter um número par de transações, tudo que se faz é repetir a última transação. São projetadas para que um valor de folha possa ser verificado em relação a um valor de raiz conhecido publicamente. Para isso é necessário apenas fornecer os valores dos pares correspondentes no caminho da folha até a raiz. No Bitcoin ela é usada para resumir as transações contidas em um bloco e facilitar as operações de consulta. Cada hash deste tem o tamanho de 32 bytes e a complexidade de busca na árvore de Merkle cresce O(log2(N)) na notação "Big-O"com N sendo o número de transações. Para tal é necessário produzir $2 * \log_2 N$ hashes, para verificar se uma transação consta em um bloco, portanto ela fornece um processo muito eficiente. As transações não são armazenadas na árvore de merkle;

ao invés disso, seus dados são "hashed"e o hash resultante é armazenado em cada nó folha. Para construir esta árvore deve-se iniciar pelas folhas, que contêm os hashes das transações. As folhas são então agrupadas em pares e o hash do par produz um nó pai, repete-se o processo recursivamente até a raiz, chamada de raiz de merkle, conforme Figura 3.2.

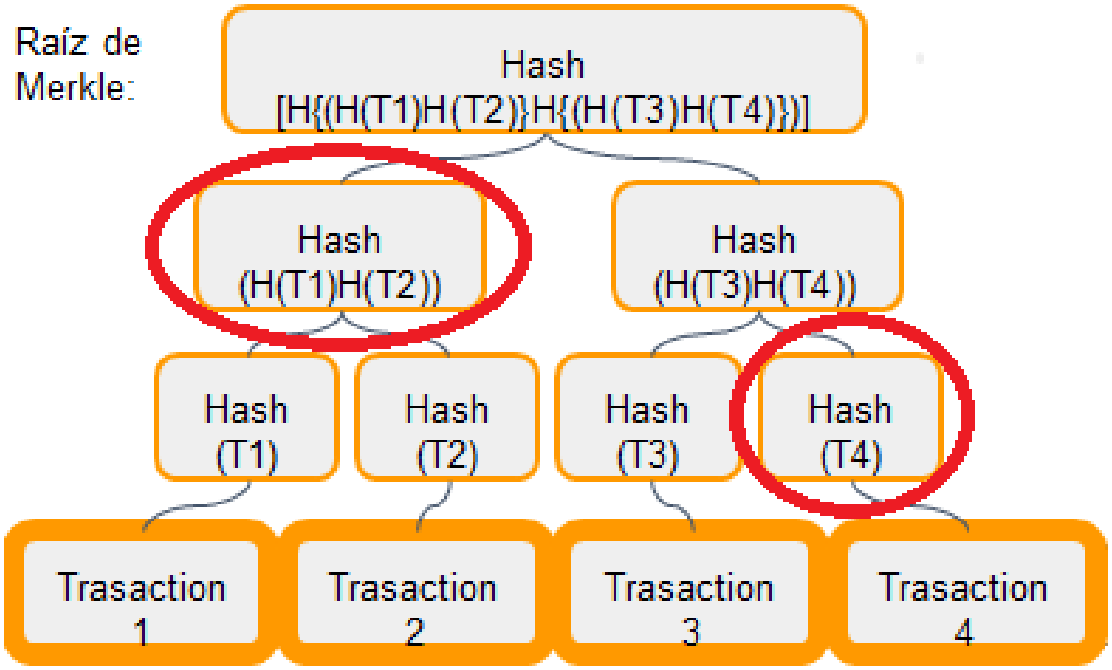

Figura 3.2: Árvore de merkle

Ao receber uma transação um nó simplificado precisa ter a certeza de que a transação recebida é válida, mas ele não possui a cadeia completa armazenada. Ele realiza uma consulta a um nó completo que responde com o cabeçalho do bloco e o caminho até a transação. Isto é particularmente útil pois para verificar se uma transação consta em um determinado bloco não é necessário solicitar o bloco inteiro à rede, e a árvore de Merkle permite a verificação rápida em meio a milhares de transações. Por exemplo, na Figura 3.2 cada folha corresponde a uma transação e os valores circulados em vermelho correspondem ao caminho para provar que a transação consta no bloco. Para provar que a transação 3 consta no bloco, o nó completo enviará o cabeçalho deste bloco e os Hash(T4) e Hash(H(T1)H(T2)) ao nó simplificado. De posse desses dados é possível calcular a raiz da árvore e comparar com o valor da raiz de merkle que consta no cabeçalho do bloco. O Nó simplificado fará o cálculo do Hash(T3) que juntamente com o Hash(T4) calculará o Hash(H(T3)H(T4)), e

por último, usa o valor do Hash(H(T1)H(T2)) para calcular a raiz, cujo valor é Hash[H(H(T1)H(T2)H(H(T3)H(T4))]. Desta forma o nó simplificado pode calcular o valor da raiz e provar que a transação é válida e está na cadeia e terá recebido menos que um kilobyte de dados que é mais do que mil vezes menor do que um bloco completo (cerca de 1 megabyte atualmente).

3.2 Mineração

A mineração é o processo responsável por atualizar a Blockchain, pelo qual alguns nós especiais, chamados de mineradores, incluem as transações em um bloco e geram um cabeçalho válido para essas transações. O processo de gerar um cabeçalho válido é encontrar o *nonce* que satisfaça a inequação: $SHA256(SHA256(data+nonce)) < Dificuldade$, onde *data* é o conteúdo do cabeçalho do bloco. Para gerar esse cabeçalho os mineradores devem calcular a árvore de merkle das transações, verificar a dificuldade estabelecida, incluir a estampa de tempo e realizar uma série de cálculos a fim de encontrar um nonce que satisfaça a dificuldade em vigor, fazendo com que o esforço de qualquer tentativa de mudança aos blocos da Blockchain cresça exponencialmente a cada bloco anterior que se tentar realizar a mudança; ao mesmo tempo em que a rede continua trabalhando adicionando novos blocos com mais prova-de-trabalho em cada um deles, fazendo a Blockchain cada vez mais segura e praticavelmente imutável.

O mineradores gastam muita energia para realizar a Prova de Trabalho, por esse motivo precisam ser recompensados. A primeira transação do bloco é sempre uma transação especial chamada de *Coinbase*. Na rede Bitcoin, a mineração tem dois propósitos, primeiramente, incluir novas moedas ao sistema (até atingir o limite de cerca de 21 milhões satoshis) e em segundo lugar proteger as transações realizadas. A mineração garante a segurança do sistema Bitcoin e permite a existência de um consenso em toda a rede sem a necessidade de uma autoridade central.

O minerador primeiro cria um "rascunho"de um bloco. É sobre esse rascunho que ele vai trabalhar até que obtenha um bloco viável para ser enviado a todos os nós da rede. O rascunho é a estrutura de dados que vai comportar os dados do cabeçalho e as transações. Após criar essa estrutura em branco, o minerador preenche alguns campos do cabeçalho: hash do bloco anterior, estampa de tempo, versão e dificuldade. Restando selecionar as transações, calcular a raiz da arvore de merkle, encontrar o nonce.

As transações, ao serem geradas, são enviadas via *broadcast* a todos os nós vizinhos

e estes reencaminham aos seus vizinhos. Os mineradores, ao receberem uma mensagem com uma transação, às armazenam em uma base de dados de transações ainda não mineradas. Elas permanecem temporariamente em uma de fila com prioridade até que sejam selecionadas para ser incluídas em um novo bloco. A prioridade da fila é dada pela taxa paga ao minerador e pelo tempo que a transação está na fila sem ser minerada. Assim, mesmo que haja transações com baixo valor, implicando em uma taxa pequena, em algum momento ela terá prioridade alta para ser minerada em razão do tempo. Cada minerador possui uma fila diferente de transações. Após selecionar quais transações serão incluídas ele irá gerar uma árvore de merkle para estas transações e incluir o valor da sua raiz no cabeçalho.

A etapa mais custosa do processo é encontrar o valor do nonce que fará parte do novo bloco. Esta etapa requer um grande poder computacional dos mineradores e consequentemente um enorme gasto de energia. Para exemplificar a dificuldade em encontrar o nonce, atualmente são comercializados dispositivos especializados em calcular hash. Esses dispositivos atingem a marca de 9TH/s, ou seja, conseguem calcular nove trilhões de hash por segundo. Com a dificuldade atual da rede Bitcoin e um dispositivo de 9TH/s seriam necessários, em média, 13 anos para achar um hash válido.

A seguir um exemplo de [13]: Para achar o nonce que produza um hash válido para "Simpósio Brasileiro de Segurança"com a dificuldade alvo de "000", ou seja o alvo é que o hash hexadecimal inicie com três zeros em sequência. para isso pode-se concatenar o nonce com a informação que será resumida - sha256("Simpósio Brasileiro de Segurança+nonce") - incrementando o nonce a cada insucesso até que seja encontrado um hash válido. Em um terminal do Linux é possível fazer:

for nonce in $(echo {0..10000}); do echo "$(echo "Simpósio Brasileiro de Segurança +$nonce sha256sum) $nonce"; done | grep ^000

Como resultado será obtido:

000a45ed0ebded66019dff14fc916c429aac6021494 e4983960e27c917696db5 - 4060

O que significa dizer que o nonce 4060 ao ser acrescido a "Simpósio Brasileiro de Segurança"gera um hash que atende a dificuldade alvo. É importante notar que existem outros valores de nonce que geram resultados válidos, como o 5470. A partir deste momento qualquer um que possua uma implementação do sha256 pode calcular o resumo de "Simpósio Brasileiro de Segurança+4060"e comparar com o hash fornecido, demonstrando

assim que o resultado é válido [3].

Assim que o nonce é encontrado o nosso rascunho fica completo e portanto o bloco está pronto para ser enviado a todos os nós da rede. Os nós da rede ao receberem um novo bloco iniciam uma série de verificações a fim de validar o bloco.

3.3 Mecanismos de consenso

A principal diferença entre as diversas implementações da Blockchain é o mecanismo de consenso. A maioria das plataformas Blockchain existentes, mais de 90% da capitalização do mercado total de moedas digitais, utilizam o mecanismo de consenso, na sua forma original e computacionalmente cara, chamada de Prova de Trabalho (PoW, do termo em inglês *Proof-of-Work*) [43]. Este conceito, inicialmente proposto por Adam Back [4], consiste em realizar uma colisão parcial de *hash*. É estabelecido que o *hash* do cabeçalho do bloco deve iniciar com uma determinada quantidade de zeros. Para que isso seja possível, o cabeçalho possui um campo especial chamado de *Nonce*, que é alterado constantemente até que o *hash* do cabeçalho atenda aos requisitos. O objetivo é provar que foi gasto tempo e energia para encontrar a solução da colisão. A principal desvantagem deste mecanismo é que ele é computacionalmente muito custoso. Existem vários outros mecanismos que oferecem certas vantagens desejadas em relação ao modelo original, como detalhado em [13]: Prova de Posse (PoS, do termo em inglês *Proof of Stake*); Algoritmo de Tolerância a Falhas Bizantinas (PBFT, do inglês *Practical Byzantine Fault Tolerance*) e a Prova do Tempo Decorrido (PoeT, do inglês *Proof of Elapsed Time*).

Segundo [35, 44], um protocolo de consenso tem três propriedades fundamentais com base nas quais sua aplicabilidade e eficácia podem ser determinadas:

- **Segurança:** Um protocolo de consenso é determinado seguro se todos os nós produzirem o mesmo resultado (*agreement*) e os resultados produzidos pelos nós são válidos de acordo com as regras do protocolo (*validity*). Isso também é referido como consistência do estado compartilhado.

- **Vivacidade:** Um protocolo de consenso garante a vivacidade se todos os nós que seguem o protocolo eventualmente produzem um valor (*termination*), ou seja, se um nó gerar uma transação e enviá-la a todos os nós da rede em algum momento um minerador irá incluí-la em um bloco.

[3] Esse número é o hash ou acompilação ("digest") da frase e ele é dependente de cada parte da frase. A adição de uma única letra, uma pontuação ou qualquer outro caracter irá produzir um hash diferente.

- **Tolerância a falhas:** Capacidade de continuar a operar, chegar ao consenso, adequadamente, mesmo após a falha de alguns nós da rede.

A principal invenção de Satoshi Nakamoto é o mecanismo descentralizado para o consenso emergente. Diz-se emergente, pois o consenso não é atingido de maneira explícita — não existe uma eleição ou um momento fixo no qual o consenso ocorre. Ao invés disso, o consenso é um artefato que emerge da interação assíncrona de milhares de nós independentes, todos seguindo regras simples. Todas as propriedades do Bitcoin, incluindo a moeda, as transações, os pagamentos e o modelo de segurança que não depende de confiança ou de uma autoridade central, derivam dessa invenção.

O consenso descentralizado do Bitcoin emerge da interação de quatro processos que ocorrem independentemente nos nós pela rede:

- Verificação independente de cada transação, realizada por cada nó completo - antes de propagar as transações para seus nós vizinhos, cada nó Bitcoin que recebe uma transação irá primeiro verificá-la. Isso garante que somente transações válidas serão propagadas para a rede, enquanto as transações inválidas serão logo descartadas pelo primeiro nó que as encontrarem. Ao verificar cada transação independentemente à medida que ela é recebida e antes de propagá-la, cada nó constroi um pool de transações válidas (mas ainda não-confirmadas) conhecido como pool de transações, pool de memória ou mempool.

- Agregação independente dessas transações em blocos novos pelos nós mineradores, associado à demonstração de que um processo computacional foi realizado através de um algoritmo de prova-de-trabalho - o nó minerador irá verificar todas as transações no pool de memória e removerá qualquer uma que já tiver sido incluída num bloco confirmado n. Quaisquer transações que permanecerem no pool de memória são transações não-confirmadas e estão aguardando para serem registradas em um novo bloco. O nó minerador imediatamente constroi um novo bloco vazio, um candidato para o bloco $n + 1$. Esse bloco é chamado de bloco candidato pois ele ainda não é um bloco válido, já que ele não contém uma prova de trabalho válida. O bloco só se torna válido se o minerador tiver sucesso em encontrar uma solução para o algoritmo de prova-de-trabalho.

- Verificação independente dos novos blocos por cada nó e inclusão deles em uma cadeia - cada nó ao receber um novo bloco o retransmite aos seus vizinhos, mas antes desta retransmissão o nó faz a validação do bloco a fim de garantir que somente

blocos válidos sejam propagados. Existe uma extensa lista de verificação a ser seguida, dentre elas: (estrutura do bloco; verificar se o hash do cabeçalho atende a dificuldade estabelecida; tamanho do bloco dentro dos limites projetados; verificação de todas as transações e verificação da estampa de tempo.

- Seleção independente, por cada nó, da cadeia de blocos com a maior computação acumulada demonstrada através da prova-de-trabalho.

Depois que um bloco candidato foi construído por um nó minerador, é hora do mecanismo de mineração [4] desse nó, ou do pool de mineração, minerar o bloco, ou seja, encontrar uma solução para o algoritmo de prova-de-trabalho que fará com que o bloco seja válido.

3.3.1 Prova de Trabalho

A ideia principal da Prova de Trabalho é tentar evitar ataques cibernéticos. Para atingir este objetivo, utiliza-se de um sistema onde o usuário deve provar que gastou um certo tempo para encontrar uma resposta que satisfaça a um requisito. A tarefa de encontrar tal resposta é baseada em dois princípios. Em primeiro lugar, a PoW tem que ser difícil e trabalhosa, mas não impossível; Segundo, a verificação dessa prova deve ser muito mais rápida e fácil de ser realizada. Este conceito foi inicialmente proposto por Adam Back [4] e é utilizado por diversos sistemas de prova e também pelo Bitcoin.

A Prova de Trabalho é o algoritmo que garante o consenso na rede Blockchain, através da solução de um problema criptográfico ou quebra-cabeça (um complicado enigma matemático e a possibilidade de provar rapidamente a solução). A resposta ao problema PoW ou a equação matemática é chamada de hash. Os mineradores resolvem o quebra-cabeça, formam o novo bloco e confirmam as transações. O quão complexo será um quebra-cabeça depende do número de usuários, da energia atual e da carga da rede. O hash de cada bloco contém o hash do bloco anterior, o que aumenta a segurança e evita qualquer violação de bloco. Se um minerador conseguir resolver o quebra-cabeça, o novo bloco é formado. As transações são colocadas neste bloco e consideradas confirmadas. O algoritmo de quebra-cabeça utilizado na rede Bitcoin é chamado de Hashcash. Este algoritmo permite mudar a complexidade de um quebra-cabeça com base no poder total da rede. O tempo médio

[4] É o processo de realização do hashing do cabeçalho do bloco de maneira repetida, até que o hash resultante corresponda a um alvo específico. Essa aplicabilidade das funções de hash significa que a única maneira de se produzir um resultado de hash que corresponde a um alvo específico é através de sucessivas tentativas, modificando aleatoriamente o input até que o hash resultante desejado apareça por acaso.

de formação do bloco é de 10 minutos.

Comumente a Prova de Trabalho é gerada da seguinte forma: O minerador constroi um bloco candidado preenchido com transações. Os blocos são compostos por cabeçalho e transações, e fazem parte do cabeçalho, o *hash* do cabeçalho do bloco anterior, a *dificuldade* e o *nonce*. A seguir, o minerador calcula o hash do cabeçalho desse bloco e verifica se ele é menor do que o alvo atual (dificuldade [5]). Se o hash não for menor do que o alvo, o minerador irá modificar o nonce [6] (geralmente somando o número 1) e tentará novamente. O minerador repete o procedimento variando o *nonce* até achar essa resposta. Como é relativamente difícil encontrar tal resposta, ao receber a mensagem contendo a resposta da PoW, todo usuário será capaz de verificar que houve um grande esforço do remetente em gerá-la. Ao decifrar o problema, o minerador gera um novo bloco.

Por exemplo, usando o sha-256:

Dificuldade:	000
Texto:	MídiaCom
Nonce:	4740
Hash hexadecimal :	00026e247cf15274ed5dee17d7c8b6d03b7ec3c88ad53f463f1fb4790c0c2405

Assim sha256(MídiaCom4740) gera um resultado que atende a dificuldade estabelecida, qual seja, um *hash* que inicie com 000.

O mecanismo de consenso da PoW é composto por duas etapas: validação do bloco e seleção da maior cadeia. Estas duas etapas são realizadas de maneira independente por cada nó. A primeira etapa é justamente verificar se o *hash* do cabeçalho atende a dificuldade estabelecida, conforme visto anteriormente isso prova que um minerador gastou tempo e poder de processamento para encontrar um *nonce* válido, além disso nesta etapa são validadas todas as transações do bloco [7].

[5] A dificuldade é uma medida da quantidade de esforço necessário para gerar um *hash* válido, que pode ser medido em termos da quantidade de zeros no início do *hash* criado

[6] Nonce é um número arbitrário que deve ser alterado sempre que o *hash* do cabeçalho não atenda a dificuldade estabelecida

[7] Na rede Blockchain são propagados 3 tipos de blocos: Os blocos inválidos são rejeitados assim que qualquer critério de validação falhar e, portanto, não são incluídos em nenhuma cadeia. Os blocos descartados (do inglês stale blocks) que foram extraídos com sucesso, mas que não estão incluídos na melhor cadeia de blocos (blockchain principal), provavelmente porque algum outro bloco na mesma altura teve sua cadeia estendida primeiro e os blocos órfãos (do inglês orphan blocks) cujo bloco pai ainda não foi recebido pelo nó local, por isso não pode ser totalmente validado. Os blocos órfãos geralmente surgem quando dois blocos que foram minerados em um curto espaço de tempo são recebidos em ordem inversa, ou seja, o bloco filho é recebido antes do bloco pai, entretanto, desde o Bitcoin Core v0.10, não existem mais blocos órfãos, devido a uma alteração significativa no mecanismo de download, onde os cabeçalhos são baixados e validados primeiro, antes mesmo que o nó solicite dados do bloco, como resultado, ele nunca receberá blocos cujos pais não conhece. Portanto na literatura atual fala-se sobre blocos órfãos, provavelmente refere-se a blocos descartados ou extintos.

A segunda etapa consiste em selecionar a cadeia mais longa. Isso se faz necessário pois, pode ocorrer uma situação em que um ou mais mineradores gerem novos blocos quase ao mesmo tempo, fazendo com que haja um ou mais blocos filhos que apontem para um mesmo pai. Neste caso, entende-se que ocorreu um *fork*, uma bifurcação, na cadeia. Esta etapa serve exatamente para selecionar qual destes blocos fará parte da cadeia principal e qual será descartado (os blocos da cadeia secundária). Como existem diversos mineradores gerando blocos de forma descentralizada, os novos blocos enviados por eles podem chegar a diferentes nós em momentos diferentes, o que pode resultar em visões diferentes. Os outros mineradores deverão escolher qual bloco eles irão adotar como referência [8]. Se uma parte dos mineradores adotar um bloco e outra parte adotar o outro, essas duas cadeias irão coexistir até que uma fique maior que a outra. Para resolver esta situação, os nós sempre irão adotar a maior cadeia. Assim, o *fork* estará resolvido e o consenso da cadeia definido entre os nós da rede. Na Figura 3.3 os blocos cinza bifurcaram da cadeia principal, e como alcançaram uma altura maior passaram a ser a cadeia principal. Os blocos brancos 127, 128 e 129 são descartados.

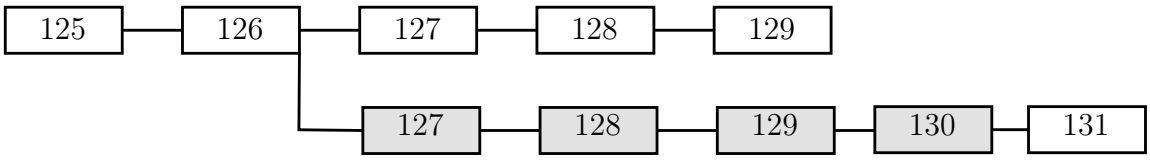

Figura 3.3: Bifurcação

Conforme descrito anteriormente, podem surgir *forks* de curta duração. Esses *forks* coexistem por um pequeno período de tempo (Entre 10 a 20 minutos, levando em consideração que o tempo médio de criação de um bloco é de 10 minutos). Quando o próximo bloco é minerado os nós escolhem a cadeia mais longa e descartam a outra[9]. A maioria dos forks ocorre naturalmente, devido a atrasos de propagação na rede, fazendo com que as poucas transações do lado descartado sejam atrasadas. Esta abordagem funciona bem,

[8] A segurança da PoW baseia-se no princípio de que nenhum usuário deve reunir mais de 50% do poder computacional da rede. Do contrário, esse usuário poderá efetivamente controlar o sistema, manipulando a cadeia mais longa.

[9] Um fork também pode ser provocado pelos próprios desenvolvedores e mantenedores de uma criptomoeda no momento em que eles decidem que as regras de validação de transações atuais devem sofrer algum tipo de alteração. Tal tipo de alteração nas regras de validação pode gerar dois tipos de fork, o soft fork e o hard fork. O soft fork pode ser definido como uma mudança no conjunto de regras que permite que blocos criados pelo novo conjunto de regras continuem sendo válidos. Em um hard fork os blocos criados com as novas regras passam a não ser válidos de acordo com a regra antiga, sendo criada "do zero"uma nova blockchain que passa a atender às novas regras da criptomoeda, com as regras antigas continuando a existir na versão prévia. Um hard fork mal realizado, contudo, poderia levar à total inutilização da blockchain de uma criptomoeda.

sob o pressuposto crucial de que nenhum atacante deve ser capaz de reunir tanto poder computacional que seja capaz de falsificar e publicar uma "cadeia alternativa" que tenha maior dificuldade total. Nesse caso, as regras de consenso fariam com que cadeia alternativa fosse adotada ao invés da cadeia honesta. O intervalo para a inclusão de novos blocos é crucial para a quantidade de *forks* observados na rede. Quanto menor for esse intervalo, maior será a quantidade de blocos gerados e consequentemente maior será a probabilidade de ocorrência de *forks*.

Tabela 3.3: Impacto do intervalo entre blocos na taxa de forks

Intervalo entre blocos	Taxa de Forks(%)	Mediana do tempo de propagação (minutos)
0,5s	38,15	0,82
1s	26,74	0,82
2s	16,65	0,84
5s	8,64	0,89
10s	4,77	1
20s	3,2	1,21
30s	2,54	1,43
1m	2,15	2,08
2,5m	1,82	4,18
10m	1,51	14,7
25m	1,72	35,73

Decker [17] verificou a existência de 169 *forks* durante sua observação de 10.000 blocos na rede Bitcoin, ou seja 1,69% dos blocos foram descartados pela ocorrência de *forks*. Gervais [23] analisou o impacto da diminuição deste tempo, variando a taxa de inclusão de novos blocos entre 0,5 segundos a 25 minutos, ele obteve seus resultados via simulações no NS-3, conforme apresentado na Tabela 3.3, onde vemos que quanto menor o intervalo entre blocos maior a quantidade de *forks* e a mediana do tempo de propagação do bloco é inversamente proporcional ao intervalo entre blocos.

Nos próximos diagramas, será demonstrado como ocorre o evento de um fork ao longo da rede. O diagrama é uma representação simplificada do Bitcoin como uma rede global. Na verdade, a topologia da rede Bitcoin não é organizada geograficamente. Ao invés disso, ela forma uma rede em malha de nós interconectados, que podem estar localizados geograficamente muito longe uns dos outros. A representação de uma topologia geográfica é uma simplificação usada com o objetivo de ilustrar um fork. Na rede Bitcoin verdadeira, a distância entre os nós é medida em "saltos" hops de um nó para o outro, e não em sua localização física. Para fins ilustrativos, diferentes blocos são exibidos com cores diferentes,

espalhados ao longo da rede e colorindo as conexões que eles cruzam.

No primeiro diagrama, a rede possui uma perspectiva unificada da Blockchain, com o bloco azul sendo o topo da cadeia principal, conforme visto na Figura 3.4 .

Figura 3.4: Visualização de um evento de fork da blockchain — antes do fork, adaptado de [3]

Um fork ocorre sempre que houver dois blocos candidatos competindo para formarem a blockchain mais longa. Isso ocorre sob condições normais sempre que dois minerados resolvem o algoritmo de prova de trabalho com uma diferença de tempo muito pequena. Quando ambos os mineradores descobrem uma solução para seus respectivos blocos candidatos, eles imediatamente transmitem o seu bloco ”vencedor” para seus vizinhos imediatos que começam a propagar o bloco pela rede. Cada nó que recebe um bloco válido irá incorporá-lo à sua blockchain, estendendo a blockchain em um bloco. Se aquele nó descobrir mais tarde outro bloco candidato estendendo o mesmo bloco pai, ele conectará o segundo candidato em uma cadeia secundária. Como resultado, alguns nós irão ”enxergar” um bloco candidato primeiro, enquanto outros nós irão enxergar o outro bloco candidato, surgindo duas versões concorrentes da blockchain.

Na Figura 3.5, dois blocos são encontrados simultaneamente, pode ser observado dois mineradores que mineram dois blocos diferentes quase que simultaneamente. Esses dois blocos são filhos do bloco azul, e são destinados a estender a cadeia ao serem adicionados no topo do bloco azul. Para facilitar o acompanhamento, um deles é visualizado como um bloco vermelho se originando do Canadá, e o outro é marcado como um bloco verde se originando da Austrália.

Agora, existem dois blocos possíveis, um ”vermelho”, se originando no Canadá, e outro ”verde”, se originando na Austrália. Ambos os blocos são válidos, ambos os blocos contém uma solução válida para a prova de trabalho e ambos os blocos estendem o mesmo pai. Provavelmente a maioria das transações de ambos os blocos é semelhante, com apenas algumas diferenças na ordem das transações.

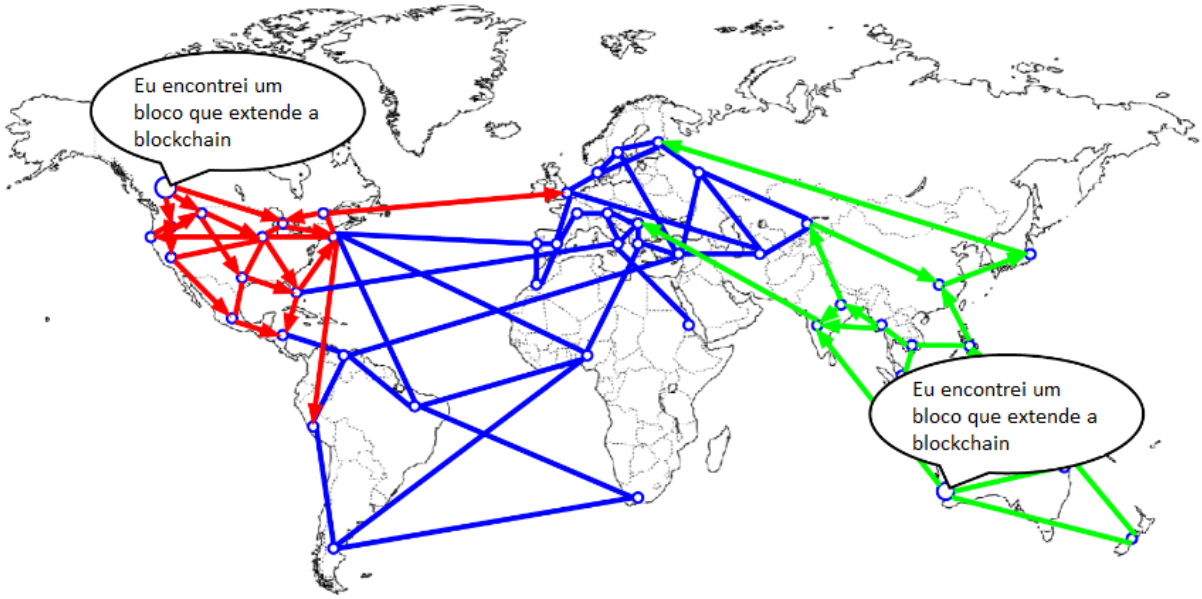

Figura 3.5: Visualização de um evento de fork da blockchain: dois blocos são encontrados simultaneamente, adaptado de [3]

À medida que os dois blocos são propagados, alguns nós recebem o bloco ”vermelho” antes e alguns recebem o bloco ”verde” antes. Como demonstrado na Figura 3.6, a rede se bifurca em duas perspectivas diferentes da blockchain, um lado coberto com um bloco vermelho, e outro lado coberto com um bloco verde.

A partir desse momento, os nós da rede Bitcoin que estiverem mais próximos (topologicamente, e não geograficamente) ao nó canadense irão ficar sabendo sobre o bloco ”vermelho” antes e irão criar uma nova blockchain com a maior dificuldade acumulada, que tem o bloco ”vermelho” como último bloco na cadeia (por exemplo, azul-vermelho), ignorando o bloco candidato ”verde” que chegar um pouco depois. Enquanto isso, os nós mais próximos do nó australiano irão considerar aquele bloco como o vencedor e irão estender a blockchain com o bloco ”verde” como o último bloco (por exemplo, azul-verde), ignorando o bloco candidato ”vermelho” quando ele chegar alguns segundos depois. Qualquer minerador que enxergar o bloco ”vermelho” antes irá imediatamente construir blocos candidatos que usam o "vermelho"como referência para o pai, e começarão a tentar resol-

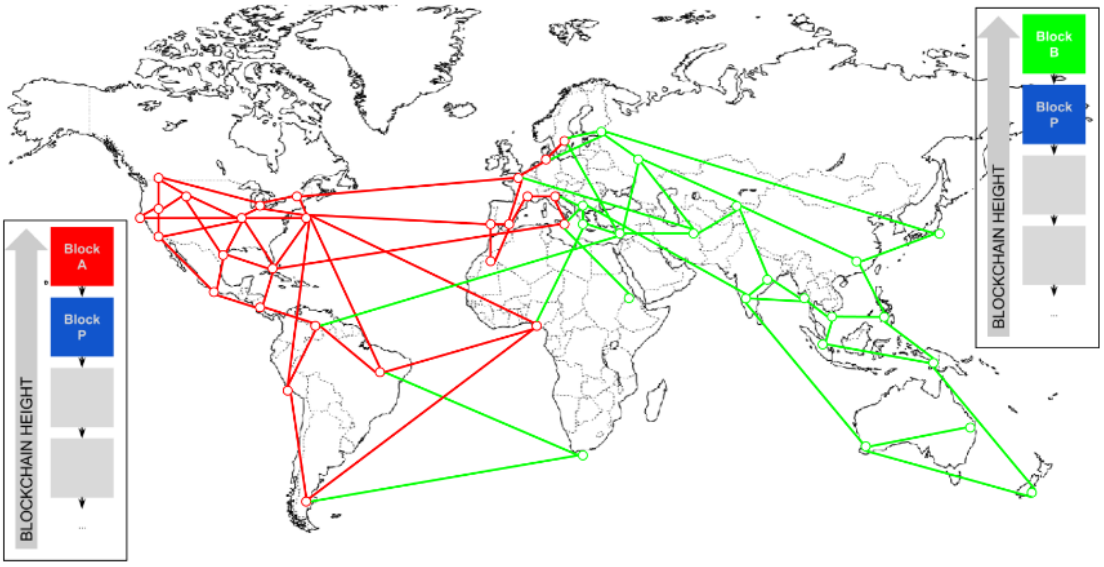

Figura 3.6: Visualização de um evento de fork da blockchain: dois blocos são propagados, dividindo a rede, adaptado de [3]

ver a prova de trabalho para esses blocos candidatos. Por outro lado, os mineradores que aceitaram o bloco ”verde” irão começar a construir no topo do ”verde”, estendendo aquela cadeia.

Os forks quase sempre são resolvidos em um bloco. Como parte do poder de hashing da rede é dedicado a construir no topo do ”vermelho” como pai, outra parte do poder de hashing é focado em construir no topo do ”verde”. Mesmo que o poder de mineração esteja dividido quase igualmente, é provável que um conjunto de mineradores encontre uma solução e propague antes que outro conjunto de mineradores consiga encontrar algumas soluções. Por exemplo, supondo que os mineradores construindo no topo do ”verde” encontre um novo bloco ”rosa” que estenda a corrente (por exemplo, azul-verde-rosa). Eles imediatamente propagam esse bloco novo e toda a rede o enxerga como uma solução válida, como demonstrado na Figura 3.7.

Todos os nós que escolheram o ”verde” como vencedor na rodada anterior irão simplesmente estender a cadeia em mais um bloco. Os nós que escolheram o ”vermelho” como vencedor, entretanto, enxergarão agora duas cadeias: azul-verde-rosa e azul-vermelho. A cadeia azul-verde-rosa agora é maior (mais dificuldade acumulada) do que a cadeia azul-vermelho. Como resultado, aqueles nós irão definir a cadeia azul-verde-rosa como a cadeia principal e passarão a considerar a cadeia azul-vermelha como sendo a cadeia secundária, como demonstrado na Figura 3.8. Isso se chama de reconvergência da cadeia, pois aqueles nós são forçados a revisar sua visão da blockchain para incorporar a nova evidência

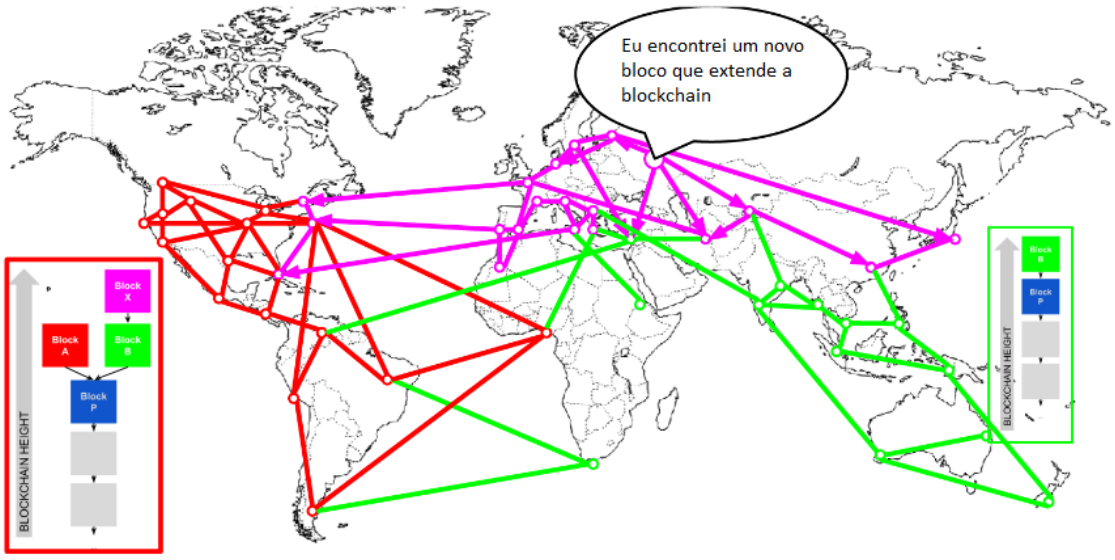

Figura 3.7: Visualização de um evento de fork da blockchain: um novo bloco estende uma dos forks, adaptado de [3]

de uma cadeia mais longa. Qualquer minerador que estiver trabalhando em estender a cadeia azul-vermelho terá que parar o trabalho porque o seu bloco candidato é um ”órfão”, já que o seu pai ”vermelho” não está mais na cadeia mais longa. As transações contidas no "vermelho"são enfileiradas novamente para serem processadas no próximo bloco, pois aquele bloco não está mais na cadeia principal. Toda a rede reconverge para uma única blockchain azul-verde-rosa, com o ”rosa” sendo o último bloco na cadeia. Todos os mineradores imediatamente começam a trabalhar nos blocos candidatos que referenciam o ”rosa” como seu pai, para estender a cadeia azul-verde-rosa.

É teoricamente possível que um fork se estenda em dois blocos, se os dois blocos forem encontrados quase que simultaneamente por mineradores nos ”lados” opostos do fork anterior. Entretanto, a chance disso acontecer é muito baixa. Enquanto um fork de um bloco pode ocorrer semanalmente, uma bifucarcação de dois blocos é extremamente rara.

O intervalo de bloco de 10 minutos do Bitcoin é um ajuste equilibrado entre tempos de confirmação rápidos (liquidação das transações) e a probabilidade de um fork. Um tempo de bloco mais rápido faria com que as transações fossem liquidadas mais rapidamente, mas isso causaria forks na blockchain mais frequentes, enquanto um tempo de bloco mais lento iria diminuir o número de forks, mas faria com que as transações fossem liquidadas mais lentamente.

Existem vários outros mecanismos que oferecem certas vantagens desejadas em relação

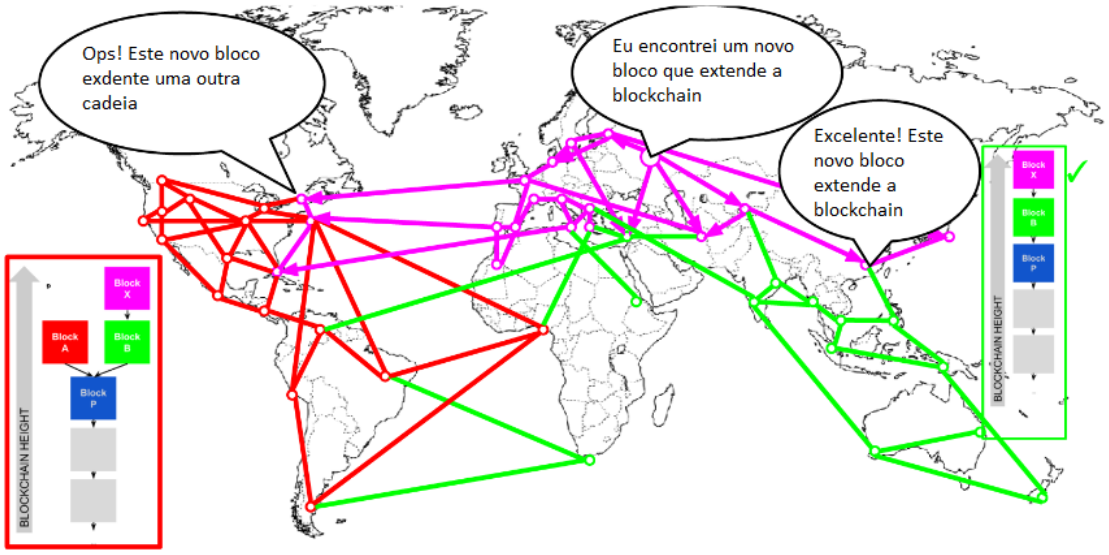

Figura 3.8: Visualização de um evento de fork da blockchain: a rede reconverge em uma nova cadeia mais longa, adaptado de [3]

ao modelo original, como a prova de Prova de Posse [37], o Algoritmo de Tolerância a Falhas Bizantinas [12] e a Prova do Tempo Decorrido [30] que aparecem como outras alternativas e serão brevemente explicadas a seguir.

3.3.2 Prova-de-Posse

A Prova de Posse é utilizada em plataformas Blockchain, incluindo criptomoedas. A PoS requer que um minerador prove a posse de uma certa quantidade de moeda para sua participação. O criador do próximo bloco é escolhido de uma maneira probabilística, e a chance de uma conta ser escolhida depende da "riqueza" (ou seja, da posse). Os algoritmos de Prova-de-Posse são projetados para superar as desvantagens dos algoritmos PoW em termos do alto consumo de energia envolvido nas operações de mineração. Dito de outra forma, em vez de um usuário gastar R$ 2000,00 comprando equipamentos de mineração para iniciar os trabalhos no algoritmo PoW e ganhar uma recompensa pela mineração, o mesmo usuário, utilizando a PoS pode comprar R$ 2000,00 em criptomoedas e usá-las como participação para comprar cotas de criação de blocos, tornando-se um validador[10]. Em sistemas que usam PoS, os blocos costumam ser validados, em vez de minerados. Essas implementações não oferecem incentivos para que os nós votem no bloco correto. Portanto, os nós podem votar em vários blocos que suportam vários *forks* para maximizar suas chances de ganhar uma recompensa. Neste caso não gastam nada ao fazê-lo em

[10]No mecanismo de consenso de Prova-de-Posse o minerador é conhecido como validador

oposição a PoW, onde o nó dividiria seus recursos para votar em múltiplos *forks*. A seleção pelo saldo da conta resultaria em centralização (indesejável), pois o usuário mais "rico" teria uma vantagem permanente. Vários métodos de seleção diferentes foram planejados, por exemplo, Nxt [36] e BlackCoin [62] usam aleatorização para prever o próximo gerador de blocos, usando uma fórmula que procura o menor valor de *hash* em combinação com o tamanho da participação. Uma vez que as apostas são públicas, cada nó pode prever, com precisão razoável, qual conta ganhará o direito de validar um bloco.

3.3.3 Algoritmo de Tolerância a Falhas Bizantinas

Uma das funções de um protocolo de consenso é manter a ordem das transações em uma rede de cadeias de blocos, apesar das ameaças a essa ordem. Uma dessas ameaças é a falha arbitrária simultânea, um dos tipos de falha bizantina, de múltiplos nós de rede. Usando PBFT, uma rede de nós Blockchain pode tolerar que uma fração dos nós operem em falha. O algoritmo PBFT funciona da seguinte forma:

- Um cliente envia uma solicitação de serviço para a máquina primária.
- A primária replica o pedido para os backups.
- As réplicas executam o pedido e enviam respostas.
- O cliente aguarda $f + 1$ respostas idênticas de réplicas diferentes para considerar um resultado correto, onde f é a quantidade de nós em falha na rede.

Como o número total de nós precisa ser conhecido, o PBFT não é adequado para sistemas públicos, sendo utilizado apenas em sistemas privados. Uma rede PBFT garante a consistência e a integridade dos dados quando ocorrem falhas bizantinas em até 1/3 dos nós da rede. Por exemplo, usando PBFT, uma rede de cadeia de blocos de nós N pode suportar f número de nós Bizantinos, onde $f = (N - 1)/3$. Em outras palavras, o PBFT garante que um mínimo de $2 * f + 1$ nós alcancem consenso sobre a ordem das transações antes de anexá-las à Blockchain.

3.3.4 Prova do Tempo Decorrido

É um algoritmo de consenso, projetado pela Intel. PoET usa um modelo de eleição aleatória de um líder, que irá validar os blocos. Funciona essencialmente da seguinte forma. Existe um *hardware* especializado em gerar um valor de tempo aleatório. Cada

validador solicita um tempo de espera a este *hardware*. O validador com o tempo de espera mais curto para um determinado bloco é eleito o líder, e espera este tempo para validar o bloco. Após este bloco ser incluído na cadeia, o processo se repete. Este modelo é proposto para uso em Blockchains privados, pois, em teoria, os validadores são honestos. A aleatoriedade na geração de tempos de espera garante que a função líder seja distribuída de forma uniforme e entre todos os validadores. Uma desvantagem desse algoritmo é a dependência de hardware especializado.

3.4 Classificação das Blockchains

Blockchain pode ser classificada com base no acesso aos dados e na participação do mecanismo de consenso sobre quaisquer mudanças propostas no seu livro razão. Podendo ser:

- **Blockchain sem permissão ou (Pública):** O mecanismo de consenso está aberto a todos. O objetivo de uma cadeia sem permissão é possibilitar que qualquer pessoa contribua com dados. Isso cria a chamada resistência da censura, o que significa que nenhum ator pode evitar que uma transação seja adicionada à cadeia. Os participantes mantêm a integridade da cadeia ao chegar a um consenso quanto ao seu estado. Qualquer um pode se juntar à rede e participar do processo de verificação de blocos para criar consenso e também criar contratos inteligentes. Ter um sistema sem permissão implica assumir que pode não haver confiança entre os nós, portanto, um mecanismo de consenso fortemente distribuído deve ser imposto. Em tal sistema, existe a possibilidade de um ataque Sybil [20], onde um nó de rede tenta aparecer como vários nós distintos criando um grande número de pseudoidentidades. Uma influência desproporcionalmente grande por um único nó é uma ameaça, então a introdução do PoW na validação da transação é logicamente justificada e necessária.

- **Blockchain permissiva ou (Privada):** Participantes no processo de consenso estão pré-selecionados. Quando um novo registro é adicionado, a integridade do livro razão é verificada por um processo de consenso realizado por um número limitado de atores confiáveis. Isso torna a manutenção de um registro compartilhado muito mais simples do que o processo de consenso sem permissão. As cadeias de blocos permitidas fornecem conjuntos de dados altamente verificáveis porque o processo de consenso cria uma assinatura digital, que pode ser vista por todas as partes. As características que derivam de sistemas confiáveis podem abrir a possibilidade de

evitar um protocolo de consenso computacionalmente exigente, como a PoW.

Muitos projetos foram iniciados para tornar a Blockchain mais popular e viável para diferentes modelos de negócios e aplicações, aproveitando as categorias existentes. A Tabela 3.4 resume as principais características de algumas aplicações que usam Blockchain. Bitcoin e Ethereum [65] são exemplos de Blockchain sem permissão, já os Hyperledger [10] e Ripple [49] são exemplos de Blockchain com permissão. É possível verificar uma diferença crítica entre essas duas categorias que é o modelo de mineração. Blockchains sem permissão usam a Prova de Trabalho, onde o poder computacional é oferecido para criar confiança. Como todos os atores, nas Blockchains permissivas, são conhecidos não é necessário usar um mecanismo custoso para alcançar o consenso, o que acarreta em um menor tempo de processamento de bloco, sendo considerado, praticamente, em tempo real.

Tabela 3.4: Comparação entre sistemas Blockchain

Blockchain	**Bitcoin**	**Ethereum**	**Hyperledger**	**Ripple**
Natureza	Sem Permissão	Sem Permissão	Permissiva	Permissiva
Validação	PoW SHA-256	PoW - ethash	PBFT	BFT customizado
Propósito	criptomoeda	contrato inteligente	Chaincode	criptomoeda
Linguagem	Scripts em pilha	Código interno Turing completo	Go, Java	C++
Tempo de processamento do bloco	$\sim$ 600 s	$\sim$ 15 s	$\sim$ tempo real	$\sim$ tempo real

Também é possível classificar a Blockchain com base em sua evolução, sendo dividida em três fases [57]:

- A Blockchain 1.0 é o uso para criptomoedas, transferências e sistemas de pagamento digital, amplamente difundido pelo Bitcoin e criptomoedas derivadas.
- A Blockchain 2.0 é o uso com contratos inteligentes [11], toda a lista das questões econômicas, mercado e aplicações financeiras que a utilizam de maneira mais extensa, como: ações, títulos, empréstimos, hipotecas.
- A Blockchain 3.0 refere-se o seu uso em aplicações além da moeda, finanças e mercados, particularmente nas áreas de governo, saúde, ciência, Internet das Coisas.

[11] Os Contratos Inteligentes são *scripts* armazenados na Blockchain e executam suas instruções de maneira distribuída em todos os participantes do contrato. Os Contratos Inteligentes possuem um endereço, e são acionados endereçando uma transação para ele. Em seguida, é executado de modo independente e automático da forma prescrita em cada nó da rede, de acordo com os dados que foram incluídos na transação desencadeadora.

Diversas soluções estão sendo propostas para o uso da Blockchain 3.0, dentre as quais é possível citar: Suportar Ambientes Inteligentes [19]; estabelecer mecanismos de pagamento para IoT [66]; realizar Computação Segura entre Múltiplos Participantes (MPC) [73]; e, criar serviços de Infra-Estrutura Pública de Chaves (PKI) [46, 2].

3.4.1 Casos de Uso

O Bitcoin é apenas uma das muitas aplicações da tecnologia Blockchain. O aumento do interesse pela Blockchain ocorreu em virtude de seu uso crescente em serviços financeiros. No entanto, muitas outras áreas estão começando a se beneficiar de suas características, tais como, transparência, descentralização, integridade, auditabilidade e não repúdio. Que possibilitam o desenvolvimento de registros de dados seguros. Essas características são fundamentais para grande parte das aplicações que usam a Blockchain. A estes usos nos referimos como *Blockchain 3.0*. Diversas soluções estão sendo propostas para o uso da Blockchain 3.0, dentre as quais é possível citar:

- **Suportar Ambientes Inteligentes:** As abordagens baseadas em Prova de Trabalho oferecem segurança, descentralização e privacidade, mas envolvem consumo excessivo de energia e atrasos que não são adequados para a maioria dos dispositivos IoT. Dorri [18, 19] apresenta uma solução leve de um Blockchain especialmente orientada para uso com IoT, eliminando a Prova de Trabalho e o conceito de moedas. Sua abordagem foi exemplificada em uma implementação em Ambientes Inteligentes (*Smart Home*) e consiste em três estruturas principais: armazenamento em nuvem, uma camada de sobreposição e casa inteligente. Cada casa inteligente está equipada com um dispositivo com maior poder computacional que está sempre *on-line*. Este dispositivo é chamado de "minerador", e é responsável por lidar com todas as comunicações dentro e fora da casa. O minerador mantém uma Blockchain privada, usada para controlar e auditar as comunicações e prover controle de acesso entre os dispositivos. Neste cenário uma Prova de Trabalho custosa torna-se desnecessária, pois somente um dispositivo terá o trabalho de manter a Blockchain. Os demais dispositivos da casa recebem um par de chaves para que possam realizar transações.

- **Estabelecer mecanismos de pagamento para IoT:** Em [67] os autores descreveram uma implementação prototípica simples do processo de troca de dados por dinheiro eletrônico entre um sensor e um requisitante, utilizando a rede Bitcoin. Seguindo a mesma linha, [71] propõe uma arquitetura de comércio eletrônico projetada

especificamente para uso em IoT. Uma parceria entre a IBM e a Samsung desenvolveu ADEPT [48] - Telemetria Autônoma Ponto-a-Ponto descentralizada (do inglês, Automated Decentralized Peer-to-Peer Telemetry). Utiliza a Blockchain para fornecer a espinha dorsal do sistema. Esta plataforma foi testada em vários cenários, incluindo um que envolve uma máquina de lavar inteligente que pode automaticamente comprar e pagar por detergente. O Filament [16] é um sistema desenvolvido para permitir que os dispositivos tenham identidades únicas em um livro público e possam descobrir, comunicar e interagir uns com os outros de forma autônoma e distribuída. Além disso, os dispositivos envolvidos podem trocar valor diretamente ou indiretamente.

- **Realizar Computação Segura entre Múltiplos Participantes (MPC):** O MPC é a generalização do problema dos milionários proposto por Yao [69]. O problema proposto Yao é resolvido com algoritmos para computação segura entre dois participantes. O problema para múltiplos participantes é definido como: N participantes calcularem uma função com suas entradas de forma segura, onde a segurança significa garantir o resultado correto e a privacidade das entradas, mesmo com a presença de alguns participantes maliciosos. Ao final, cada participante obterá apenas o resultado da função e não conhecerá as entradas dos demais participantes. Seu uso abre caminho para diversas aplicações como, votação na internet, e mineração e compartilhamento de dados. Para realizar estas funções, é necessários que os participantes troquem mensagens. Mas, as trocas de mensagens crescem exponencialmente com o número de participantes. Para realizar funções que operam multiplicações, a quantidade de troca de mensagens é da ordem de $O(n^2)$. Este fato torna a implementação da MPC restrita a poucos participantes e cenários específicos. Ao longo dos anos surgiram propostas para otimizar as soluções e aumentar a quantidade de participantes [15, 22, 73].

  Em [73] é criada uma plataforma chamada de Enigma, que realiza MPC usando uma Blockchain para controlar a rede, gerenciando o controle de acesso e servindo como log de eventos para o compartilhamento dos segredos. Yue [70] usa a Blockchain para realizar controle de acesso e armazenamento dos dados dos pacientes. O autor considera que o uso dos dados dos pacientes sem seu consentimento é um ataque à privacidade, mas também descreve a importância da utilização destes dados para a pesquisa médica. Ele classifica os dados em dois tipos: privados e públicos. Qualquer pesquisador ou entidade governamental poderá utilizar os dados públicos. Os dados privados somente poderão ser utilizados via MPC. Assim, o uso de MPC

torna possível saber, por exemplo, a quantidade de pacientes que possuem AIDS e pertencem a grupos de risco. Torna possível extrair conhecimento sobre esses dados sem revelar a privacidade dos pacientes.

- **Criar serviços de Infra-Estrutura de Chaves Públicas (PKI) e sistema de domínio de nomes (DNS):** A Blockchain não possui ponto central de falha e não é governada por uma única entidade, ela permite uma nova classe de aplicativos e serviços descentralizados, como por exemplo, um servidor raiz DNS ou uma autoridade de certificação raiz. Esses benefícios motivaram Ali [2] a usar a Blockchain para construir um novo sistema de PKI descentralizado e um sistema de identidade, chamado *Blockstack ID*. Hari [26] propôs o uso de um mecanismo baseado em Blockchain para proteger a infraestrutura BGP e DNS da Internet. As principais vantagens dessa abordagem incluem: a eliminação do nó raiz de confiança PKI e um registro de histórico de transações verificável e distribuído.

- **Sistema eletrônico de votação:** Em [47] a Blockchain funciona como um banco de dados de transações seguras, para registrar votos e resultados de votação. O método proposto envolve as árvores de Merkle para verificar a listas de eleitores e exploradores de blocos para realizar a apuração dos votos. Zhichao [72] projetou um protocolo para votação usando Bitcoin e MPC, garantindo a privacidade do voto individual.

Capítulo 4

Principais Ataques à Blockchain

No final de 2017, a criptomoeda Bitcoin ficou em evidência nas manchetes. Seu valor atingiu um pico de quase US$ 20.000 por moeda, chamando a atenção das principais agências de notícias e de potenciais investidores. A principal criptomoeda, Bitcoin, baseia-se na Blockchain, uma tecnologia nova e revolucionária, que registra transações de uma maneira descentralizada, e que começou a mudar a forma de relacionamento com o dinheiro, oferecendo um caminho para resolver velhos problemas de negócios de maneiras novas. Porém, novas tecnologias trazem novas preocupações com a segurança. Elementos maliciosos já atacaram diversas implementações de Blockchain utilizando engenharia social, malware e explorações. À medida que mais grupos começam a utilizar a Blockchain e a construir ferramentas ao seu redor, eles começam a compreender os riscos à segurança. Neste capítulo será abordado os problemas de segurança atuais e incidentes específicos dentro das implementações de Blockchain.

4.1 Ataques de negação de serviço (DoS)

Nas Blockchain sem permissão qualquer nó pode criar um bloco, e teoricamente alterar blocos de uma cadeia. As transações são consideradas válidas quando estão em um bloco e confirmadas sempre que exista uma certa quantidade de blocos com altura maior na cadeia. Isso se deve ao fato de que para alterar uma transação é necessário possuir grande poder computacional para alterar o bloco onde a transação se encontra e os posteriores. Assim, existe uma quantidade de blocos que torna computacionalmente impossível alterar a cadeia. Para criar blocos os nós recebem alguns incentivos econômicos. Parte desses incentivos é dada pelas taxas pagas pelos usuários em cada transação. Além de remunerar os nós que gastam recursos para criar os blocos, as taxas tem um outro papel: evitar

ataques de negação de serviço. Um atacante até poderia inundar a rede com transações, mas ficaria caro demais, pois ele necessitaria pagar taxas.

Ao contrário dos sistemas centralizados, os atacantes em uma rede Blockchain estão lidando com um grande número de nós que possuem o estado atual da Blockchain. Então, basicamente, pode-se concluir que os ataques DoS são muito difíceis de serem executados em uma rede peer-to-peer, a fim de comprometer a segurança de toda a rede e da blockchain.

Em uma rede peer-to-peer, um nó pode se conectar a outros nós de duas maneiras diferentes, ou seja, conexões de entrada e saída. Conexões de saída são aquelas em que um nó se inicia, solicitando conexões de outros nós, em que confia (ou não) pedindo para estabelecer uma conexão. Pelo contrário, conexões de entrada são aquelas recebidas dos outros nós solicitando uma conexão. Desta vez, são os outros nós que fazem a solicitação. Em muitos casos, esses nós estão completamente aleatórios. Isso ajuda a criar uma topologia dinâmica que torna o ataque DoS mais complexo.

Existem maneiras de limitar a probabilidade de sucesso de ataques DoS, mas não é possível eliminá-lo completamente. Por exemplo, alterando determinados parâmetros de rede, como o tamanho dos blocos que contêm transações. No entanto, o tamanho do bloco é inversamente proporcional à escalabilidade da Blockchain. Quanto maior, mais transações podem ser realizadas e menos taxas de transação são necessárias. Pelo contrário, quanto menor se torna, menor o número de transações que ele pode manter e as taxas de transação aumentam. Do ponto de vista de segurança, é melhor diminuir o tamanho para aumentar as taxas de transação, tornando os ataques de negação de serviço mais caros, pois os atacantes precisam pagar por cada transação que enviam à rede. Muitos protocolos estão alterando esses parâmetros de maneira experimental, pois eles próprios não sabem como defini-los de maneira ideal. Essencialmente, quando há uma rede que está em constante alteração, onde alguns nós estão entrando enquanto outros estão saindo, fica muito difícil definir esses parâmetros. Isso significa que precisa-se de alteração instantânea de parâmetros para satisfazer as necessidades da rede conforme ela muda. Em resumo, os ataques DoS são inevitáveis, mas com uma diversificação justa das conexões de entrada e saída, é possível reduzir a probabilidade de sucesso e inviabilizar esses ataques.

4.2 Ataque de Gasto Duplo

Uma das preocupações mais comuns para os sistemas de moedas digitais é a possibilidade do gasto duplo, quando um usuário malicioso gasta um mesmo valor em duas transações diferentes. Para ocorrer a tentativa de um gasto duplo é necessário que haja um fork, pois se o gasto ocorrer na mesma cadeia, quando o novo bloco for criado, ele não passará nas verificações iniciais de consistência e será descartado. Com o *fork*, o usuário malicioso faz um gasto e envia para rede, gasta a mesma quantia novamente em outro lugar e começa a minerar sobre esse gasto. Desta forma, há a possibilidade de conseguir minerar um bloco e realizar o fork. A partir deste momento, a rede estará dividida e haverá uma corrida que será vencida pela maior cadeia. Como uma das cadeias irá ser aceita pela rede e, a outra descartada, o gasto duplo será detectado. Uma das transações será descartada e o gasto duplo será rejeitado.

4.3 51%

Um ataque contra o mecanismo de consenso é o "ataque de 51%". Nesse cenário, um grupo de mineradores, controlando uma maioria (51%) [21] do poder de hash total da rede, conspira para atacar o mecanismo de consenso. Com a habilidade de minerar a maioria dos blocos, os mineradores atacantes podem gerar bifurcações deliberadas na Blockchain, gerar transações de gasto duplo ou executar ataques de negação de serviço (DoS) contra usuários ou transações específicas. Como já mencionado, um ataque de gasto duplo ou fork é um ataque onde o atacante faz com que blocos já confirmados sejam invalidados ao fazer uma bifurcação em um nível abaixo, com uma posterior re-convergência em uma cadeia alternativa. Com poder suficiente, um atacante pode invalidar seis ou mais blocos em uma sequência, invalidando transações que antes eram consideradas imutáveis (com seis confirmações). Note que o gasto duplo só pode ser feito nas transações do próprio atacante, para as quais o atacante pode produzir uma assinatura válida. Fazer um gasto duplo da própria transação é rentável quando, ao invalidar uma transação, o atacante puder receber um pagamento irreversível ou um produto sem ter que pagar por isso.

É usualmente aceito na rede Bitcoin que uma transação é considerada confirmada quando existem seis novos blocos com altura maior que a sua, pois será necessário muito esforço para alterá-la. A segurança das Blockchains depende do consenso distribuído alcançado pela prova de trabalho. Assume-se, que não há conluio de mineradores, ou seja, nenhum grupo coordenado de mineradores (ou um único minerador) pode possuir

mais de 50% da capacidade computacional da rede.

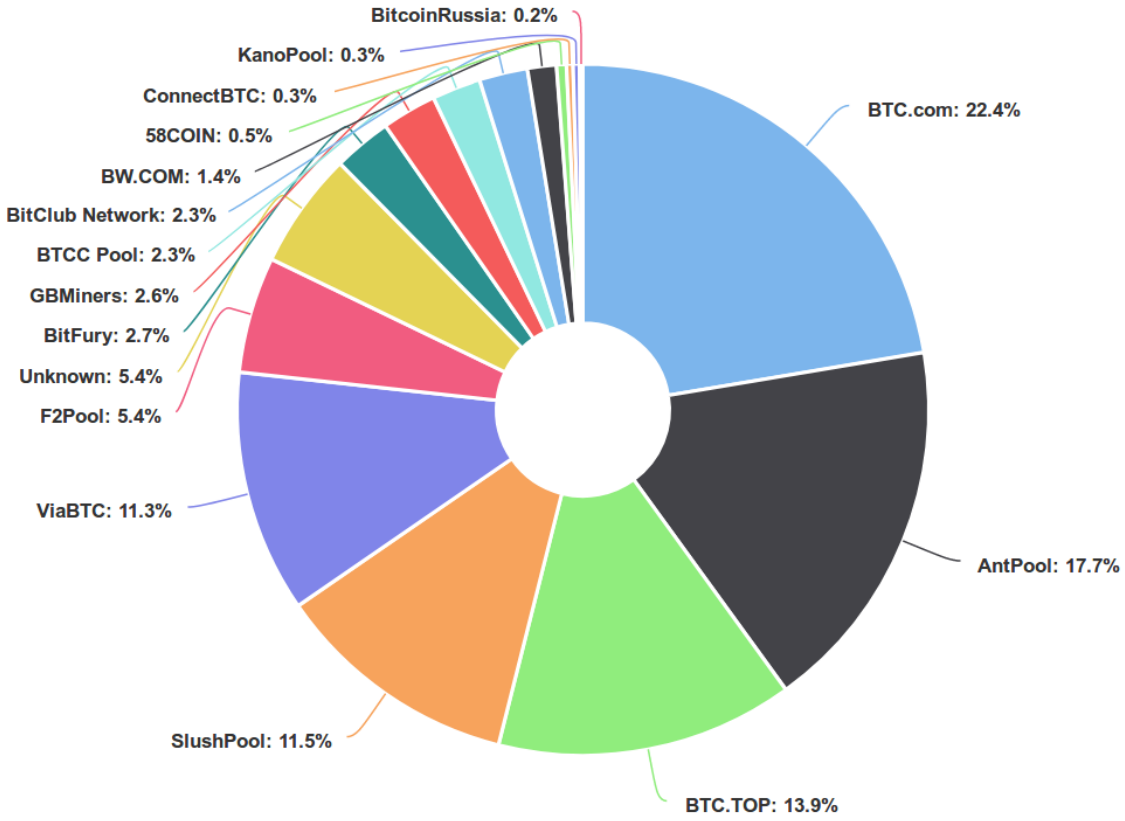

Figura 4.1: Cooperativas de mineradores [7].

No entanto, esta suposição é questionável. Primeiramente, é observado que há algum tempo que a mineração passou a ser organizada em cooperativas de mineradores (*mining pool*), que juntam esforços e compartilham as recompensas. Na Figura 4.1 é possível observar as maiores cooperativas em atividade na rede Bitcoin, onde apenas as três maiores cooperativas possuem 54% do poder de hash da rede. Em segundo lugar, não existe nenhuma entidade reguladora e nenhum minerador é obrigado a seguir o protocolo. Portanto, uma cooperativa, que possua maioria de poder de computação, pode alterar as regras que são aplicadas pelo mecanismo de consenso. Assim, os mineradores que não participam da cooperativa provavelmente verão seus ganhos diminuírem e serão compelidos a se juntar a ela. Por exemplo, uma cooperativa, com mais de 50% do poder computacional da rede, pode escolher aceitar blocos vindos de outros mineradores na razão de 1:2, ou seja, a cada dois blocos enviados pelos nós honestos apenas um será aceito. Isso é possível pois a cooperativa possuirá o poder de manipular o consenso. Os mineradores honestos terão seus blocos ignorados e, portanto, perderão os pagamentos.

4.4 Ataques *Sybil*

Um dos maiores problemas ao se conectar a uma rede peer-to-peer é o ataque *Sybil*. Foi descrito pela primeira vez por um pesquisador da *Microsoft*, John Douceur [20]. Um ataque de *Sybil* é aquele em que o atacante finge ser muitas pessoas ao mesmo tempo. Baseia-se no fato de que as redes peer-to-peer não conseguem distinguir de forma confiável entre membros em alguns serviços de internet que usam NAT (*Network Address Translation*). Considerando um nó mal-intencionado que tenha conseguido criar muitas identidades na rede. Dentro da rede, esse nó malicioso parece ser um grande grupo de nós representando uma grande porcentagem da rede. Os outros nós honestos podem não ser capazes de detectar esse comportamento e podem aceitar informações compartilhadas a partir desse nó mal-intencionado, achando que os dados estão chegando de muitas fontes diferentes (ou seja, a aleatoriedade é imposta). Isso implica que os ataques da Sybil visam a rede como um todo, não um nó em particular.

Tais ataques são difíceis de detectar, mas infelizmente ocorrem. Por exemplo, a empresa suíça Chainalysis, que fornece análises de Blockchain, criou mais de 250 nós falsos de Bitcoin e estava tentando coletar informações sobre transações que se propagam pela rede. Logo, depois de ser acusado pelos desenvolvedores do Bitcoin, o CEO da empresa admitiu que a empresa realmente criou esses nós, mas apenas para orientar o seu propósito.

Embora haja algumas precauções que possam ser tomadas para diminuir as chances de tais ataques, não há como eliminá-los completamente. Por exemplo, decidir nunca se conectar a nenhum nó da rede, exceto aqueles em que se confia. Mas realmente é possível confiar em qualquer nó? E se esta for a primeira vez que um nó se conecta e não se sabe de nenhum nó em que se possa confiar. Algumas soluções para este problema sugerem a utilização de sistemas baseados em identidade ou baseados em reputação que podem dar uma indicação de quão honesto é um nó. Isso é o suficiente? E quanto a privacidade? Se confiar na divulgação de identidades para poder realizar uma conecção à rede, qual é o propósito por trás do anonimato em tais sistemas?

Vale ressaltar que outras contramedidas como Prova de Trabalho (PoW) têm sido utilizadas por muitas criptomoedas para proteger contra ataques de Sybil durante a mineração. PoW requer que cada nó que deseja participar do processo de mineração realize uma competição em um enigma de criptografia bem custoso. Agora, a criação de múltiplas identidades ainda é possível, mas fornecer o poder computacional para resolver esse quebra-cabeça torna-se um empecilho. Isso não impede completamente que ataques de

Sybil ocorram, mas com certeza diminui suas chances.

Bissias [5] propõe o *XIM*, uma heurística que detecta o *Sybil* e cobra taxas extras, tornando o ataque custoso.

4.5 Eclipse

O Ataque *Eclipse* [27] é um ataque a nível de rede. Ocorre quando um atacante monopoliza todas as conexões de um determinado nó, isolando a vítima e filtrando todas as mensagens enviadas e recebidas. Como resultado, a vítima tem uma visão diferente da cadeia, pode ter seus blocos impedidos de serem incluídos na cadeia e pode ser forçada a trabalhar na cadeia do atacante. Em contraste com os ataques Sybil, os ataques do Eclipse são principalmente focados em atacar nós únicos e não toda a rede de uma só vez.

Na rede Bitcoin, os nós mantém no máximo 125 conexões com seus vizinhos, sendo 8 conexões de saída, e 117 de entrada. As conexões de saída são aquelas iniciadas pelo próprio nó, e as de entrada são as que outros nós solicitam. Para armazenar o endereço destas conexões os nós usam duas tabelas, uma tabela de conexões bem sucedidas, onde são armazenadas as informações de todas as conexões estabelecidas, de entrada e de saída, e uma tabela de endereços fornecidos, por solicitação ou não. A primeira é chamada de *Tried Table* e a segunda de *New Table*.

- ***Tried Table:*** É formada por 64 recipientes que podem armazenar 64 endereços cada. Os recipientes são escolhidos da seguinte forma: Quando o nó é iniciado ele escolhe um valor aleatório SK, e calcula:

  $$Recipiente = Hash(SK, Grupo, Hash(SK, IP)\%4)\%64,$$

  onde o Grupo é o prefixo /16 do endereço IP. Quando um nó estabelece uma conexão ele então mapeia o IP do novo vizinho para um recipiente. Caso o recipiente esteja cheio, o nó então chama uma função para remover endereços do recipiente. Quatro endereços são escolhidos aleatoriamente e o mais antigo é então movido para a *New Table*.

- ***New Table:*** É formada por 256 recipientes e cada um armazena até 64 endereços. É preenchida pelos endereços removidos da *Tried Table* ou por endereços fornecidos pelos *DNS seeders* ou por mensagens *ADDR*, que são mensagens para informar novos endereços aos vizinhos. De forma similar ao item anterior, aqui também é

executada uma função para mapear o recipiente e existe uma função de remoção de endereços antigos.

Quando um nó precisar estabelecer uma nova conexão, ele irá escolher um endereço de uma das duas tabelas: *Tried ou New*. Para isso, ele usa a seguinte fórmula, que dá a probabilidade de escolha da *Tried*:

$$P_{tried} = \frac{\sqrt{\theta(9-\epsilon)}}{(\epsilon+1)+\sqrt{\theta(9-\epsilon)}}$$

Onde θ é a razão entre a quantidade de endereços armazenados na *Tried* sobre a *New* e ϵ é quantidade conexões iniciadas.

O ataque consiste em encher a *Tried Table* com endereços controlados pelo atacante e encher a *New Table* com endereços inválidos. Desta forma sempre que a vítima buscar uma nova conexão ela se conectará a um endereço controlado pelo atacante. A *New Table* é preenchida com endereços inválidos para que o atacante economize IPs. Assim, basta realizar duas rotinas repetidas vezes para popular as tabelas da vítima. Primeiro, para estabelecer conexões de entrada, basta que o atacante solicite uma conexão. Em seguida, desconecta e solicita uma nova conexão com outro endereço. Isto basta para preencher a *Tried Table*. Para o atacante preencher a *New Table*, é necessaŕio enviar diversas mensagens *ADDR* com endereços inválidos para a vítima. Cada *ADDR* pode conter até 1000 endereços. São usados endereços classe C ou reservados para uso futuro, como a faixa destinada ao multicast, pois o atacante precisa de no mínimo 16384 endereços para preencher a *New Table*. Em seus experimentos, com uma *botnet* com 400 zumbis e foi capaz de preencher totalmente a *New Table* e 60% da *Tried Table*, alcançando sucesso em controlar todas as conexões da vítima em 80% das tentativas de ataque.

O maior medo dos ataques do Eclipse são os ataques que podem vir depois. Por exemplo, um nó atacante poderia facilmente executar um ataque de gasto duplo em tal configuração. Isso pode ser feito facilmente, enviando ao nó da vítima uma transação mostrando o comprovante de pagamento, eclipsando-o da rede e finalmente enviando outra transação para toda a rede, gastando os mesmos tokens novamente. Dado o fato de que o nó da vítima é isolado e recebe apenas dados dos nós maliciosos, ele continuará acreditando em um estado incorreto da Blockchain. A Figura 4.2 mostra um diagrama representando a rede em um ataque do Eclipse. Os nós maliciosos no eclipse vermelho (isolam) um dos nós honestos em azul de toda a rede, monopolizando suas conexões.

Houve várias contramedidas discutidas que podem ajudar a reduzir esses ataques. A

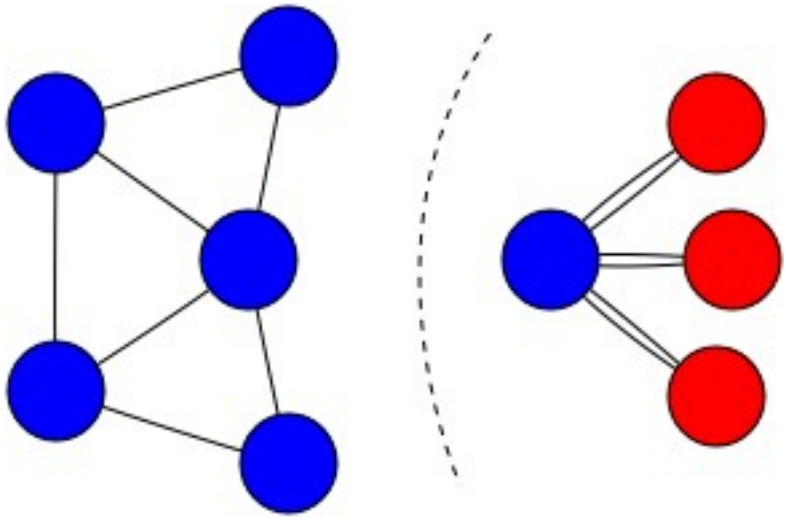

Figura 4.2: Um diagrama representando a rede em um ataque do Eclipse. Os nós maliciosos em "vermelho" no eclipse (isolam) um dos nós honestos em "azul" de toda a rede, monopolizando suas conexões

maioria deles envolve como um nó localmente armazena os endereços IP que serão usados para se reconectar posteriormente à rede. No entanto, semelhante aos ataques anteriores, não há como eliminá-los completamente.

4.6 Ataques de roteamento

Ataques de roteamento interceptam as mensagens propagadas pela rede, adulterando-as antes de empurrá-las para seus colegas. A única maneira de os nós detectarem tal adulteração é quando recebem uma cópia diferente de outro nó. Mas e se eles não tiverem outra fonte de recepção de dados propagados pela rede? E se o nó malicioso for capaz de dividir a rede de modo que ela seja dividida em duas ou mais partições que não podem se comunicar mais ou se ver .

Os ataques de roteamento são divididos em dois ataques menores separados.

- ***Ataque de particionamento:*** o invasor tenta dividir a rede em dois ou mais grupos separados. Isso pode ser feito seqüestrando certos pontos dentro da rede que

atuam como o ponto de ligação entre dois grupos.

- ***Ataque de atraso:*** Em seguida, o atacante pega as mensagens de propagação, adultera-as e, finalmente, as empurra para o lado da rede que não a viu antes.

A Figura 4.3 mostra um diagrama representando a rede em um tipo de ataque de roteamento.

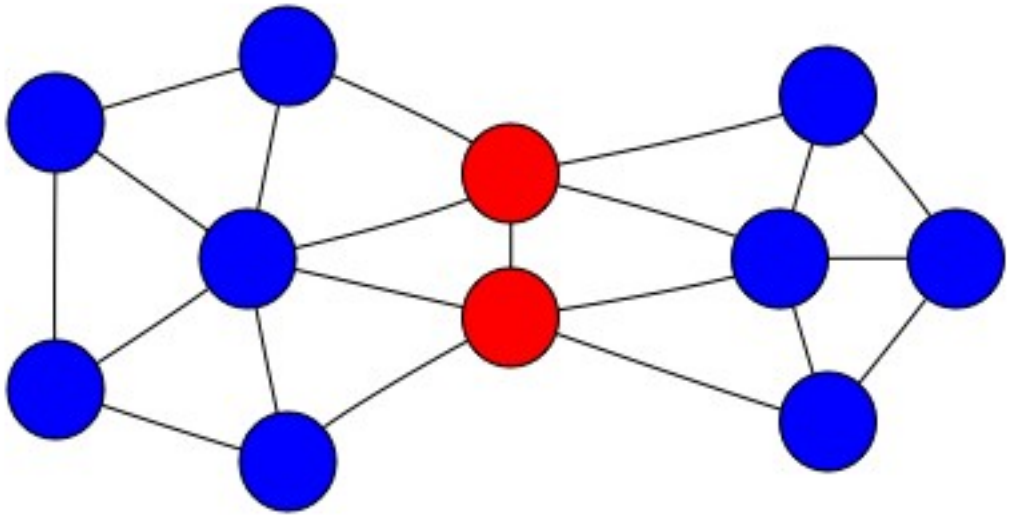

Figura 4.3: Um diagrama representando um exemplo de um ataque de particionamento. A rede se divide em dois grupos separados. Nós maliciosos em "vermelho" que isolam os nós honestos em "azul"

Não é possível evitar que esses ataques aconteçam, no entanto, existem métodos válidos para limitá-los. Por exemplo, ao diversificar continuamente as conexões de rede, dificulta-se muito a tarefa de um invasor de encontrar pontos para seqüestrar e dividir a rede em dois ou mais grupos separados. Outro método é monitorar os parâmetros de rede como o RTT *(Round-Trip Time)* e reconhecer padrões irregulares. O RTT é basicamente o tempo necessário para que os nós compartilhem dados e confirmem sua entrega. Uma vez detectado, os nós podem simplesmente se desconectar e tentar se conectar a outros nós aleatórios.

4.7 Mineração Egoísta *(Sefish Mining)*

Os mineradores maliciosos podem desviar seu comportamento do padrão ao não divulgarem imediatamente os seus blocos recém minerados. Estes ataques são chamados na literatura de *Selfish Mining* [21, 45]. Primeiramente é preciso entender como a latência na propagação dos blocos e o tempo alvo para inclusão de novos blocos afetam o mecanismo de consenso.

- **Latência de propagação dos blocos:** A propagação de mensagens na rede segue o protocolo P2P próprio. Um nó, ao receber um novo bloco, o retransmitirá aos seus vizinhos. Antes de iniciar a transmissão, ele faz uma série de verificações a fim de garantir que somente blocos válidos sejam propagados. Cada nó que recebe um novo bloco faz essas verificações independentemente. Após isso, o nó envia uma mensagem de inventário (INV), informando aos seus vizinhos que possui um novo bloco e sua altura. Caso estes vizinhos ainda não tenham recebido este bloco por outros nós eles enviam uma mensagem solicitando o novo bloco(GET_DATA). Somente então será iniciada a transmissão efetiva do novo bloco, Fig. 4.4. A soma de todos esses tempos, verificação e propagação da mensagens, durante a propagação de um bloco, é chamada de *latência de propagação dos blocos*. Decker [17] fez a análise do tempo de propagação de 10.000 blocos, com diferentes tamanhos, e verificou que a mediana desde o anúncio de um novo bloco criado até o recebimento por um determinado nó foi de 6,5 segundos e a média de 12,6 segundos. Outra observação interessante foi que após 40 segundos, 5% dos nós ainda não haviam recebido o novo bloco.

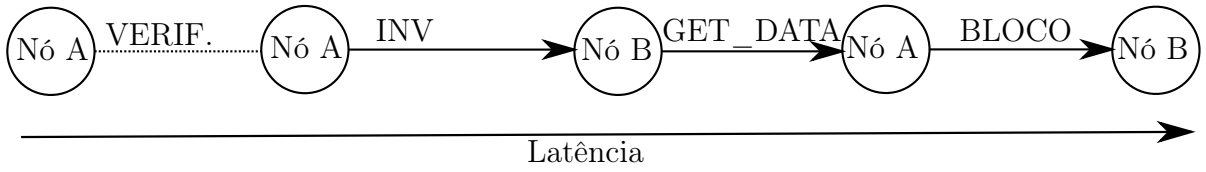

Figura 4.4: Latência de propagação dos blocos

O intervalo para a inclusão de novos blocos é crucial para a quantidade de *forks* observados na rede. Quanto menor for esse intervalo, maior será a quantidade de blocos gerados e consequentemente maior será a probabilidade de ocorrência de forks.

Um atacante pode desviar do comportamento padrão e manter secretamente uma cadeia onde somente ele produza blocos, e adote uma heurística própria para divulgação

destes blocos. Neste momento, todos, inclusive o atacante estariam minerando sobre o bloco n. Ao realizar esta ação, caso ele consiga produzir o próximo bloco, o atacante conseguirá uma vantagem em relação aos demais mineradores, mesmo sem possuir maior poder computacional, pois ele poderá iniciar o processo de mineração do bloco $n+1$ antes de todos os outros mineradores. Se o atacante receber um bloco da rede ele pode decidir adotar este bloco e jogar fora seu trabalho, ou ignorar o bloco recebido e continuar minerando na cadeia privada. Como ele começou a minerar antes, há uma probabilidade de minerar o bloco $n+1$ antes dos outros, gerando um fork com altura maior que a cadeia principal. Como os demais nós se comportam honestamente, eles também irão adotar esta cadeia e o atacante atingirá seu objetivo. Estes ataques são chamados na literatura de *Selfish Mining* [21, 45]. Em [21], Eyal faz uma análise matemática e propõe um modelo de transição de estados para capturar o melhor momento do atacante liberar seus blocos privados. Ele analisa as probabilidades de ocorrência de cada estado e chega a conclusão que para o atacante conseguir êxito, ou seja, publicar mais blocos que a cadeia honesta, seu poder de mineração deve satisfazer a seguinte inequação:

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2} \tag{4.1}$$

onde α é o poder de mineração do atacante e γ é a razão dos nós honestos que escolhem minerar a cadeia desonesta. Isso é uma vantagem ao atacante, pois ele necessitará de menor poder para conseguir suplantar os nós honestos. Assim, a partir desta inequação é obtido o menor valor de α para um determinado γ. O pior caso para o atacante é quando nenhum nó honesto adota a sua cadeia, neste caso é necessário que o atacante possua um terço do poder computacional da rede.

Nayak [45] amplia a pesquisa de Eyal e verifica que o ataque proposto anteriormente não é ótimo. Ele propõe novas estratégias para aumentar o ganho do atacante, levando em conta não apenas o tamanho das cadeias, mas também seu poder computacional. Por exemplo, mesmo que o atacante esteja perdendo a corrida, caso ele possua uma parcela significativa de poder computacional, é melhor que ele continue a minerar em sua cadeia privada, pois há uma probabilidade de alcançar e ultrapassar a cadeia honesta. Outra contribuição de seu trabalho é mostrar que se o *Selfish Mining* for combinado com o *Eclipse* [27], quando o atacante controla todas as conexões com um determinado nó, o atacante aumentará seus ganhos e surpreendentemente, com determinados parâmetros, o nó que foi alvo do *Eclipse* também conseguirá publicar mais blocos, em relação aos nós honestos.

O autor modela a mineração como um processo de decisão de Markov, que usa como

informação as transições de estado, quando os mineradores acham um novo bloco. Quando um nó honesto produz um novo bloco ele o publica imediatamente, enquanto o atacante mantém seus blocos ocultos. Depois, ele verifica as alturas das cadeias e se os nós honestos estão minerando sobre a cadeia honesta ou desonesta. Suas estratégias são chamadas de *Stubborn Mining*, são elas:

- ***Lead Stubborn:*** Vimos anteriormente que γ é um fator importante, pois quanto mais nós honestos estão minerando na cadeia do atacante menor será o α necessário. Vimos também que a latência de propagação dos blocos influencia fortemente quais blocos cada nó recebe primeiro. Assim, uma das estratégias adotadas pelo autor define que, caso o atacante esteja liderando por um bloco, tão logo os nós honestos liberem um bloco, o atacante também libera o seu. O objetivo é que seu bloco alcance uma parcela dos nós honestos que irão adotá-lo como referência para mineração, aumentando o γ e a probabilidade do atacante vencer a corrida. Caso esteja vencendo por dois ou mais ele mantém sua cadeia privada, somente revelando os blocos quando a diferença alcançar 1.

- ***Trail Stubborn:*** Quando a cadeia privada do atacante é menor que a pública. O atacante continua minerando sob a cadeia privada ao invés de abandoná-la, na esperança de alcançar e ultrapassar a cadeia pública. Esta estratégia se mostra promissora caso o atacante possua grande poder computacional.

- ***Equal Fork Stubborn:*** Esta estratégia é usada quando os nós honestos igualam a corrida e o atacante continua minerando em sua cadeia até que fique um bloco a frente, quando então ele libera sua cadeia.

Como vimos acima, uma parcela da latência de propagação dos blocos é gerada pela obrigatoriedade de verificação dos novos blocos por todos os nós. Caso um atacante controle alguns nós da rede, como em uma cooperativa, ele pode tirar vantagem deste fato para amplificar o ataque do minerador egoísta. Estes nós controlados pelo atacante podem ser configurados para não realizar a verificação dos blocos minerados pelo atacante e retransmiti-los tão logo eles cheguem. Assim os blocos do atacante alcançam os outros nós em menor tempo que os nós honestos. Isso pode ser uma vantagem. Outra observação a respeito deste ataque é que era de se esperar que o total de blocos adicionados à cadeia seja a soma dos blocos produzidos pelo atacante e pelos nós honestos. Mas a ocorrência de forks gera blocos que serão descartados, *stale blocks*. Assim, a quantidade total de blocos incluídos é menor que o total de blocos produzidos. Isso é importante pois se for

desconsiderada a entrada de novos mineradores na rede quando os nós forem recalcular a nova dificuldade, esta será menor que a dificuldade anterior.

4.8 Ataque do Perseguidor *(Stalker Atack)*

O atacante perseguidor *(Stalker Attack)* [32] é uma variação da mineração egoísta com um objetivo diferente. Enquanto o *Selfish miner* tem por objetivo de auferir maiores ganhos em relação aos nós honestos, o *Stalker* tem o objetivo de impedir que um determinado nó publique um bloco, ou que uma determinada transação seja incluída na cadeia.

A corrida em busca de um novo bloco é modelada pelo passeio aleatório binomial [51]. O evento de sucesso ocorre quando os nós honestos encontram um bloco, o evento de fracasso ocorre quando nós maliciosos encontram o bloco, e a probabilidade do fork malicioso alcançar a desvantagem de n blocos é análoga ao problema da ruína do jogador, podendo ser calculada por [6]:

$$a_n = \begin{cases} 1, & \text{se} \quad h \leq a, \\ (\frac{a}{h})^n, & \text{se} \quad h > a. \end{cases} \tag{4.2}$$

Onde: h é a probabilidade do minerador honesto encontrar o próximo bloco, a é a probabilidade do minerador malicioso encontrar o próximo bloco; e a_n é a probabilidade do minerador malicioso compensar a desvantagem inicial de n blocos.

A força do minerador malicioso, ou seja, quantos blocos ele consegue minerar durante o período de mineração de n blocos honestos, pode ser modelado como uma distribuição de Poisson de valor médio α calculado por:

$$\alpha = n\frac{a}{h} \tag{4.3}$$

A partir das Equações 4.2 e 4.3 é possível calcular a probabilidade de um atacante suplantar os n blocos minerados pelos nós honestos, dado que ele minere k blocos, por meio da equação:

$$P_n = 1 - \sum_{k=0}^{n} \frac{\alpha^k e^{-\alpha}}{k!} \left[1 - \left(\frac{a}{h}\right)^{(n-k)}\right] \tag{4.4}$$

A partir da Equação 4.2 pode-se concluir que se os nós maliciosos possuírem a maioria do poder computacional da rede a probabilidade de sucesso é de 100%. E, existe uma probabilidade não nula de que mesmo sem possuir maior poder computacional ele alcance a cadeia honesta.

A fim de tirar proveito desta probabilidade não nula, alguns autores criaram heurísticas [23, 52, 45, 21] que possibilitam auferir maiores ganhos relativos em relação aos nós honestos. A Equação 4.1 mostra que adotando o comportamento egoísta os atacantes já começam a ter maiores ganhos relativos a partir de 33% de poder computacional.

Estas heurísticas funcionam de maneira desonesta. Elas fogem ao protocolo, pois os atacantes não divulgam imediatamente seus blocos. Realizam *forks* deliberadamente e com isso desperdiçam energia e tempo dos mineradores honestos.

O *Stalker* também adota esse comportamento, mas ele busca um alvo pré-definido. podendo ser uma transação ou um nó. O *Stalker* permanece constantemente minerando em uma cadeia que somente ele conhece. Quando o nó alvo publica um bloco existe uma probabilidade de a cadeia do Stalker ser maior que a cadeia honesta. Caso sua cadeia seja maior ele a publica. Desta forma o bloco enviado pelo alvo será descartado por todos os nós honestos, que adotaram a cadeia maliciosa.

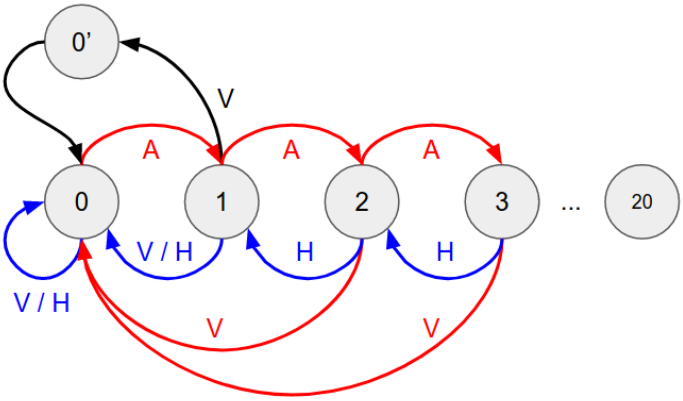

Figura 4.5: Transição de estados

Para definir o melhor momento de iniciar uma cadeia privada, ou publicar seus blocos, foi utilizada estratégia descrita por [52], que usa um processo de decisão de Markov para modelar os principais estados em que um atacante egoísta pode estar, calcula as probabilidades de transição entre os estados e o melhor momento de liberar seus blocos.

Os estados em que o atacante pode estar variam em função da altura relativa da cadeia privada em relação a cadeia honesta, e são apresentados na Figura 4.5. Os números

dentro dos círculos correspondem a altura relativa da cadeia privada em relação a honesta. Quando o atacante está no estado 0 e a vítima ou um dos outros nós honestos publicam um bloco ele permanece no estado 0 e reinicia a mineração. Quando o atacante encontra um bloco antes dos demais nós ele aumenta sua cadeia privada e vai ao estado 1. O atacante vai mudando então de estado sempre que minera ou recebe um bloco dos demais nós da rede. Caso ele esteja em um estado que possua altura maior ou igual a dois e receba um bloco da vítima ele publica sua cadeia imediatamente. Caso ele esteja no estado 1 e receba um bloco da vítima ele pode decidir adotar o bloco da cadeia honesta e reiniciar o ataque, ou publicar a sua cadeia, tirando proveito do efeito da latência de propagação dos blocos. Neste caso ele continua minerando sobre seu bloco e espera que alguns nós honestos receberão o seu bloco primeiro que o nó da vítima, aumentando assim as suas chances de impedir a publicação deste bloco.

Tabela 4.1: Matriz de Decisão

la \ lh	1	2	3	4	5	6	7	8	9	10	11
1	“w”, “w”	-, “a”	-	-	-	-	-	-	-	-	-
2	“w”, “w”	“w”, “w”	-, “a”	-	-	-	-	-	-	-	-
3	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-	-	-	-	-	-
4	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-	-	-	-	-
5	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-	-	-	-
6	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-	-	-
7	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-	-
8	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	-, “a”	-	-
9	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “w”	“w”, “a”	-, “a”	-
10	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“a”, “w”	“w”, “w”	-, “a”
11	-	-	-	-	-	-	-	-	-	“a”, “a”	“w”,“w”

Existem duas cadeias concorrentes. Uma pública vista por todos os nós, e uma privada que apenas o atacante tem conhecimento. O atacante mantém as duas cadeias. A cada novo bloco, recebido ou minerado, o atacante verifica qual cadeia é maior, e usa uma matriz de decisão para verificar que ação tomar. Estas decisões são as mesmas das heurísticas propostas para o minerador egoísta, e podem ser observadas na Tabela 4.1. As linhas da tabela correspondem à altura relativa da cadeia do atacante, e as colunas à cadeia dos nós honestos, “a” corresponde a ação adotar, o “w” a ação esperar, e o “i” a ação igualar. O primeiro valor de cada célula é usado quando o bloco foi descoberto por um nó honesto, e o segundo valor é usado quando o bloco for minerado pelo atacante.

A partir do momento que o atacante inicia o ataque ele atribui valor zero a altura das duas cadeias. Quando recebe ou minera o próximo bloco o atacante altera também a altura da cadeia correspondente, consulta a matriz e toma a ação decorrente. Sempre que o atacante recebe um bloco do alvo e a cadeia privada é maior que a honesta ele publica sua cadeia. O Algoritmo 1 demonstra o comportamento do atacante. onde *la* corresponde

Algoritmo 1: Stalker Attack

```
1  if Próximo Bloco == vítima then
2  |   if la > lh then
3  |   |   Publicar a cadeia do atacante;
4  |   |   return;
5  |   end
6  end
7  switch Ler_Matriz do
8  |   case Esperar do
9  |   |   Continuar minerando na cadeia do atacante;
10 |   end
11 |   case Adotar do
12 |   |   Reiniciar o ataque;
13 |   end
14 |   case Igualar do
15 |   |   Publicar a cadeia do atacante;
16 |   end
17 end
```

a altura da cadeia do atacante e lh a altura da cadeia honesta. As ações são descritas a seguir:

- **Publicar:** Corresponde a revelar sua cadeia e ocorre quando o atacante estava no estado esperar e recebe um bloco do alvo (Algoritmo 1 linha 3). No exemplo da Figura 4.6 o minerador estava minerando secretamente em uma cadeia com altura 5 e a cadeia honesta possuía altura 3. Quando a vítima envia o bloco 4 o atacante verifica que sua cadeia é maior que a honesta e que há um bloco da vítima nela. Decide revelar sua cadeia privada. Todos os nós da rede ao receberem esta nova cadeia irão verificar que ela é maior que a que eles possuem, descartando assim estes blocos e adotando a cadeia maliciosa.

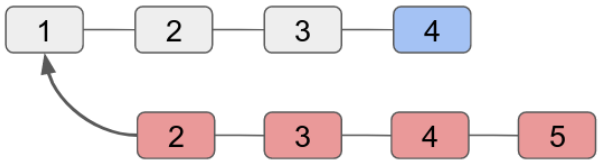

Figura 4.6: Ação: publicar

- **Esperar:** O atacante minera constantemente em sua cadeia, sem a revelar. Ocorre quando a cadeia do atacante é maior que a honesta, mas o atacante não recebeu um bloco do alvo (Algoritmo 1 linha 8). Na Figura 4.7 o atacante continua minerando

sob a sua cadeia privada a espera de um bloco da vítima. Caso sua cadeia fique muito maior que a cadeia honesta ele pode decidir abandoná-la e reiniciar o ataque.

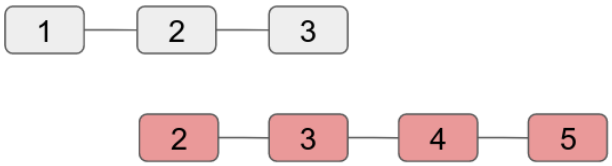

Figura 4.7: Ação: esperar

- **Adotar:** O Adversário adota a cadeia honesta. Isto corresponde a reiniciar o ataque, pois o atacante infere que a cadeia dos nós honestos possuem uma probabilidade maior de vencer a corrida (Algoritmo 1 linha 11). Na Figura 4.8 o atacante verifica que sua cadeia é menor que a honesta e decide abandoná-la.

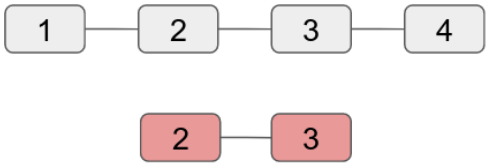

Figura 4.8: Ação: adotar

- **Igualar:** O atacante publica tantos blocos quantos possuir a cadeia honesta. desta forma ele desencadeia uma corrida pela adoção entre as duas cadeias tirando proveito da força de alguns nós honestos (Algoritmo 1 linha 14). Na Figura 4.9 a medida que os nós honestos publicaram o bloco 2, o atacante decide igualar as cadeias, liberando assim o seu bloco 2. Ocorre da mesma forma com o bloco 3. Alguns nós honestos receberão o bloco 3 do atacante primeiro que o bloco 3 dos honestos e o adotarão como referência. Um destes nós encontrou o bloco 4 e o publicou com referência ao bloco do atacante. O atacante alcançou o objetivo, que é impedir os blocos azuis, e reiniciou o ataque.

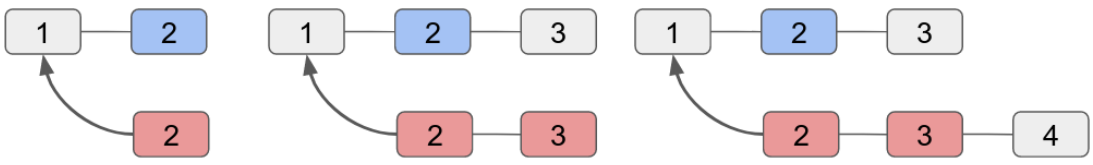

Figura 4.9: Ação: igualar

Para realizar ataques do tipo de mineração perseguidora em sistemas que utilizam Prova de Trabalho, o atacante deverá possuir enorme poder computacional e certamente

arcará com grandes gastos financeiros. Seus objetivos nem sempre são financeiros. Imagine que duas empresas decidam adquirir um grande ativo de uma terceira, e que haja um *Smart-Contract* com uma cláusula de prioridade para uma das empresas. Este contrato prescreve que a segunda empresa somente poderá adquirir o ativo caso a primeira não realize o pagamento em criptomoeda até uma data específica. A segunda empresa pode então realizar o ataque de modo a impedir que a transação realizada pela primeira empresa seja confirmada pela rede. Um atacante pode querer impedir que um sensor, que utilize Blockchain para vender seus dados, disponibilize seus dados. O dado pode ser estratégico para outras empresas que o utilizam, ou simplesmente o atacante pode querer acabar com a imagem da dona do sensor. Ali [2] anunciou o primeiro ataque, conhecido, de Minerador Egoísta a uma rede em produção, provando que o ataque é factível, apesar das motivações ou de ser muito dispendioso. Este sistema é uma aplicação de DNS e PKI que usa Blockchain. Foi notado uma grande quantidade de forks na rede e demora na validação de transações.

Outra consideração importante a ser feita se refere a identificação do usuário ou da transação. Ao realizar as transações, os nós não fornecem seus endereços IP e nem suas identidades, apenas as chaves criptográficas que provam a posse dos recursos. Por esta razão, é possível dizer que a Blockchain fornece um certo grau de privacidade aos usuários. Descobrir quais nós estão realizando transações específicas é muito difícil, mas não impossível. É possível inferir o endereço IP e suas identidades através de várias técnicas [56, 41, 28, 42, 38, 60]. Desta forma, parte-se do pressuposto que o usuário já foi identificado.

4.9 Outros Ataques

Os algoritmos de computação quântica, como o Shor [54, 55], em teoria, poderão quebrar o sistema de curvas elípticas e os algoritmos de hash. Consequentemente, será possível calcular chaves privadas a partir das chaves públicas e realizar a Prova de Trabalho rapidamente. Aggarwal [1] discorre que a Prova de Trabalho usada pela Bitcoin pode ser resistente aos computadores quânticos pelos próximos 10 anos. Então, o desenvolvimento de computadores quânticos representa uma séria ameaça para quase toda a criptografia e, portanto, para Blockchain. Esses algoritmos podem ser usados de duas maneiras para atacar a Blockchain: Podem ser usadas para realizar ataques de 51 % e acelerar a geração de *nonce*, consequentemente, recriar uma nova cadeia.

Capítulo 5

Heurística para a Detecção dos Ataques de Mineração Egoísta

Neste capítulo serão descritos os componentes da heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain, especificamente para os ataques de Mineração Egoísta e de Mineração Perseguidora, seu pseudocódigo, bem como resultados dos experimentos da sua aplicação em um conjunto de simulações e um modelo de avaliação de sua precisão com base na apresentação da taxa de Verdadeiros Positivos e Falsos Positivos.

5.1 Modelo Heurístico

Uma heurística pode ser vista como um modelo, processo ou método composto por uma ou mais regras que visam encontrar a solução para um determinado problema com base em informações observadas. Estudamos o comportamento da rede Blockchain baseada em PoW na presença de ataques de *Selfish Miner* e *Stalker*, e identificamos uma heurística simples e de fácil implementação para sinalizar a ocorrência desses ataques na rede.

Como comparação, destacamos os trabalhos de Decker [17] e Gervais [23]. O primeiro deles observou 10.000 blocos que foram propagados na rede entre a altura 180.000 e a altura 190.000 na cadeia Bitcoin, com a ocorrência natural de 169 forks, ou 1,69 % de forks. O segundo autor, nos resultados de simulação, observou uma taxa de fork de 1,85 %, demonstrando que os resultados da simulação estão muito próximos dos resultados observados em uma aplicação em produção.

Através da página Blockchain.info, que oferece estatísticas da rede real Bitcoin[7],

acessamos a taxa de forks no período correspondente a 10.000 blocos, que foram propagados na rede entre a altura 300.000 e a altura 310.000 na cadeia Bitcoin e verificamos a ocorrência de 67 forks ou 0,67 % de forks naturais na rede. Também fizemos o Download da rede Bitcoin, através do Bitcoin Core[6]. Através da interface RPC[1] aplicamos o comando *getchaintips* que retorna informações sobre a cadeia principal e os forks, conforme visto na Figura5.1. Observamos a ocorrência de 7 Forks no período correspondente a 1.000 blocos, ou seja uma taxa de 0,70%.


```
                {
                        "height": 472975,
                        "hash":
"000000000000000000067a24962827bc94d347066a2acec324625da67a0b10168",
                        "branchlen": 1,
                        "status": "valid-fork"
                },
```


Figura 5.1: Retorno do Comando *getchaintips* - Bitcoin RPC

Consultamos 1.000 blocos e obtivemos 8 Forks, ou seja uma taxa de 0,8% Alguns forks da Blockchain podem ser observados participando da rede e recebendo os dois blocos conflitantes. Porém, é difícil observar todos os forks de blockchain, pois se um nó detectar que um bloco entra em conflito com o bloco que esse nó acredita ser o topo da cadeia, ele não propagará mais o bloco. A maioria dos forks naturais apresenta o tamanho igual a 1. Como resultado direto, o relato fiel de todas as bifurcações de blockchain exigiria a conexão com todos os nós da rede. Devido ao fato de alguns nós não serem alcançáveis, por estarem protegidos por uma tradução de firewall ou de endereço de rede, apenas uma aproximação do número real de forks de Blockchain podem ser obtidas.

Analisando o comportamento de cada ataque, verificou-se uma alteração no padrão dos *forks* (bifurcações produzidas nas Redes Blockchain do Bitcoin baseadas em PoW) Esses *forks* podem aparecer por diversos motivos como visto na Subseção 3.3.1 e em função da presença de ataques como visto no Capítulo 4. Essa heurística identifica os *forks* de acordo com os seguintes parâmetros: frequência e a altura. Essas características e informações serão agrupadas e utilizadas para identificar um ataque na rede.

Observamos que um fork com altura maior ou igual a 2 já poderia caracterizar um ataque e com base nessa informação, alteramos o código do simulador para dois possíveis cenários: O primeiro cenário para detectar todos os forks de altura $\alpha \geq 1$ e apresentar como um possível ataque e outro para detectar qualquer fork de altura $\alpha \geq 2$ como possíveis ataques. Conforme os resultados, verificamos que o Detector para forks de

[1]O Bitcoin Core fornece uma interface RPC (chamada de procedimento remoto) para várias tarefas administrativas, operações de carteira e consultas sobre dados de rede e cadeia de blocos.

altura $\alpha \geq 2$ apresentou um menor número de Falsos Positivos, conforme esperado.

Posteriormente foi construído um avaliador para verificar a eficiência do Detector para forks de altura $\alpha \geq 2$. Uma análise dos forks de altura alpha 1 verificou quantos deles tratavam-se de efetivos ataques. Essas análises foram realizadas para ambos ataques, a fim de verificar diferenças na identificação. Os resultados são demonstrados na próxima seção.

5.2 Topologia e Simulação

Para realizar as simulações foi utilizado o simulador NS-3. O ns-3 é um simulador de redes, de código aberto, baseado em eventos discretos desenvolvido especialmente para pesquisa e uso educacional [11]. O script de simulação é escrito em C++, com suporte opcional a Python. Apesar de ser usado popularmente para simular redes, o NS-3 também pode ser usado para emulação. Weingartner [63] analisou os requisitos de desempenho e a escalabilidade de cinco ferramentas de simulação diferentes. Os resultados mostram que três deles, NS-3, OMNeT ++ e JiST, são capazes de realizar simulações de rede em larga escala de maneira eficiente. O JiST provou ser o simulador mais rápido em nossos experimentos, no entanto, o exaustivo consumo de memória pode limitar sua aplicabilidade em alguns cenários de simulação. Em comparação de desempenho, o NS-3 demonstrou o melhor desempenho geral.

A topologia foi composta por 16 nós de mineradores, entre eles 1 atacante, conforme ilustrado na Figura 5.2. A quantidade de nós mineradores foi escolhida para simular a quantidade de *Pools* que compoem a rede atual Bitcoin [7] conforme visto na Subseção 4.3, assim como a distribuição da porcentagem de *Hash Power*.

As simulações foram realizadas usando o NS-3 com o módulo de ataque de mineração egoísta criado por Gervais et al. [23] e o módulo do ataque de mineração perseguidora criado por Jesus et al. [32] e [33]. Pequenas alterações em seus códigos foram feitas para capturar o número de *forks* vistos pela rede e por um nó de mineração malicioso, bem como a altura desses *forks*,

A Tabela 5.2 condensa os parâmetros utilizados para realizar as simulações.

A simulação da Prova de Trabalho é obtida ao atribuirmos um valor de poder computacional a um determinado nó, os blocos são gerados obedecendo a uma função de distribuição geométrica, que define que nó gerou o bloco. Os nós honestos seguem o

Figura 5.2: Topologia

Parâmetro	Descrição
Tamanho do bloco	Traduz a quantidade de transações nos blocos, segue a distribuição real do tamanho do bloco da rede Bitcoin, conforme estimado coletando estatísticas do blockchain.info
Número de Blocos	O número de blocos gerados. Foram 2016 para cada iteração. Essa escolha refere-se ao número de blocos que a rede Blockchain leva para ajustar a dificuldade
Quantidade de Nós	Quantidade de nós completos e de nós mineradores na rede
Poder de Hash	Distribuição em porcentagens do poder computacional entre os mineradores
Distribuição Geográfica	Simula a latência de propagação dos blocos
Mínimo de Conexões	O mínimo de conexões que um nó faz com o restante da rede, conforme em [41]
Máximo de Conexões	O máximo de conexões que um nó faz com o restante da rede, conforme em [41]
Intervalo entre Blocos	Tempo médio para achar um bloco, conforme o protocolo Bitcoin de 10 minutos
Mecanismo de propagação	Simula o envio dos blocos por meio de cabeçalhos ou blocos completos

Tabela 5.1: Parâmetros de entrada na simulação

mesmo mecanismo de consenso do *Bitcoin*: sempre mineram usando como referência o bloco com a maior altura; e sempre que houver um fork ele será resolvido adotando a maior cadeia e descartando a menor.

A conexão entre os nós é feita por intermédio de canais ponto a ponto. Existem dois

tipos de nós: os mineradores e os nós completos. Pode-se atribuir aos nós mineradores o comportamento padrão, o comportamento Egoísta ou o Perseguidor.

A Tabela 5.2 elenca os principais resultados obtidos pelas simulações.

Resultado	**Descrição**
Altura do Bloco	Altura do bloco onde inicializou-se um *Fork*
Altura do Fork	Altura dos *Forks* visualizados na Blockchain
Diferença de Tempo	Diferença entre o tempo em que o bloco é colocado na cadeia e o tempo em que o bloco é criado por um minerador. Calculado no momento da resolução do *Fork*
Minerador	Minerador que solucionou o Fork.
Blocos descartados	Quantidade de blocos descartados (*stale blocks*) na Blockchain
Distribuição dos *Forks*	Quantidade dos *Forks* produzidos na Blockchain

Tabela 5.2: Valores de gerados pelo simulador

Durante a simulação da rede Bitcoin sem ataque para 10.000 blocos e 16 nós de mineração, a maior altura de *fork* observada foi de no máximo igual a 1. Conforme visto na Figura 5.3 a rede apresentou um total de 30 forks.

```
Total Stats:
Average Connections/node = 0
Average Connections/miner = 15
Mean Block Receive Time = 634.974 or 10min and 34.9739s
Mean Block Propagation Time = 0.711799s
Median Block Propagation Time = 0.676063s
10% percentile of Block Propagation Time = 0.526611s
25% percentile of Block Propagation Time = 0.606814s
75% percentile of Block Propagation Time = 0.918056s
90% percentile of Block Propagation Time = 0.982271s
Miners Mean Block Propagation Time = 0.711799s
Miners Median Block Propagation Time = 0.676063s
Mean Block Size = 535319 Bytes
Total Blocks = 9448
Stale Blocks = 15 (0.158764%)
The size of the longest fork was 1 blocks
There were in total 30 blocks in forks
```

Figura 5.3: Resultados da Simulação da Rede Real Bitcoin

Realizamos 100 simulações para cada conjunto de diferentes poderes computacionais (*Hash Power*) do atacante e para um possível nó alvo. Cada ataque teve uma duração de 2016 blocos gerados, essa quantidade de blocos foi escolhida pois a cada 2016 blocos, todos os nós reajustam o alvo da dificuldade da prova de trabalho. Conforme visto na Seção

3.1, a equação para reajustar o alvo mede o tempo que levou para encontrar os últimos 2016 blocos e o compara com o tempo esperado de 20160 minutos, aproximadamente duas semanas de dados da rede Bitcoin em produção, baseando-se em um tempo desejado de 10 minutos para gerar um bloco, conforme as regras em vigor da PoW da Blockchain.

Para cada simulação de ataque o poder computacional (*Hash Power*) do alvo foi variado de 5 a 30 % do *Hash Power* total e o *Hash Power* do atacante variou de 20 a 40 % do total de *hash*.

No *Selfish Miner*, onde o objetivo é obter a maior cadeia e receber a recompensa financeira, todos os nós são possíveis alvos, A Figura 5.4 mostra os *forks* alcançando suas alturas mais altas com o Poder de *hash* do atacante acima de 30 %, onde a maior altura de um *fork* variou entre 13 e 19 blocos. O poder de *hash* de um dos nós alvo foi variado para ver como isso influenciaria o comportamento do atacante na presença de um nó alvo com alto *Hash Power*. Verificou-se uma variação muito baixa na altura máxima obtida pelos *forks*, uma das razões é que esse alto poder de *hash* de um possível nó alvo só será vantajoso quando um bloco desse possível alvo estiver de fato na cadeia atacada.

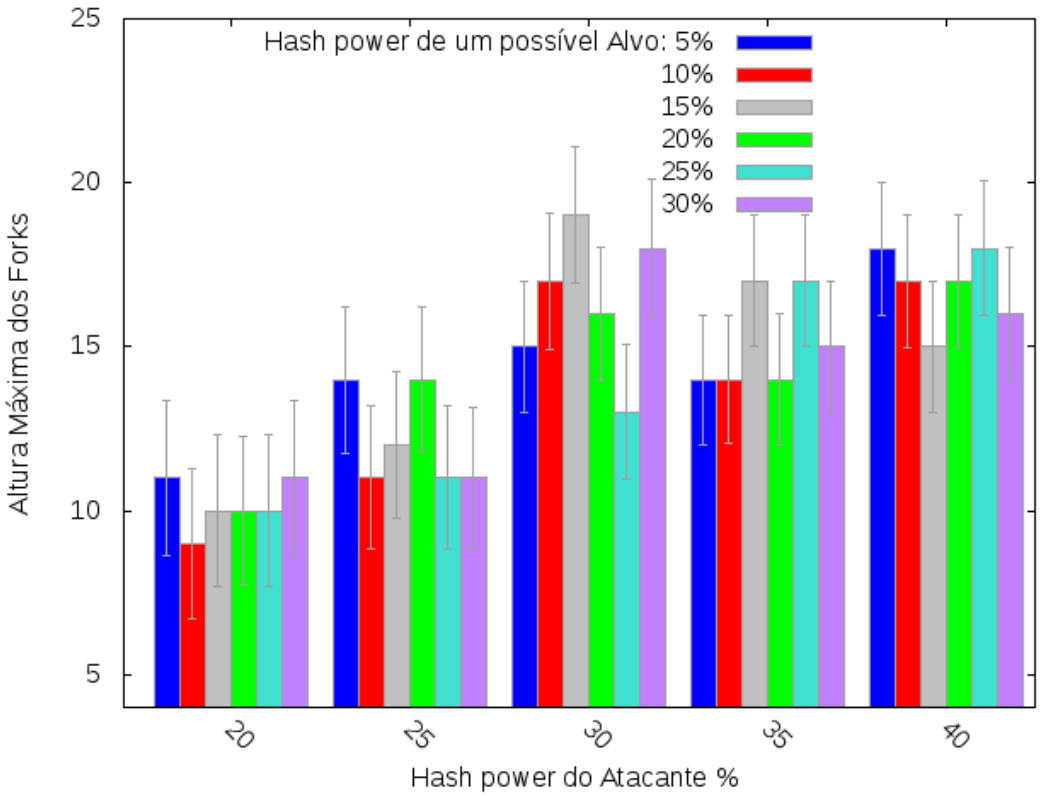

Figura 5.4: Altura Máxima dos Forks - Selfish Miner

No *Stalker Miner*, devido ser um ataque onde existe um alvo específico, as alturas máximas dos *forks* variam mais e alcançam valores mais altos, contanto que o atacante

descubra que há blocos do nó alvo na cadeia, o atacante insiste no ataque, tentando bloquear esses blocos. As alturas mais altas de *forks* são obtidas em simulações em que o *Hash Power* do atacante é maior que 30 % do valor total, a maior altura sendo obtida onde o *Hash Power* Alvo vs. o *Hash Power* Atacante é o valor de 10x40 quando o *Hash Power* Alvo aumenta o valor da altura máxima dos *forks* diminui , como pode ser visto na Figura 5.5.

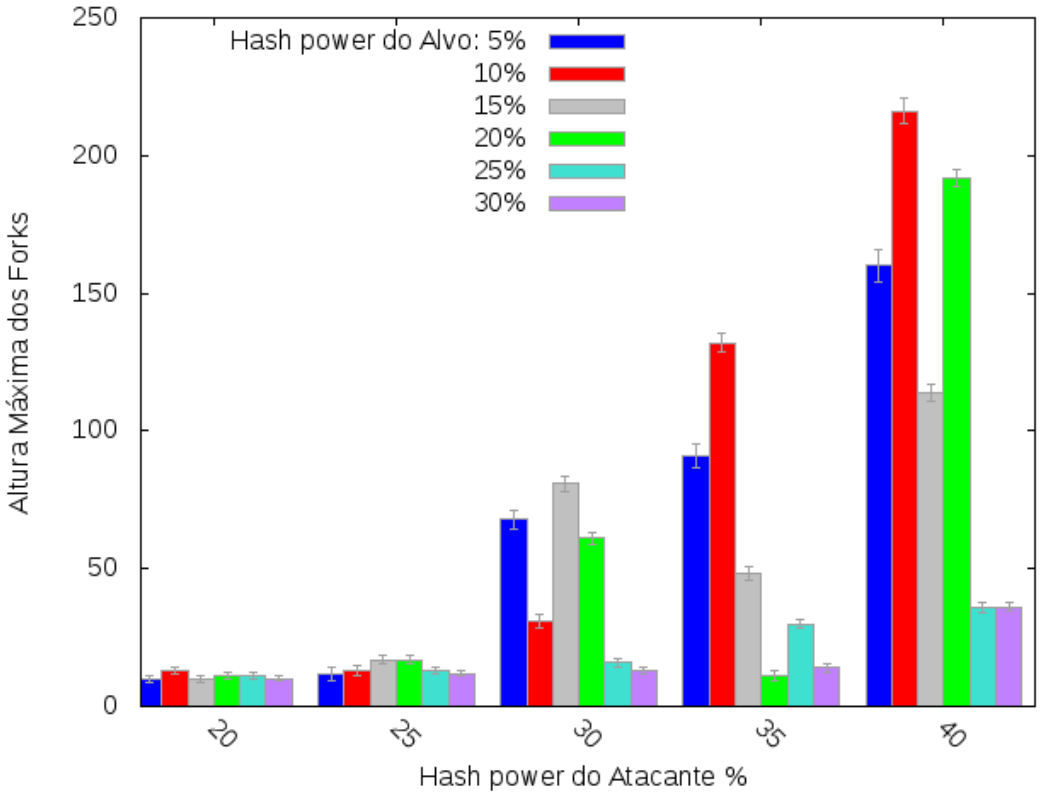

Figura 5.5: Altura Máxima dos Forks - Stalker Miner

A média das alturas máximas dos *forks* no minerador egoísta varia menos do que no perseguidor, porque as alturas máximas dos *forks* não são influenciadas pelo aumento do valor de *Hash Power* de um possível alvo. Já o aumento no *Hash Power* do atacante provoca um aumento na altura média do *Fork* porque o atacante consegue colocar mais blocos na cadeia adversária, como podemos ver na Figura 5.6.

No *Stalker Miner*, observamos que a altura média dos *Fork* aumenta com o aumento do Poder de *Hash* do Atacante e o comportamento da altura média dos *Forks* é inversamente proporcional ao *Hash Power* do Alvo, quanto maior o *Hash Power* do Alvo, menor o altura média dos *Forks*, como podemos ver na figura Figura: 5.7.

A Figura 5.8 mostra o comportamento da rede Bitcoin, em relação à freqüência de *forks* sem a presença de ataques. Os *forks* ocorrem de forma natural por vários motivos,

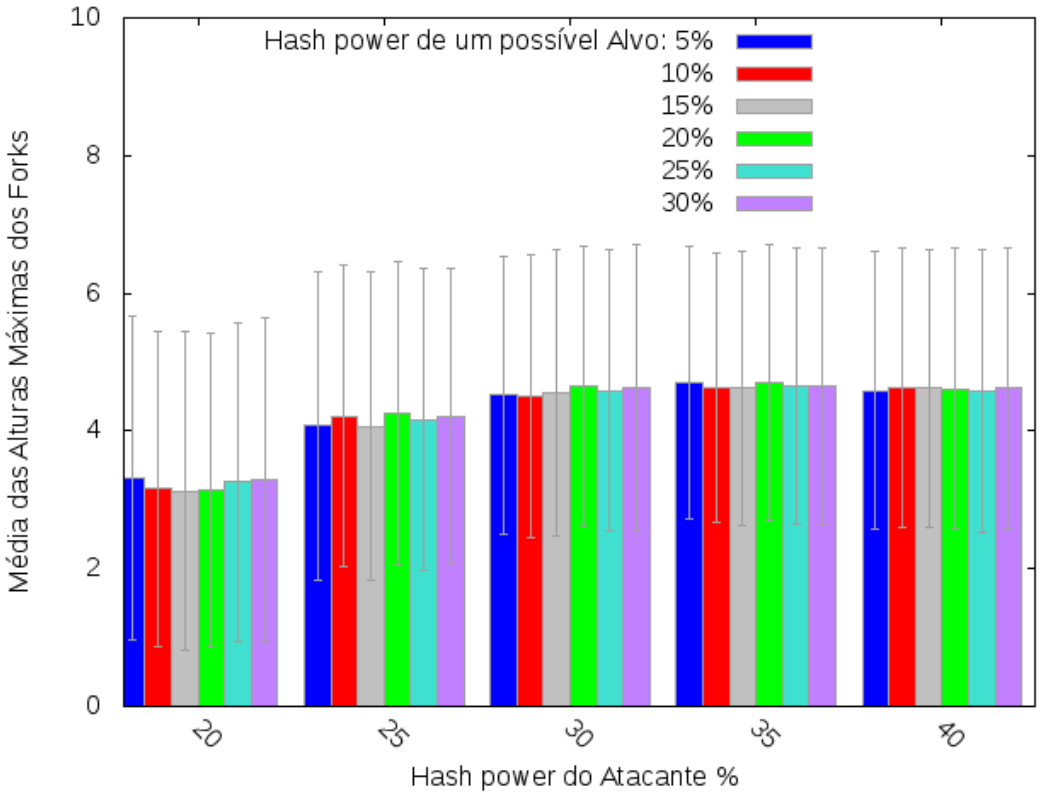

Figura 5.6: Média das Alturas Máximas dos Forks - Selfish Miner

porém com uma frequência bem pequena.

A Figura 5.9 e a Figura 5.10 mostram o comportamento dos ataques em relação à freqüência de *forks* apresentados para um determinado valor de *Hash Power* para AlvoxAtacante igual a 15x35.

O ataque *Stalker* tem uma frequência maior de *forks*, porém com vários *forks* com alturas menores e alguns picos de *forks*. Embora o *Stalker* tenha alturas máximas de *forks* muito maiores do que o *Selfish miner*, as alturas médias são bastante próximas, uma vez que estes picos ocorrem em poucas quantidades e há muitas ocorrências de *forks* com alturas muito pequenas.

O *Selfish miner* tem uma frequência menor. Uma das razões reside no propósito do ataque, o minerador egoísta só ataca quando ele tem uma chance real de derrubar a corrente honesta, a fim de obter a recompensa financeira por esse bloco.

O objetivo do *Stalker* não é financeiro, mas sim impedir que um determinado nó seja capaz de ter seu bloco publicado na cadeia, portanto, sempre que o nó de destino colocar seu bloco na tentativa de uma possível publicação, o nó atacante também colocará o seu bloco na disputa.

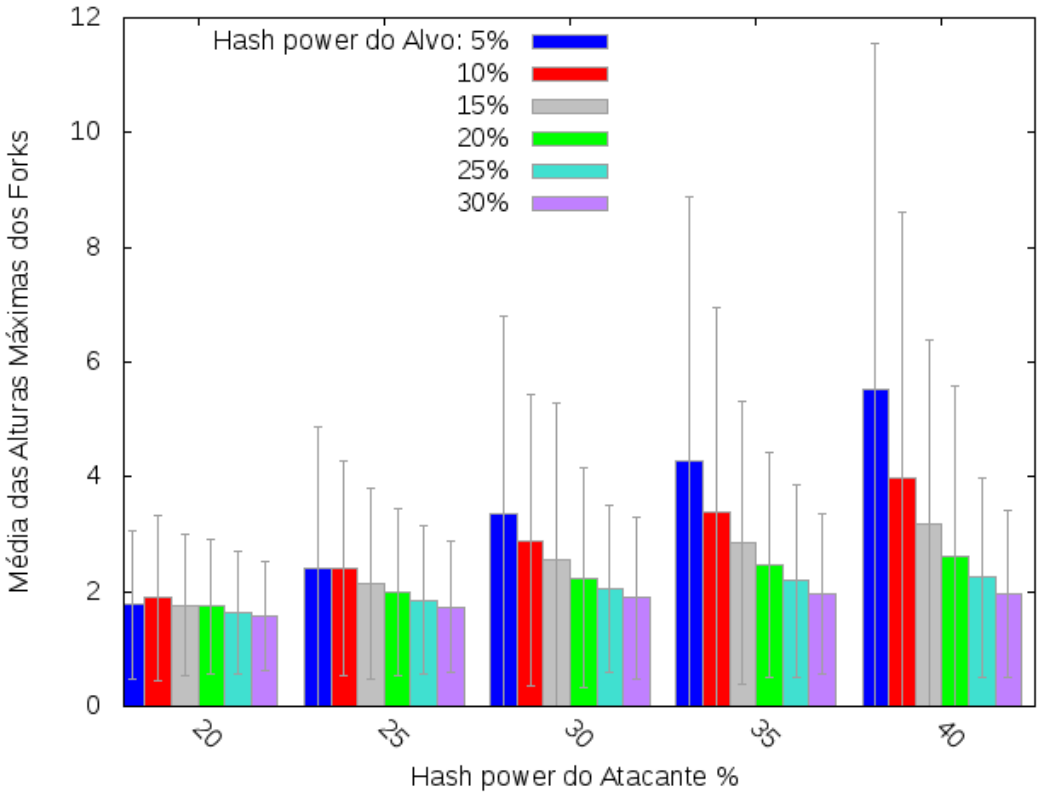

Figura 5.7: Média das Alturas Máximas dos Forks - Stalker Miner

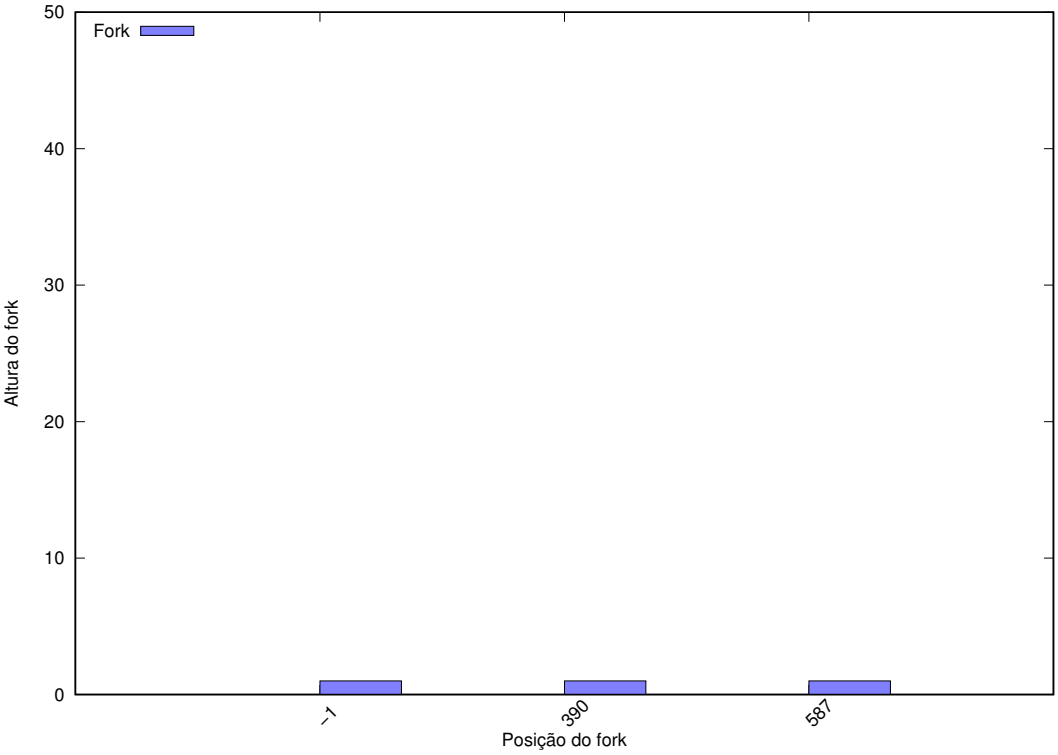

Figura 5.8: Posição de Forks para um determinado valor de Hash Power para AlvoxAtacante igual a 15x35 - Selfish Miner

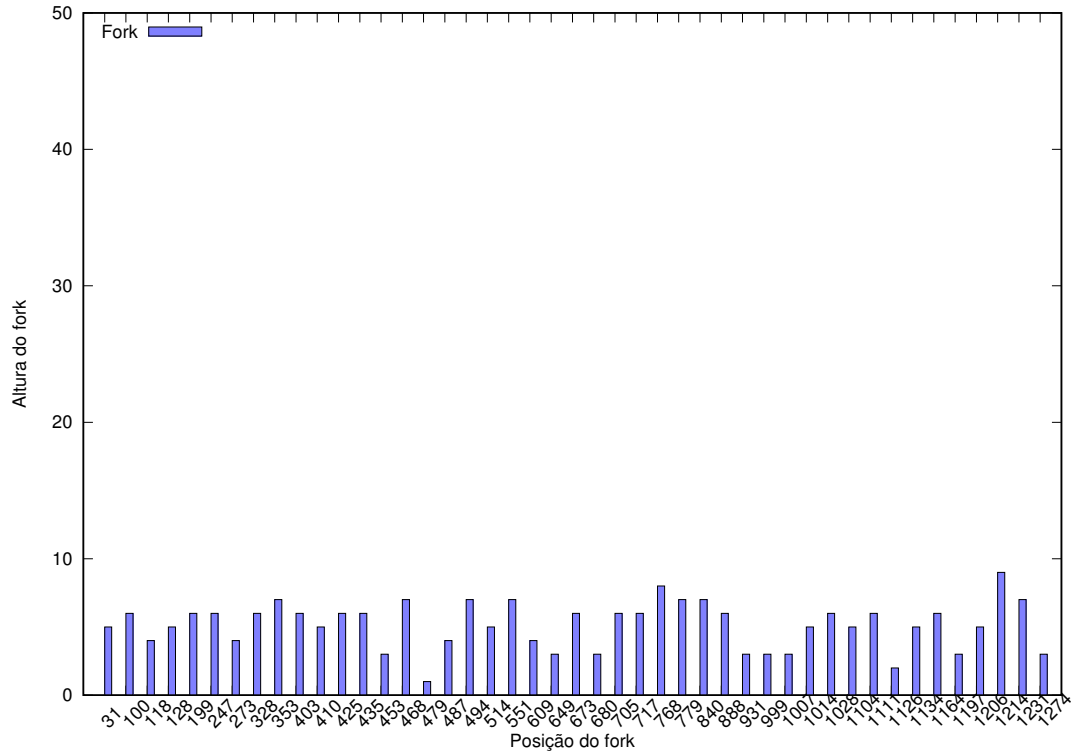

Figura 5.9: Posição de Forks para um determinado valor de Hash Power para AlvoxAtacante igual a 15x35 - Selfish Miner

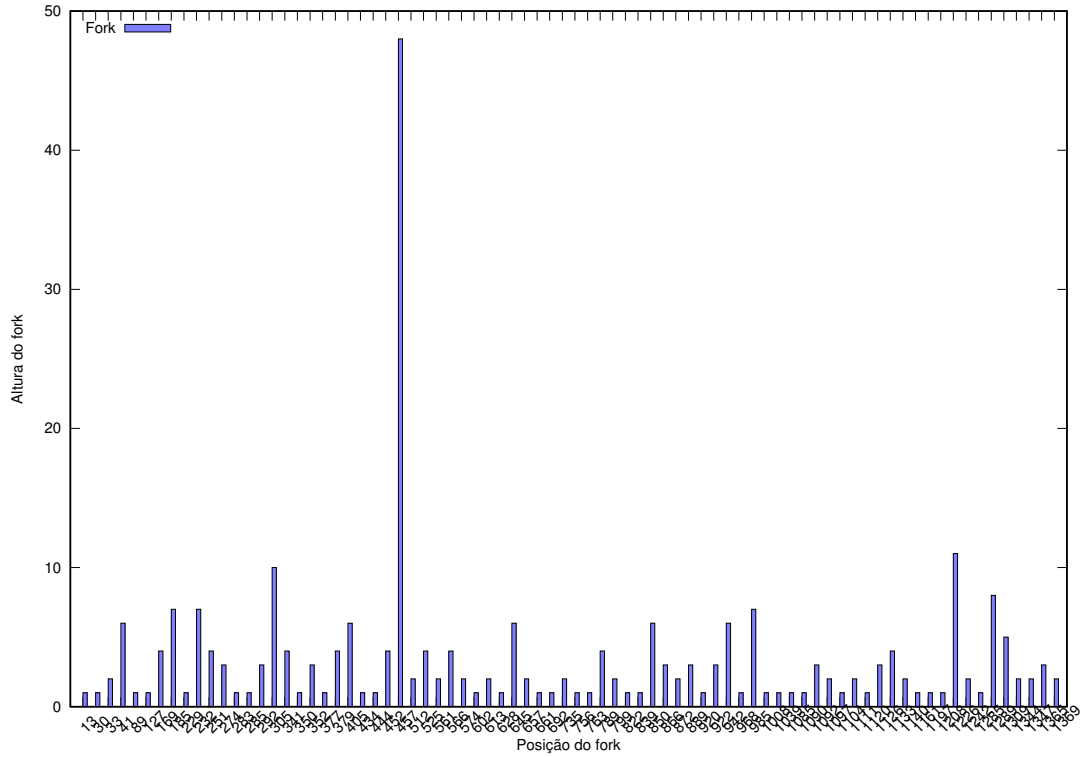

Figura 5.10: Posição de Forks para um determinado valor de Hash Power para AlvoxAtacante igual a 15x35 - Stalker Miner

Em média, a cada 10 minutos, um bloco é colocado na cadeia, nessa frequência, para o ataque do minerador Egoísta, a uma chance de 1,35% de derivar em um *forks*, enquanto, no Perseguidor, essa chance é maior, sendo 3,35%. Isto é, no Perseguidor, com cada bloco colocado, a chance de virar um *fork* é duas vezes maior do que na mineração egoísta.

Realizamos algumas alterações no código do simulador de Rede Bitcoin com os módulo de ataque de mineração egoísta [23] e o módulo do ataque de mineração perseguidora [32], de modo que durante a simulação de cada ataque, obtivessemos as informações de cada fork que ocorreu na Rede. A Figura5.11 apresenta um exemplo dos resultados extraídos.

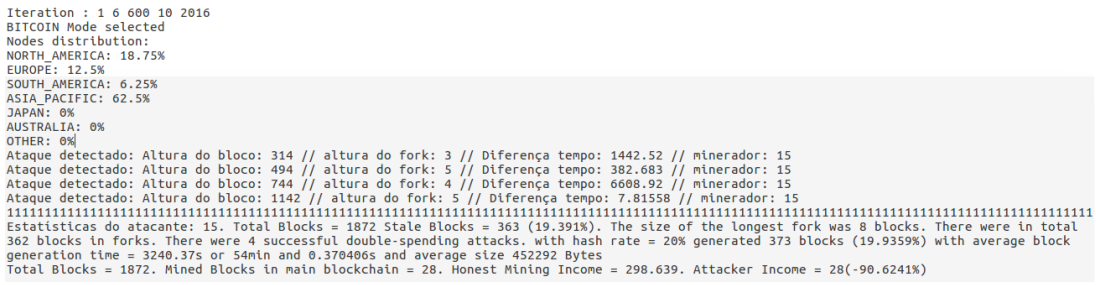

Figura 5.11: Resultado da Simulação de Detecção de Ataque

Os resultados encontrados nessas simulações serão as entradas para o algoritmo de Classificação da Detecção de Ataques.

5.3 Avaliação da Detecção de Ataque

Com base na análise das estatísticas extraídas durante as simulações realizadas para cada ataque, observou-se que um fork de altura igual ou superior a 2 já poderia caracterizar um ataque. Para verificar a veracidade dessa heurística, construímos o algoritmo 2 com o objetivo de verificar a taxa obtida de falsos positivos (quando o detector registra um ataque por engano) e falsos negativos (quando o detector falha em detectar um ataque).

Neste trabalho, a detecção de um ataque é simplesmente realizada observando em tempo real a altura de cada fork. Se uma altura do fork maior que α for observada, um ataque será registrado. Nas experiências, testamos $\alpha \geq 1$ e $\alpha \geq 2$. Espera-se que, ao usar $\alpha \geq 1$, o detector possa gerar um grande número de falsos positivos, mas nunca apresentará um falso negativo, enquanto o uso de $\alpha \geq 2$ pode gerar menos falsos positivos, mas pode apresentar alguns falsos negativos.

Algoritmo 2: Classificação da Detecção de Ataque

```
1  h := Altura do Fork;
2  if h ≥ α then
3  |  registrar ataque;
4  |  if ausença de blocos do atacante then
5  |  |  Falso Positivo;
6  |  end
7  else
8  |  não registrar ataque;
9  |  if presença de blocos do atacante then
10 |  |  Falso Negativo;
11 |  end
12 end
```

Foram realizadas simulações para todo o *Hash Power* proposto para o atacante e para um possível nó de destino no caso do ataque de Mineração Egoísta. Os resultados observados mostraram que a porcentagem de Falsos Positivos obtidos para o Detector $\alpha \geq 2$ foi muito pequena em todas as distribuições de Hash Power, com um máximo de 0,38% para um *Hash Power* Alvo × Atacante de 15x20 e 0,13% para um *Hash Power* de 25x25, como mostra a Figura 5.12, provando que, para o Selfish Miner, esse Detector é mais eficiente que o Detector $\alpha \geq 1$.

A mesma análise foi realizada para o ataque Stalker e o Detector de garfos com $\alpha \geq 2$ provou ser mais eficiente, apresentando uma taxa de Falso Positivo menor que o Detector de forks $\alpha \geq 1$, como mostra a Figura 5.13.

Comparando o desempenho para os dois tipos de ataques, com base no número de falsos positivos, para o ataque de Mineração Egoísta, o detector foi mais eficiente do que para o Perseguidor, apresentando uma menor taxa de falsos positivos, conforme visto nas Figuras 5.12 e 5.13.

Outra análise foi realizada com base no número de falsos negativos encontrados para o Detector $\alpha \geq 2$, pois para esse detector, considera-se que qualquer fork de altura igual a um não é considerado um ataque.

Observamos um comportamento semelhante para o ataque egoísta e o perseguidor, como pode ser visto nas figuras 5.14 e 5.15. Quando o poder de *hash* do atacante é baixo, há um número baixo de falsos negativos, à medida que o poder de *hash* do atacante

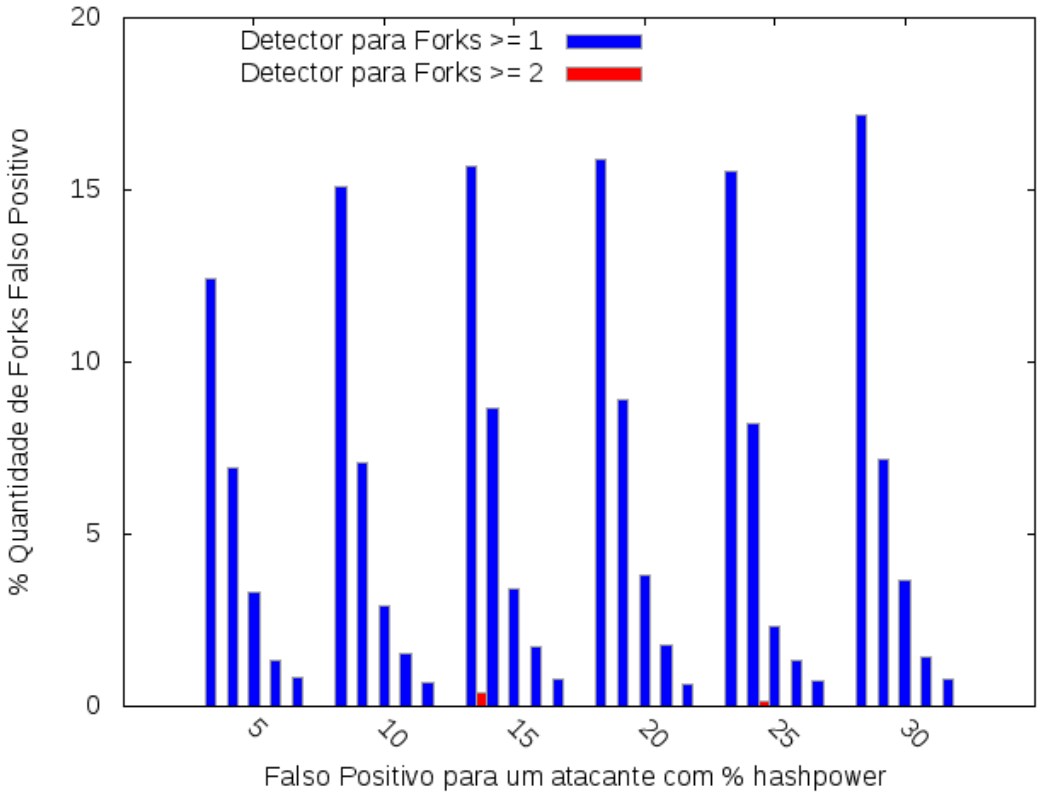

Figura 5.12: Detecção de Falsos Positivos do Ataque *Selfish Miner* para $\alpha \geq 1$ and $\alpha \geq 2$

aumenta, também é observado um aumento no número de falsos negativos. Isso acontece porque, como o atacante tem mais poder de *hash*, ele tem maior capacidade de colocar mais blocos na cadeia principal, aumentando a probabilidade de *forks*. Assim, também teremos um aumento no número de *forks* igual a um.

Observe, no entanto, que quando o atacante coloca somente um bloco e o fork atinge a altura igual a 1 na cadeia principal, ele não está realmente atacando a rede. O minerador atacante também coloca blocos honestamente quando verifica a possibilidade de obter a recompensa pelo bloco, sem ser um ataque. Podemos, portanto, considerar que os *forks* considerados como falsos negativos não eram de fato ataques, mas um comportamento honesto do atacante em busca da recompensa pela mineração de um bloco.

Observamos que, quando um atacante coloca os blocos de maneira maliciosa, a diferença entre o tempo de publicação do primeiro bloco e o último bloco desse minerador é muito menor do que o tempo de um minerador honesto, uma vez que o atacante secretamente extrai os blocos e os mantém oculto, publicando tudo de uma só vez quando ele observa uma grande chance de suplantar a cadeia principal, diminuindo muito o tempo de publicação de seus blocos.

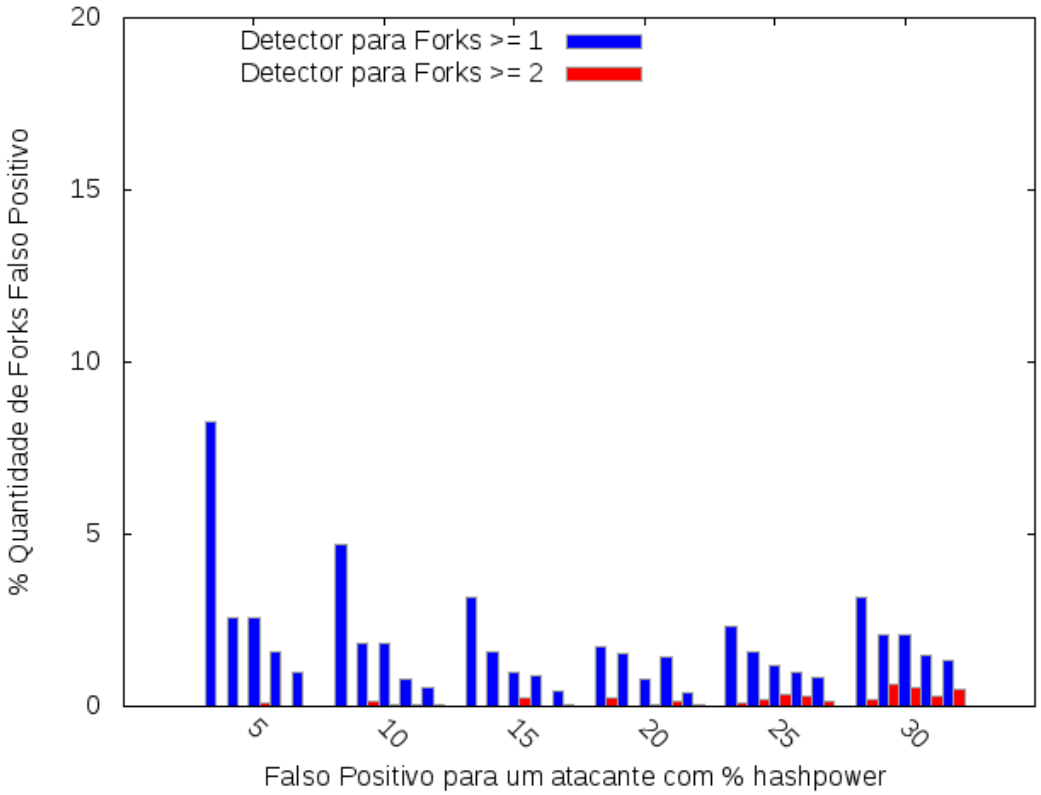

Figura 5.13: Detecção de Falsos Positivos do Ataque *Stalker Miner* para $\alpha \geq 1$ and $\alpha \geq 2$

A média de falsos negativos encontrados para as simulações realizadas com todas as variações de poder de *hash* alvo x atacante foi de 14,14% para o ataque de *Selfish Miner* e 16,25% para o ataque *Stalker*.

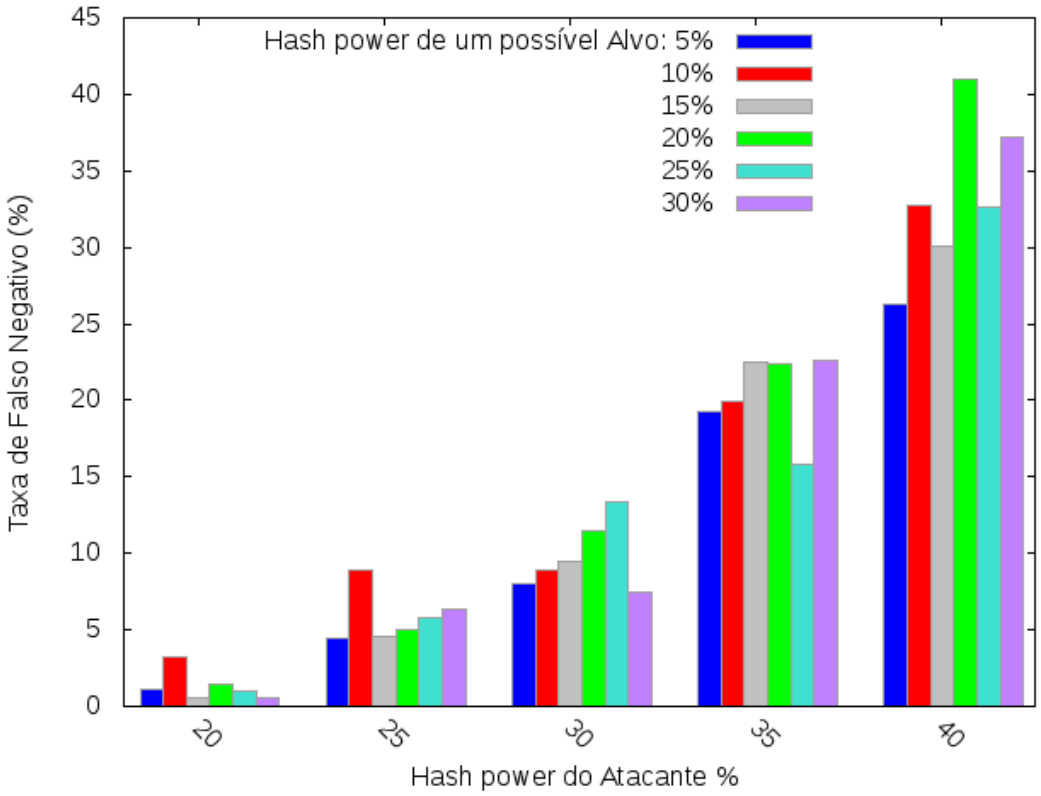

Figura 5.14: Detecção de Falso Negativo para ataques de *Selfish Miner* para $\alpha = 2$.

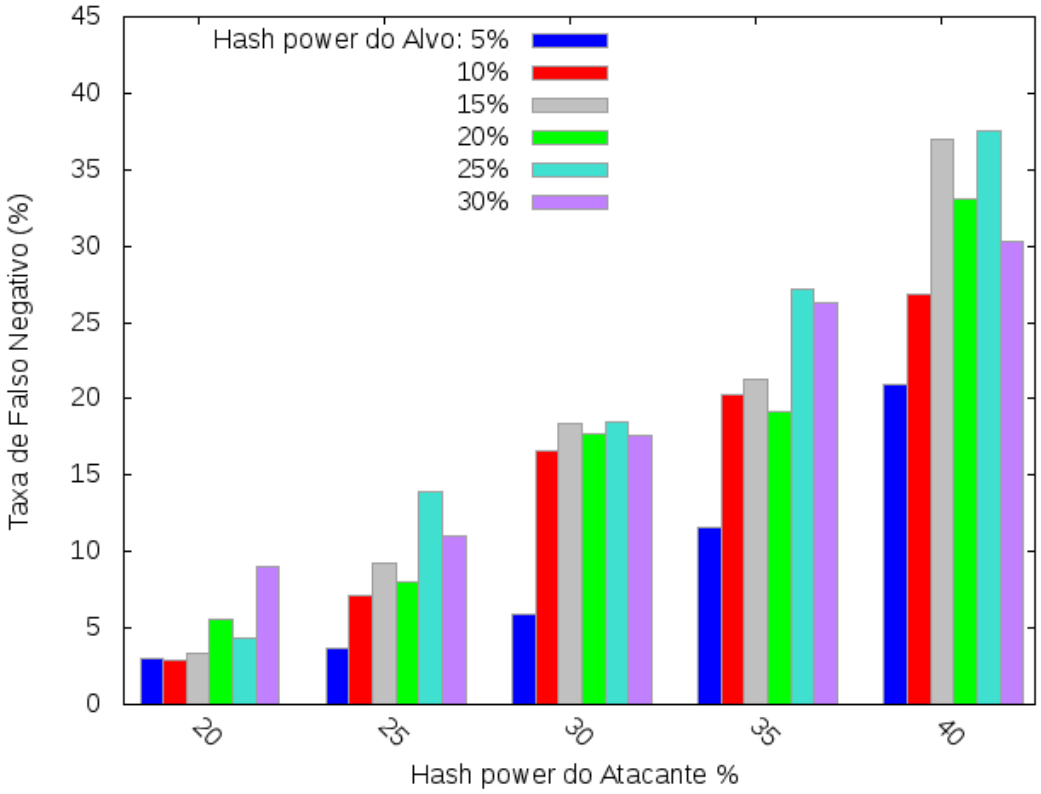

Figura 5.15: Detecção de Falso Negativo para ataques de *Stalker Miner* para $\alpha = 2$.

Capítulo 6

Conclusão

Blockchain apresenta muitas oportunidades promissoras. Não só permite o movimento do dinheiro, mas também pode ser usado para transferir informações e servir como um meio de autorização e autenticação entre dispositivos, permitindo o uso de Blockchain como um serviço [58], podendo ser aplicada numa grande variedade de negócios e podendo ser adequada para alguns usos específicos. Mas, independentemente do contexto, uma rede Blockchain vai ser construída com base em um protocolo que define como o sistema deve funcionar, então todas as diferentes partes do sistema e todos os participantes da rede precisam seguir as regras do protocolo.

Muitos desafios, no entanto, permanecem, como modelos de consenso e os altos custos computacionais envolvidos na verificação das transações. Devido à sua natureza descentralizada e pública, pode ser alvo de ataques ao mecanismo de consenso, que é o coração do protocolo. No contexto das criptomoedas, os algoritmos de consenso são cruciais, visto que são responsáveis por manter a integridade e segurança das redes Blockchain, eles garantem que as regras do protocolo sejam seguidas e que todas as transações ocorram de forma confiável, fazendo com que cada moeda seja gasta uma única vez. Pesquisadores e empresas estão constantemente buscando desenvolver mecanismos para detectar e prevenir ataques.

Neste trabalho, apresentamos uma heurística de detecção para os ataques conhecidos como Selfish Miner e Stalker em Blockchains que utilizam o mecanismo de consenso mais comum conhecido como Prova de trabalho (Pow). Esses dois tipos de ataques pretendem suplantar a cadeia honesta usando o mecanismo de consenso, minerando uma "cadeia secreta"e revelando-a apenas quando essa cadeia for maior que a cadeia honesta. A heurística proposta baseia-se na análise das alturas dos forks, que são bifurcações que podem surgir naturalmente na Blockchain até que o consenso seja atingido.

Foram realizadas análises utilizando o simulador NS3 e os resultados mostraram que quando a blockchain apresenta forks com altura maior ou igual a 2 existe uma grande probalilidade da cadeia está sofrendo um dos ataques conhecidos como selfish miner ou stalker. Quando aplicamos o algoritmo de classificação da detecção de ataques, a probabilidade de detecção de um desses ataques é muito grande, sendo 99,98% no caso do Selfish Miner e 99,85% no caso de ataques de Stalker.

6.1 Contribuições

Como contribuições deste trabalho, pode-se citar:

- O desenvolvimento de uma heurística para detectar ataques do tipo Selfish Miner e Stalker nas Blockchains que utilizam o mecanismo de consenso conhecido como prova de trabalho - PoW. ;
- Implementação do código do algoritmo de Classificação da Detecção de ataque no NS-3;
- Publicação do minicurso: Uso de Blockchain para Privacidade e Segurança em Internet das Coisas, no livro de minicursos do VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais [13];
- Três artigos publicados:
  - *A Survey of How to Use Blockchain to Secure Internet of Things and the Stalker Attack.* Publicado pela Security and Communication Networks 2018;
  - Stalker - Uma Nova Estratégia para o Atacante Egoísta em Blockchains. Apresentado no Simpósio Brasileiro de Redes de Computadores (SBRC) (2018), vol. 36; e
  - *A Heuristic for the Detection of Selfish Miner and Stalker attacks in Blockchains Networks..* Apresentado na 1st Blockchain, Robotics and AI for Networking Security Conference BRAINS 2019.

6.2 Trabalhos Futuros

Apesar de possuírem intenções diferentes para a realização do ataque, enquanto o Selfish miner a motivação é financeira, o nó malicioso minera uma cadeia secreta, com a

intenção de revelar quando essa cadeia secreta atingir um tamanho maior que a honesta, recebendo o pagamento pelos blocos minerados, o Stalker o objetivo é negar a publicação de um bloco específico de um nó, negando o serviço a esse nó alvo. O método utilizado por ambos ataques é similar, minerando secretamente uma cadeia, ou seja promovendo um *fork* temporário na cadeia, até que haja um momento oportuno para revelá-Ía, corrompendo o mecanismo de consenso.

Este processo causa anomalias, dentre as quais pode-se citar: aumento da taxa de forks, altura dos forks, e o recebimento de inúmeros blocos em um curto intervalo de tempo. Pesquisas futuras são necessárias para melhorar a heurística de detecção de ataques, uma possibilidade é realizar uma análise da frequência de forks específica para cada tipo de ataque, não apenas para identificar que o comportamento malicioso está ocorrendo, mas também para identificar qual tipo de ataque está ocorrendo.

Pesquisar uma correlação entre as alterações identificadas na cadeia também é uma outra heurística possível, como por exemplo a correlação entre a presença de foks de altura maior que 2 e o curto intervalo de inclusão/geração de blocos.

Referências

[1] Aggarwal, D.; Brennen, G. K.; Lee, T.; Santha, M.; Tomamichel, M. Quantum attacks on bitcoin, and how to protect against them. *arXiv preprint arXiv:1710.10377* (2017).

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

[3] Andrea, A. *Mastering BitCoin*, vol. 50. 2014.

[4] Back, A., et al. Hashcash-a denial of service counter-measure, 2002.

[5] Bissias, G.; Ozisik, A. P.; Levine, B. N.; Liberatore, M. Sybil-resistant mixing for bitcoin. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (2014), ACM, pp. 149–158.

[6] Bitcoin. Bitcoin developer reference. https://bitcoin.org/en/developer-reference, 2009. Accessed: 2017-07-30.

[7] Blockchain Luxembourg S.A. Bitoin's block explorer. https://www.blockchain.info. Acessado em: 10/01/2018.

[8] Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13*, 7 (1970), 422–426.

[9] Broder, A.; Mitzenmacher, M. Network applications of bloom filters: A survey. *Internet mathematics 1*, 4 (2004), 485–509.

[10] Cachin, C. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers* (2016).

[11] Carneiro, G. Ns-3: Network simulator 3. In *UTM Lab Meeting April* (2010), vol. 20.

[12] Castro, M.; Liskov, B. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS) 20*, 4 (2002), 398–461.

[13] Chicarino, V. R.; Jesus, E. F.; Albuquerque, C. V. N.; Aragão Rocha, A. A. Uso de blockchain para privacidade e segurança em internet das coisas. *Livro de Minicursos do VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Brasília: SBC,* (2017).

[14] Chirgwin, R. Android bug batters bitcoin wallets. *The Register 3* (2013).

[15] Cramer, R.; Damgård, I.; Dziembowski, S.; Hirt, M.; Rabin, T. Efficient multiparty computations secure against an adaptive adversary. In *Eurocrypt* (1999), vol. 99, Springer, pp. 311–326.

[16] Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. *Applied Innovation 2* (2016), 6–10.

[17] Decker, C.; Wattenhofer, R. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (2013), IEEE, pp. 1–10.

[18] Dorri, A.; Kanhere, S. S.; Jurdak, R. Blockchain in internet of things: Challenges and Solutions. *arXiv:1608.05187 [cs]* (2016).

[19] Dorri, A.; Kanhere, S. S.; Jurdak, R.; Gauravaram, P. Blockchain for iot security and privacy: The case study of a smart home, 2017.

[20] Douceur, J. R. The sybil attack. In *International Workshop on Peer-to-Peer Systems* (2002), Springer, pp. 251–260.

[21] Eyal, I.; Sirer, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 436–454.

[22] Gennaro, R.; Rabin, M. O.; Rabin, T. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing* (1998), ACM, pp. 101–111.

[23] Gervais, A.; Karame, G. O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the Security and Performance of Proof of Work Blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16* (2016).

[24] Gutteridge, D. Japanese cryptocurrency monacoin hit by selfish mining attack. *CCN https://www. ccn. com/japanese-cryptocurrency-monacoin-hit-by-selfish-mining-attack/(retrieved 2018-10-16)* (2018).

[25] Hankerson, D.; Menezes, A. J.; Vanstone, S. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[26] Hari, A.; Lakshman, T. The internet blockchain: A distributed, tamper-resistant transaction framework for the internet. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (2016), ACM, pp. 204–210.

[27] Heilman, E.; Kendler, A.; Zohar, A.; Goldberg, S. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security* (2015), pp. 129–144.

[28] Herrera-Joancomartí, J. Research and challenges on bitcoin anonymity. In *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*. Springer, 2015, pp. 3–16.

[29] Hotz, G. Console hacking 2010-ps3 epic fail. In *27th Chaos Communications Congress* (2010).

[30] Intel. Proof of elapsed time (poet). available from: http://intelledger.github.io/.

[31] Jansma, N.; Arrendondo, B. Performance comparison of elliptic curve and rsa digital signatures. *nicj. net/files* (2004).

[32] Jesus, E. F.; Chicarino, V. R.; Albuquerque, C.; Antônio, A. d. A. Stalker-uma nova estratégia para o atacante egoísta em blockchains. In *Simpósio Brasileiro de Redes de Computadores (SBRC)* (2018), vol. 36.

[33] Jesus, E. F.; Chicarino, V. R.; de Albuquerque, C. V.; Rocha, A. A. d. A. A survey of how to use blockchain to secure internet of things and the stalker attack. *Security and Communication Networks 2018* (2018).

[34] Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security 1*, 1 (2001), 36–63.

[35] Kim, J. Safety, liveness and fault tolerance—the consensus choices stellar, 2014.

[36] Kim, T. H. A study of digital currency cryptography for business marketing and finance security. *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology 6*, 1 (2016), 365–376.

[37] King, S.; Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August 19* (2012).

[38] Koshy, P.; Koshy, D.; McDaniel, P. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 469–485.

[39] Mayer, H. Ecdsa security in bitcoin and ethereum: a research survey. *CoinFaabrik, June 28* (2016), 126.

[40] Merkle, R. C. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques* (1987).

[41] Miller, A.; Litton, J.; Pachulski, A.; Gupta, N.; Levin, D.; Spring, N.; Bhattacharjee, B. Discovering bitcoin's public topology and influential nodes.(2015), 2015.

[42] Moser, M.; Bohme, R.; Breuker, D. An inquiry into money laundering tools in the bitcoin ecosystem. In *eCrime Researchers Summit (eCRS), 2013* (2013), IEEE, pp. 1–14.

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

[44] Natoli, C.; Gramoli, V. The blockchain anomaly. In *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on* (2016), IEEE, pp. 310–317.

[45] Nayak, K.; Kumar, S.; Miller, A.; Shi, E. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on* (2016), IEEE, pp. 305–320.

[46] Nguyen, K. T.; Laurent, M.; Oualha, N. Survey on secure communication protocols for the internet of things. *Ad Hoc Networks 32* (2015), 17–31.

[47] Noizat, P. Blockchain electronic vote. In *Handbook of digital currency*. Elsevier, 2015, pp. 453–461.

[48] Panikkar, S.; Nair, S.; Brody, P.; Pureswaran, V. Adept: An iot practitioner perspective. *IBM Institute for Business Value* (2014).

[49] Pilkington, M. Blockchain technology: principles and applications. research handbook on digital transformations, edited by f. xavier olleros and majlinda zhegu, 2016.

[50] Pornin, T. Rfc6979: Deterministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm (ecdsa)(2013).

[51] Ross, S. M. Stochastic processes. 1996, 1996.

[52] Sapirshtein, A.; Sompolinsky, Y.; Zohar, A. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security* (2016), Springer, pp. 515–532.

[53] Sayeed, S.; Marco-Gisbert, H. Assessing blockchain consensus and security mechanisms against the 51% attack. *Applied Sciences 9*, 9 (2019), 1788.

[54] Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (1994), Ieee, pp. 124–134.

[55] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review 41*, 2 (1999), 303–332.

[56] Spagnuolo, M.; Maggi, F.; Zanero, S. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 457–468.

[57] Swan, M. *Blockchain: Blueprint for a new economy*. "O'Reilly Media, Inc.", 2015.

[58] Swanson, T. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. *Report, available online, Apr* (2015).

[59] Ulrich, F. *Bitcoin: a moeda na era digital*. LVM Editora, 2017.

[60] Valenta, L.; Rowan, B. Blindcoin: Blinded, accountable mixes for bitcoin. In *International Conference on Financial Cryptography and Data Security* (2015), Springer, pp. 112–126.

[61] Valsorda, F. Exploiting ecdsa failures in the bitcoin blockchain. *Proceedings of Hack In The Box (HITB)* (2014).

[62] Vasin, P. Blackcoin's proof-of-stake protocol v2, 2014.

[63] Weingartner, E.; Vom Lehn, H.; Wehrle, K. A performance comparison of recent network simulators. In *Communications, 2009. ICC'09. IEEE International Conference on* (2009), IEEE, pp. 1–5.

[64] Wilmoth, J. Bitcoin gold hit by double spend attack, exchanges lose millions, 2018.

[65] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper 151* (2014).

[66] Wörner, D.; von Bomhard, T. When your sensor earns money. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct* (2014).

[67] Wörner, D.; von Bomhard, T. When your sensor earns money: exchanging data for cash with bitcoin. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (2014), ACM, pp. 295–298.

[68] Wuille, P. Bip32: Hierarchical deterministic wallets. *h ttps://github. com/genjix/bips/blob/master/bip-0032. md* (2012).

[69] Yao, A. C. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on* (1982), IEEE, pp. 160–164.

[70] Yue, X.; Wang, H.; Jin, D.; Li, M.; Jiang, W. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of medical systems 40*, 10 (2016), 218.

[71] Zhang, Y.; Wen, J. An iot electric business model based on the protocol of bitcoin. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on* (2015), IEEE, pp. 184–191.

[72] Zhao, Z.; Chan, T.-H. H. How to vote privately using bitcoin. In *International Conference on Information and Communications Security* (2015), Springer, pp. 82–96.

[73] Zyskind, G.; Nathan, O.; Pentland, A. Enigma: Decentralized Computation Platform with Guaranteed Privacy. *arXiv:1506.03471 [cs]* (2015).