UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

José Eduardo de Azevedo Sousa

Uma Análise de Correlação entre Tempo de Espera e Transações no Ethereum

Juiz de Fora
2020

José Eduardo de Azevedo Sousa

Uma Análise de Correlação entre Tempo de Espera e Transações no Ethereum

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Alex Borges Vieira

Coorientadores: Prof. Dr Heder S. Bernardino, Glauber D. Gonçalves

Juiz de Fora

2020

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Sousa, José Eduardo de Azevedo.
Uma Análise de Correlação entre Tempo de Espera e Transações no Ethereum / José Eduardo de Azevedo Sousa. – 2020.
75 f. : il.

Orientador: Alex Borges Vieira
Coorientadores: Heder S. Bernardino, Glauber D. Gonçalves
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2020.

1. Ethereum. 2. Correlação de atributos. 3. Aprendizado de Máquina. I. Vieira, Alex Borges, orient. II. Bernardino, Heder Soares, coorient. III. Gonçalves, Glauber Dias, coorient. IV. Título.

José Eduardo de Azevedo Sousa

Uma Análise de Correlação entre Tempo de Espera e Transações no Ethereum

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 16 de outubro de 2020

BANCA EXAMINADORA

Prof. Dr. Alex Borges Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Heder Soares Bernardino - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Glauber Dias Gonçalves - Coorientador
Universidade Federal do Piauí

Prof. Dr. Allan Edgard Silva Freitas
Instituto Federal da Bahia

Prof. Dra. Heloísa Pinna Bernardo
Universidade Federal de Juiz de Fora

Prof. Dr. Saulo Moraes Villela
Universidade Federal de Juiz de Fora

Dedico este trabalho à minha família, que tornou tudo isso possível.

AGRADECIMENTOS

Agradeço em primeiro lugar à minha família, a qual esteve sempre presente em todos os momentos de minha vida, sejam eles os mais felizes que já tive ou em pontos delicados. Sem a compreensão deles sobre o sacrifício necessário para esta conquista, este sonho não se concretizaria. A participação desta entidade na minha vida serve como pilar para que possa me inspirar em valores idôneos e buscar cada vez mais conhecimento.

Agradeço meus professores, os quais proveram conhecimento e direção. Tivemos uma jornada longa, envolvendo sabedoria, paciência, resiliência e que contribuíram para me tornar quem sou. O trabalho de vocês me edificou, e por isso o meu muito obrigado.

Dentre o corpo docente gostaria de ressaltar meus agradecimentos especiais a quem necessitei de grande apoio para traçar este caminho, que são meu orientador e coorientadores Prof. Dr. Alex Vieira, Prof. Dr. Heder Bernardino e Prof. Dr. Glauber Gonçalves. Várias horas empreendidas neste curso foram junto a essas pessoas que dedicam suas vidas a trazer luz e educação.

Um agradecimento especial também para ao professor Saulo Moraes Vilela, que me ajudou a fazer ciência. Buscou, com paciência e entusiasmo, trazer luz às minhas ideias, para que pudéssemos alcançar objetivos acadêmicos. Aos estudantes Vinícius Carlos Oliveira e Júlia Valadares meu eterno agradecimento pelos diversos momentos os quais me assistiram para execução das nossas pesquisas.

Aos meus amigos, que me deram suporte emocional, agradeço o companheirismo. Muitos deles participaram indiretamente da minha vida acadêmica, cooperando com a troca de conhecimentos, cobranças e apoio.

Por fim, agradeço ao PPGCC e a UFJF pela oportunidade a mim concedida.

"It is the question, not the answer,
that is the higher view."
(ZHU)

RESUMO

Iniciada em julho de 2015, Ethereum é a segunda maior criptomoeda em volume de transações nesse momento, atrás apenas do Bitcoin, que é a criptomoeda pioneira. A popularidade de Ethereum está em franca ascensão, atraindo a comunidade científica. Uma importante questão em discussão pelo comportamento da plataforma diz respeito à cobrança obrigatória da taxa por transação no Ethereum e a sua possível relação com o tempo de espera para confirmação da transação. Ao enviarem suas transações a outros utilizadores da rede, os usuários são obrigados a fornecer uma espécie de gorjeta, a taxa de mineração, para que sua transação seja incluída na rede. A crença comum na rede Ethereum é que quanto maior a taxa, menor o tempo de confirmação.

No presente trabalho, foi realizada uma investigação e análise detalhada dessa questão. Para conduzir esse trabalho, foram coletadas 7.2 milhões de transações no Ethereum. A seguir, foram realizadas diversas análises acerca das taxas fornecidas pelos usuários e o tempo de espera para uma transação ser minerada. Primeiramente foi verificado se existe alguma correlação entre o tempo de espera de uma transação para adentrar a rede e diversos recursos relacionados às taxas pagas aos operadores da rede Ethereum para execução de transações, por meio de correlações de Pearson e Spearman. Posteriormente, análises agrupando as transações por características são conduzidas de duas formas: (i) por valores absolutos e (ii) por grupos definidos a partir de algoritmos de clusterização. Esses agrupamentos buscam encontrar usuários com comportamentos parecidos, tentando encontrar correlação entre o tempo de espera das transações nesses casos. As análises empíricas mostraram fortes evidências de que não há uma correlação clara entre as taxas pagas pelos usuários e o tempo de espera da transação, concluindo que tais taxas, assim como as características do sistema delas derivadas, inerentes a plataforma Ethereum, não determinam o tempo de espera.

Além da avaliação da relação existente entre tempo de confirmação e taxa, nesse trabalho, foi avaliado ataque de negação de serviço, um problema que também pode afetar o tempo de confirmação de transações na rede Ethereum. Inicialmente, foi definido um ataque semelhante ao de negação de serviço distribuído (DDoS), em que usuários maliciosos inserem transações específicas, as quais possuem valores muito baixos de *gas price*, com intuito de provocar lentidão na rede são inseridas na rede, aumentando o tempo de espera das transações. Para estudar este ataque, aqui denominado de *Penny Attack*, foi definida uma arquitetura capaz de replicar a rede ETH do Ethereum em uma estrutura local. Em seguida foram executadas simulações para verificar a resposta do sistema em situações normais da rede, em período sobre ataque, e também em períodos de alta demanda da rede. Por fim, foram propostas sete técnicas de aprendizado de máquina buscando melhores métricas de classificadores capazes de identificar o ataque aqui proposto, onde foram

obtidos métodos com acurácia maior que 0,95 e AUC de 0,89, fornecendo uma modelo confiável para detecção deste ataque.

Palavras-chave: Ethereum. Correlação de atributos. Aprendizado de máquina. Penny Attack.

ABSTRACT

Started in July 2015, Ethereum is the second-largest cryptocurrency in terms of transaction volume at the moment, behind only Bitcoin, which is the pioneer cryptocurrency. Ethereum's popularity is on the rise, attracting the scientific community. An important issue under discussion regarding the behavior of the platform concerns the mandatory collection of the fee per transaction on Ethereum and its possible relationship with the pending time for confirmation of the transaction. When submitting their transactions to other users, they are required to provide a kind of tip, the mining fee, for their transaction to be included in the network. The common belief in the Ethereum network is that the higher the rate, the shorter the confirmation time.

In the present work, a detailed investigation and analysis of this issue were carried out. To conduct this work, 7.2 million transactions were collected on Ethereum. Next, several analyzes were performed on the fees provided by users and the pending time. Firstly, it is verified if there is any correlation between the transaction pending time to enter the network and several resources related to the fees paid to the operators of the Ethereum network for the execution of transactions, through Pearson and Spearman correlations. Subsequently, analyzes grouping transactions by characteristics are conducted in two ways: (i) by absolute values and (ii) by groups defined using clustering algorithms. These groups seek to find users with similar behaviors, trying to find a correlation between the pending time in these cases. Empirical analyzes showed strong evidence that there is no clear correlation between the fees paid by users and the transaction pending time, concluding that such fees, as well as the system characteristics derived from them, inherent to the Ethereum platform, do not determine the time waiting.

In addition to assessing the relationship between confirmation time and fee, in this work, a denial of service attack was assessed, a problem that can also affect the confirmation time of transactions on the Ethereum network. Initially, an attack similar to that of distributed denial of service (DDoS) was defined, in which malicious users with the intention of causing slowness in the network insert specific transactions in the network, containing low gas price, increasing the pending time for transactions. To study this attack, here called *Penny Attack*, an architecture was defined capable of replicating Ethereum's ETH network in a local structure. Then, simulations were performed to check the system's response in normal situations of the network, in periods of attack, and also in periods of high network demand. Finally, seven machine learning techniques were proposed seeking better metrics for classifiers capable of identifying the attack proposed here, where methods with accuracy greater than 0.95 and AUC of 0.89 were obtained, providing a reliable model for detecting this attack.

Keywords: Ethereum. Feature correlation. Machine learning. Penny Attack.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Aplication Programming Interface
AUC	Area Under The Curve
DDoS	Distributed Denial of Service
EVM	Ethereum Virutal Machine
KNN	K-Nearest Neighbors
P2P	Peer to Peer
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
ROC	Receiver Operating Characteristics
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UFJF	Universidade Federal de Juiz de Fora

SUMÁRIO

1 INTRODUÇÃO

Plataformas baseadas em *blockchain* tiveram um crescimento significativo nos últimos anos [Xu e Huang 2020, Meshcheryakov, Ivanov et al. 2020, Alonso 2020]. Neste contexto, o Bitcoin [Nakamoto 2008] é a primeira e mais conhecida plataforma da tecnologia. No entanto, desde a ascensão do Bitcoin, várias outras plataformas baseadas em *blockchain* emergiram. Por exemplo, o Ethereum [Wood et al. 2014] se tornou rapidamente popular, alcançando um valor de mercado de 27 bilhões de dólares[1], somando mais de 757 milhões de transações[2] desde a sua criação em julho de 2015 [Chen et al. 2020].

O principal aspecto que difere o Ethereum do Bitcoin é a possibilidade de realizar transações financeiras e aplicações distribuídas com contratos inteligentes, ou *smart contracts* em inglês [Bareis, Angelo e Salzer 2020]. Esses contratos inteligentes são programas de computador autônomos que executam lógica predefinida quando ativados. O Ethereum, então, dispõe de maior flexibilidade para seus usuários por prover funcionalidades de programação para transações financeiras, suplementando assim a simples operação de transferência de valores disponível no Bitcoin.

De forma similar à maioria dos outros sistemas de criptomoedas, a função principal do Ethereum é manter um livro-razão público e global, conhecido como *blockchain*. A *blockchain* contém todas as transações entre participantes. Os remetentes de uma transação confirmam sua participação através de uma chave de criptografia assimétrica. Então, essa transação deve ser adicionada a um bloco, o qual vai ser futuramente adicionada à cadeia de blocos do sistema, de onde vem o nome *blockchain*. Os sistemas de criptomoedas normalmente são organizados como uma rede par-a-par (P2P) [Hao et al. 2020], onde usuários especiais conhecidos como mineradores executam ações com custo computacional para confirmar a transação e então adicioná-las à *blockchain* [Ren et al. 2020], sendo estes mineradores recompensados pelo seu esforço, de acordo com a política da plataforma *blockchain* em questão. O custo computacional para processar transações é um mecanismo de segurança em *blockchain*, denominado *prova de trabalho* (PoW) e intencionalmente proposto para evitar a participação de mineradores desonestos [Berg, Davidson e Potts 2020]. Este processo recompensa os primeiros mineradores que completarem o desafio com um valor fixo por bloco juntamente com as taxas de mineração fornecidas pelos usuários em suas transações, sendo ambos valores definidos em Ether, a unidade monetária do Ethereum. A recompensa remunera o esforço do minerador e os custos operacionais respectivos, sendo crucial para a maioria dos sistemas que usam a *blockchain* como estrutura. Isto se deve ao fato de que mineradores compartilham seus recursos computacionais com a rede, o que é essencial para a natureza descentralizadas das criptomoedas [Eyal e Sirer 2018].

[1] https://coinmarketcap.com/currencies/ethereum/ (visitado em 08/06/2020)
[2] https://etherscan.io/ (visitado em 08/06/2020)

1.1 PROBLEMA

Um usuário que tem o interesse em realizar uma transação ou contrato inteligente no Ethereum é obrigado a pagar uma taxa para um minerador. Atualmente, o valor da taxa não é predefinido no Ethereum e funciona como um mecanismo de leilão simples, onde usuários oferecem uma quantia que eles desejam para ter sua transação inserida na *blockchain*. Pode-se esperar que mineradores selecionem as maiores taxas entre todas disponíveis, como em uma abordagem gulosa. No entanto, usuários têm reportado dificuldade em estimar as taxas das transações [Antonio Pierro et al. 2020]. Como consequência dos usuários não oferecerem uma taxa adequada, eles podem pagar uma taxa com valores altos e mesmo assim experienciar tempos de espera longos [Buterin 2017], o qual definimos aqui como o tempo entre a primeira vez que a transação é observada e o momento que a transação é devidamente inserida na *blockchain*. Por questões de simplicidade, este período será referido como somente tempo de espera.

Observa-se que o tempo de espera é o aspecto principal para várias aplicações que são sensíveis ao atraso. Se existe uma correlação entre as taxas e o tempo de espera, seria possível prever um valor de taxa apropriado o qual o usuário deve fornecer como forma da sua transação ser validada em um certo limite de tempo. Seria possível também prever se uma transação seria minerada, dado um período. Neste caso, por exemplo, quando um usuário usa o Ethereum para pagar sua conta no restaurante, o gerente do restaurante validaria a transação em uma quantidade de tempo menor, ao invés de fazer o cliente esperar por um longo período.

Alguns trabalhos investigam as dinâmicas das taxas e o tempo de espera neste ambiente [Pierro e Rocha 2019, Antonio Pierro et al. 2020], porém a maioria das investigações foram conduzidas no Bitcoin [Houy 2014, Easley, O'Hara e Basu 2019, Möser e Böhme 2015, Ricci et al. 2019, Li, Ma e Chang 2018]. Por outro lado, as pesquisas existentes têm se focado na segurança [Xu et al. 2020, Bartoletti et al. 2020], performance da plataforma [Chan e Olmsted 2017, Chen et al. 2017] e análise através da perspectiva de grafos sobre o comportamento do usuário [Motamed e Bahrak 2019, Chen et al. 2018]. Nenhum desses trabalhos prévios focou nas taxas pagas pelos usuários e o tempo de espera encontrado no Ethereum. Além disso, estes trabalhos não correlacionam o tempo de espera com atributos das transações associados a taxas como o *gas*, *gas price* e o valor de mercado do Ether. Sem um devido conhecimento da rede do Ethereum e seu comportamento, não é possível resolver, por exemplo, questões de segurança em um dos serviços mais importante da Internet do Futuro [Zheng et al. 2017, Zheng et al. 2018], a *blockchain*.

Percebe-se também que, apesar da falta de estudos sobre o tempo de espera no Ethereum, há um senso comum que para sistemas de *blockchain* baseados no PoW, usuários que pagam taxas maiores terão suas transações mineradas em um período menor. Esse

senso comum é suportado pela mídia[3] ou então não é devidamente evidenciado [Wood et al. 2014]. No entanto, ser um senso comum e suportado pela mídia não é uma argumentação válida para o meio científico. Em sistemas reais, como o Ethereum, usuários podem fornecer taxas altas e experienciar tempos de espera altos.

Além disso, o crescimento do interesse no Ethereum, como previamente mencionado, leva também a preocupações da segurança da rede. Dentre as diversas possibilidades de ataques e brechas da plataforma, destacamos os ataques de negação de serviço (*Distributed Denial of Service* - DDoS) [Chen et al. 2017, Li et al. 2017]. Neste tipo de ataque várias transações maliciosas são efetuadas em um curto período de tempo, mantendo o sistema ocupado, de modo que não possa atender totalmente as transações reais.

A plataforma Ethereum continua susceptível a esse tipo de ataque pois as soluções utilizadas para mitigá-los consistem em um aumento de taxas sobre algumas operações o que pode punir os usuários honestos, ao passo que ainda existem várias outras operações de baixo custo que podem ser exploradas por usuários maliciosos [Chen et al. 2017].

Mesmo que as transações financeiras sejam realizadas mediante o pagamento de uma taxa, como ocorre na maioria das criptomoedas, usuários maliciosos têm explorado esse esquema de tarifação para aplicar ataques DDoS. Por exemplo, a plataforma Ethereum, em especial, já teve de se adequar a ataques que exploravam o seu mecanismo de tarifação para tornar todas as transações da rede mais lentas, ocasionando desperdício de recursos computacionais [Gautham 2016, Buterin 2016].

Em suma, nessa dissertação são tratados dois problemas que podem impactar o tempo de confirmação de transações no Ethereum: primeiro, é avaliada a correlação entre atributos relacionados as taxas fornecidas pelos usuários para minerar suas transações e o tempo de espera das mesmas e, segundo, é avaliado o ataque conhecido como *Penny Attack* e são propostas contra-medidas, tais como o uso de técnicas de aprendizado de máquina para a identificação precoce deste tipo de ataque.

1.2 OBJETIVOS

Este trabalho tem por objetivo principal investigar o comportamento da plataforma Ethereum, focando principalmente em dois aspectos: (i) avaliar o tempo de espera de transações e *smart contracts* dados diferentes comportamentos de usuários e taxas fornecidas para mineração das mesmas e (ii) avaliar o tempo de espera de transações e *smart contracts* em momentos de *Penny Attack*.

[3] https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

1.3 CONTRIBUIÇÕES

As pesquisas e análises conduzidas neste trabalho acarretaram nas seguintes contribuições:

- a metodologia de coleta de dados;
- o conhecimento sobre a relação entre atributos concernentes à taxa paga ao minerador e o tempo de espera de uma transação no Ethereum;
- o entendimento do comportamento de transações e *smart contracts* no Ethereum;
- uma arquitetura para reproduzir a rede Ethereum em ambiente controlado para avaliações;
- técnicas de aprendizado de máquina, incluindo comitês de classificadores para detectar *Penny Attack* na rede;
- a publicação de dois estudos sobre a correlação entre taxas e o tempo de espera [Sousa et al. 2019, Sousa et al. 2020a];
- duas publicações de utilização de técnicas de aprendizado de máquina para reconhecer o *Penny Attack* [Sousa et al. 2020b, Sousa et al. 2021];
- e uma publicação sobre a utilização de técnicas de aprendizado de máquina para prever a aprovação ou não de uma transação no Ethereum [Oliveira et al. 2021].

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação está organizada da seguinte maneira: o Capítulo 2 discute os conceitos relacionados ao Ethereum e sua estrutura, a *blockchain*; os trabalhos relacionados e comparações com esta dissertação estão presentes no Capítulo 3. No Capítulo 4 é descrita a metodologia de coleta dos dados utilizados nesta dissertação. No Capítulo 5 são externados os estudos conduzidos para avaliar o tempo de espera na rede do Ethereum, envolvendo metodologia e resultados. No Capítulo 6 são apresentados metodologia de uma simulação de um *Penny Attack* em uma rede privada Ethereum, juntamente com o uso de aprendizado de máquina para seu reconhecimento. Finalmente, no Capítulo 7 encontram-se as considerações finais e previsões de trabalhos futuros.

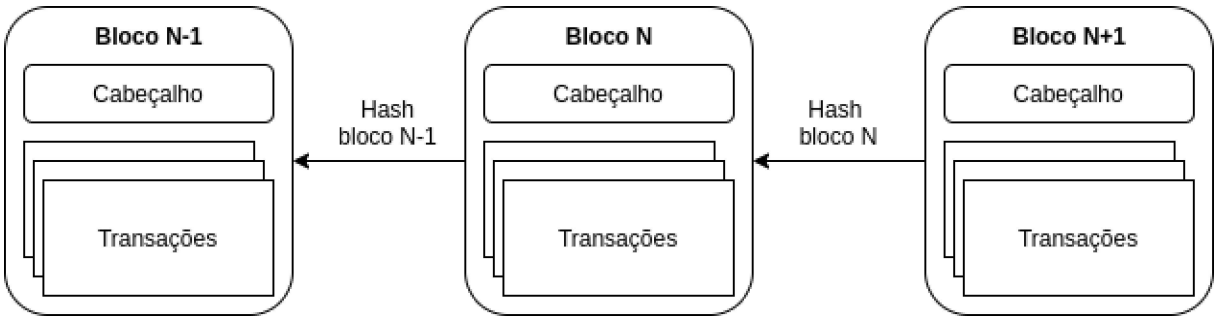
2 *BLOCKCHAIN* E ETHEREUM

Neste capítulo são abordados os principais conceitos sobre *blockchain* e Ethereum relevantes para a compreensão dos trabalhos, experimentos e análises desta dissertação.

2.1 *BLOCKCHAIN*

Blockchain é uma tecnologia [Nakamoto 2008] que possibilitou a realização de transações financeiras por uma via alternativa às instituições bancárias e de crédito tradicionais. Quase todo comércio virtual é feito através de instituições financeiras, as quais servem como uma plataforma confiável. Atualmente, a maioria as transações no mundo moderno ainda são feitas via essas instituições, que seguem uma abordagem centralizada, ou seja, a execução e o registro da transação é monopolizada por uma instituição, e emissores e destinatários da transação devem confiar plenamente nessa instituição. Essa abordagem, apesar de funcional para a maioria das pessoas, tem alguns problemas tais como o alto custo de transações: as instituições em questão tem grandes estruturas físicas, empregados e vários outros custos indiretos, os quais são repassados indiretamente aos clientes finais. Além disso, os métodos de transação monetária tradicionais não são totalmente transparentes e rastreáveis. Por exemplo, os clientes de bancos e financeiras não tem clareza sobre o que essas instituições fazem com os valores que lhes são confiados para depósitos e demais transações financeiras, fazendo com que a *blockchain* se destaque pela maior confiança devido a transparência oferecida e de governância descentralizada.

Figura 1 – Estrutura da *blockchain*.



Fonte: Elaborado pelo autor. (2020).

A *blockchain* é uma lista de blocos encadeados através de criptografia [Denisova, Mikhaylov e Lopatin 2019]. Especificamente, cada bloco contem um apontador para o bloco anterior, que é um *hash* criptográfico identificando de forma única o bloco anterior. A Figura 1 ilustra a estrutura de um bloco e seus encadeamentos. O bloco é estruturado em um cabeçalho e um conjunto de transações com tamanho pré-definido ou ilimitado,

a depender da aplicação da blockchain. Por sua vez, o cabeçalho contem o apontador *hash*, o número do bloco e demais campos que resumem o conjunto de transações contidas no bloco (meta-dados). Para construir o apontador *hash* aplica-se uma função *hash*, por exemplo o algoritmo SHA [Paar e Pelzl 2010], sobre o cabeçalho do bloco anterior apenas, reduzindo assim custos computacionais para construção e verificação da *blockchain*.

As transações que compõe o bloco, por sua vez, são interações emitidas por usuários da *blockchain*. Nelas são contidas informações como usuário remetente e destinatário e a informação que está sendo enviada. Desta forma uma transação consiste em uma instrução assinada digitalmente que é enviada dentro do sistema [Wood et al. 2014]. As transações são divulgadas pela rede e são validadas antes de serem incluídas nos blocos, os quais também são validados. A partir do momento em que uma transação é inserida na *blockchain* esta não pode ser removida, promovendo o conceito de irretratabilidade.

No caso de sistemas distribuídos, como a *blockchain*, essa estrutura é replicada em vários computadores por meio de um protocolo de consenso. Cada computador é conectado via uma rede ponto-a-ponto sem hierarquia ou nós especiais funcionando através do protocolo TCP e uma topologia aleatória [Narayanan et al. 2016]. O acesso ao sistema pode funcionar de duas formas: (i) aberto e qualquer pessoa ou organização pode aderir à rede ou (ii) privado, necessitando de um convite para participar, se tornando portanto um participante do sistema e adquirindo direitos de processar novos blocos de transações a serem inseridos na *blockchain*, desde que siga o protocolo de consenso do sistema.

Dessa forma, o funcionamento de sistemas baseados em *blockchains* têm uma estrutura descentralizada. Nessa estrutura os participantes oferecem seus recursos computacionais para o sistema e nenhum participante individual tem o controle sobre o sistema. Há um cenário específico onde um grupo de participantes pode controlar a rede. Para que isso ocorra, participantes precisam formar um conluio e possuir mais de 50% de representação de poder computacional da rede. Esse tipo de ameaça ao sistema se torna mais improvável a medida que a rede aumenta o número de participantes [Meneghetti, Sala e Taufer 2020, Gramoli 2017].

Outro aspecto importante sobre *blockchains* é que esta pode se apresentar como permissionada ou não permissionada. Em redes permissionadas, também conhecidas como privadas, para visualizar e realizar transações, além de executar o processo de mineração, é necessário obter a permissão da entidade que governa esta *blockchain*, fazendo com que a rede seja composta necessariamente por partes as quais existem uma relação de confiança. Em redes não permissionadas, ou seja, redes públicas, qualquer pessoa pode fazer transações e consultar o histórico de transações na rede sem pedir autorizações, e por isto, há necessidade de verificar as ações submetidas pelos diversos usuários que participam desse ecossistema, surgindo assim os protocolos de consenso. No escopo desta dissertação é tratado somente sobre *blockchains* não permissionadas.

2.2 PROTOCOLOS DE CONSENSO

A característica decentralizada das criptomoedas faz com que haja necessidade da existência de um protocolo que permita que todos os nós trabalhem com réplicas da *blockchain* no mesmo estado, mantendo suas características e conteúdos concisos entre os participantes do sistema, como um acordo replicado [Raynal 2010, Schneider 1990]. Esse é um problema que os mecanismos de consenso se propõe a enfrentar, visto que os sistemas podem ter participantes não confiáveis pois acesso é aberto [Wang et al. 2019]. Os protocolos de consenso devem ser capazes de oferecer um acordo entre o estado dos dados globais da blockchain [Bano et al. 2017], mesmo que entre um grande números de usuários os quais não se possui confiança, proporcionando baixa sobrecarga de comunicação.

Da perspectiva da arquitetura de sistemas distribuídos, os mecanismos de consenso devem prover um acordo tolerante a falhas bizantinas [Vukolić 2015, Schneider 1990] nas redes *blockchain*. Mais especificamente, os participantes do sistema devem entrar em um acordo para manter suas réplicas da blockchain em um mesmo estado, até mesmo em situações de possíveis conflitos de informações e falhas arbitrárias, ou seja, falhas bizantinas, em alguns participantes. A seguir, serão descritos os principais protocolos de consenso utilizados por sistemas baseados em *blockchain*.

2.2.1 Proof of Work

Atualmente o mecanismo mais utilizado em *blockchains* é a Prova de Trabalho [Zoican et al. 2018], em inglês *Proof of Work* (PoW), que tem seu surgimento antes mesmo da existência de *blockchains*. A ideia de prova de trabalho assim como utilizado em *blockchain* já havia sido descrita em trabalhos da década de 1990 [Dwork e Naor 1992, Jakobsson e Juels 1999] no contexto de evitar envio de lixo eletrônico pelo uso deste protocolo.

A Prova de Trabalho é um protocolo no qual alguém pode provar efetivamente para outrem que um esforço computacional significativo foi empregado para o processamento de alguma tarefa. No contexto de *blockchain*, especificamente criptomoedas, prova de trabalho é estabelecida na forma de desafios criptográficos, neste contexto [Kiayias, Miller e Zindros 2020]. Apesar do protocolo demandar um esforço que impacta no tempo para gerar a resposta correta, esse esforço pode ser facilmente verificado. Por exemplo, a prova de trabalho para a maioria das criptomoedas requer a geração de um *hash* para um bloco com uma quantidade de zeros específica no início, variando um campo no cabeçalho do bloco denominado *nonce* com valores aleatórios. Isso consiste em testar sucessivamente diferentes valores de *nonce* até alcançar o formato do *hash* desejado, isto é, um algoritmo de força bruta. Logo, encontrar o *hash* correto requer trabalho computacional intensivo, porém, dado um bloco e o número correto, a função geradora de *hash* sobre esses é o suficiente para verificar se o *hash* apresentado está correto, ou seja, ele está de acordo ao formato estabelecido para a prova de trabalho.

Em criptomoedas, tipicamente, cada bloco segue uma dificuldade de processamento predeterminada, definida para que o tempo entre os blocos inseridos na *blockchain* seja próximo a uma constante desejada. Desse modo, os participantes da rede realizam rodadas de prova de trabalho, por bloco, para eleger quem criará o próximo bloco a ser replicado na rede. O participante eleito é aquele que primeiro apresenta o *hash* satisfazendo a prova de trabalho. As rodadas de prova de trabalho em criptomoedas também são conhecidos por *mineração*, porque o participante (*minerador*) eleito em cada rodada tem o direito de criar um valor em criptomoedas, definido no protocolo, e possuí-lo como recompensa pelo esforço computacional.

2.2.2 Proof of Authority

A prova de autoridade, ou *Proof of Authority* (PoA), é um método de consenso simplista, baseado na reputação dos participantes da rede [Angelis et al. 2018]. Consequentemente, blocos são processados e replicados na blockchain por contas com reputação suficiente para participar do processo de mineração, divergindo do PoW, que foca no poder computacional dos participantes. PoA é amplamente usado quando a recompensa para minerar não é financeiramente o suficiente para os mineradores da rede, sendo assim, sua adoção requer menos investimentos dos mineradores com o aumento do computacional.

Portanto, esse tipo de protocolo de consenso não demanda alto desempenho computacional como o PoW, visto que somente um participante da rede, o validador, executará os cálculos necessários para o PoA. Outro ponto fundamental do PoA é que o participante ganha o privilégio de minerador eleito para criar o próximo bloco de acordo com as políticas estabelecidas na rede, o que varia entre plataformas. Em plataformas as quais existe mais de um validador, o PoA não permite que o minerador execute o processo de mineração em blocos consecutivos, exigindo um processo de rodízio entre as partes da rede capazes de executar este processo.

2.2.3 Clique PoA

A Clique PoA[1], também conhecida como Clique, é um tipo de PoA específico da criptomoeda Ethereum [Szilágyi 2017], tendo as seguintes características particulares para um sistema de N mineradores chamados de *sealers*:

- cada bloco tem um *sealer* candidato;
- se o candidato não minera o bloco, a oportunidade é passada para o próximo candidato;
- a escolha é feita através de um algoritmo Round-robin;

[1] https://eips.ethereum.org/EIPS/eip-225

- após sua participação, o *sealer* deve esperar $N/2$ rodadas para minerar um bloco novamente;

Medidas adotadas pela Clique, tais como a necessidade de esperar $N/2$ rodadas para minerar um bloco novamente, referem-se unicamente à segurança da rede. O último tópico garante uma contingência de danos feitos por nós maliciosos. Uma vez que um participante malicioso pode comportar-se de maneira idônea por um período para ganhar confiança da rede, garantindo seu papel como *sealer*, a exigência de um rodizio entre os mineradores desta *blockchain* diminui os danos causados por esse participante.

2.3 ETHEREUM

Ethereum é a segunda maior criptomoeda atualmente em valor de mercado e volume de transações [Gencer et al. 2018], atrás apenas do Bitcoin, que é a criptomoeda pioneira. Ethereum difere-se do Bitcoin principalmente pela capacidade de processar contratos inteligentes, em inglês *smart contracts*, e as ferramentas para programação desses contratos. A linguagem adotada para programação de contratos, a *Solidity*, é Turing-completa, sendo assim, suporta uma vasta gama de instruções computacionais [Buterin et al. 2013], além de poder resolver todos os problemas computáveis [Kepser 2004]. Dentre as capacidades destes contratos, estão[2]:

- gerenciamento de acordo entre usuários, emissores e destinatários de transações financeiras, por exemplo, quando um usuário faz uma compra de um seguro por um terceiro;
- armazenar informação de uma aplicação, como por exemplo o registro de um domínio ou dados de um usuário;
- ser base para outros contratos e bibliotecas de softwares.

Esta plataforma funciona como qualquer outra criptomoeda baseada em *blockchain*, porém oferece mais recursos aos seus usuários devido ao uso de contratos inteligentes. As contas no Ethereum podem ser classificadas em dois tipos de usuários: mineradores e usuários comuns. Um usuário minerador busca ganho financeiro pelo processo de mineração do bloco, e recebe as devidas recompensas do sistema devido ao esforço computacional empregado no bloco. Usuários comuns, no entanto, usam a plataforma do Ethereum para executar contratos ou transferir valores entre contas.

Devido ao objetivo mais básico da plataforma, que é a troca de valores, é necessário a utilização de uma carteira virtual, na qual o usuário pode guardar suas finanças,

[2] https://www.coindesk.com/information/ethereum-smart-contracts-work

e é identificada através de um endereço público único. Esta carteira é um software (MetaMask[3], MyCrypto[4], TrustWallet[5], entre outros) que permite ao utilizador armazenar fundos, realizar transações e verificar saldo [Tavares et al. 2018].

Para executar o Ethereum é obrigatório o uso de um cliente, tal como o go-ethereum, Parity, cpp-ethereum, pyethapp, ethereumjs-lib, Ethereum(J), ruby-ethereum e ethereumH. O cliente funciona sob uma máquina virtual, a Ethereum Virtual Machine (EVM), e dado as várias opções de configurações dos softwares clientes desta plataforma, algumas variáveis podem divergir de cliente para cliente, tais como o tamanho da fila de transações e o tempo que uma transação pode permanecer na fila.

2.4 TRANSAÇÕES NO ETHEREUM

A interação básica de um usuário do Ethereum se dá através de transação. Esta se bifurca nas classes: (i) transações financeiras, o tipo mais simples e (ii) contratos inteligentes, que por sua vez, podem gerar outros dois tipos de transações: criação de contratos e invocação de contratos. A principal diferença entre transações financeiras e contratos inteligentes é que a transação executa somente uma troca de valores entre usuários, e os contratos inteligentes envolvem a execução do código predefinido, podendo ter diferentes finalidades, de acordo com seu conteúdo, como será discutido nessa seção.

Acerca dos *smart contracts*, o processo de criação, onde o usuário define o código deste contrato, o mesmo é implantando na rede, quando é conferido a ele um novo endereço. O outro tipo de interação, a invocação do contrato que é realizada quando o destinatário de uma transação é um *smart contract* e podem ser passadas informações para o mesmo através de um campo de dados na transação. Ao serem executados, ambos os tipos de transações entram na fila de espera, a qual chamamos de *mempool*, permanecendo ali até que um minerador decida minerá-la, consequentemente sendo registrada na *blockchain*.

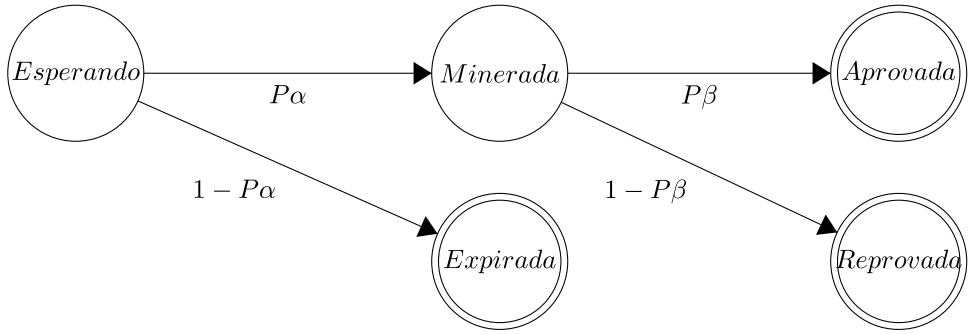
Cada transação executada por um minerador no Ethereum requer o pagamento de uma taxa para minerá-la, que é definida por dois valores: *gas* e *gas price*, e chamaremos de "taxa" para simplificar sua referência. O primeiro valor especifica a quantidade de esforço computacional máximo permitido pelo remetente para executar a transação, enquanto o segundo define o quanto quem emite a transação irá pagar por unidade de processamento utilizada. O produto desses dois valores é o quanto o usuário emitente pretende gastar com a execução da transação. Atualmente, o *gas price* não tem um valor fixo e o Ethereum adota um mecanismo de leilão simples, onde o emitente define o valor pretendido. A transação é executada na EVM até que o processamento seja concluído ou que o *gas* oferecido acabe. Dado esse mecanismo de leilão simples no Ethereum, usuários podem

[3] https://metamask.io/
[4] https://mycrypto.com/
[5] https://trustwallet.com/

oferecer valores acima da média de *gas price* na expectativa de mineradores incluírem sua transação de forma ágil na *blockchain*, ou seja, dar preferência para incluir nos próximos blocos de transações a serem processadas aquelas que oferecem maior valor de *gas price*.

Mostrados os processos de criação de blocos (Seção 2.1) e o mecanismo de taxas por transação, discute-se agora os estados de uma transação no Ethereum. A Figura 2 ilustra os estados que compõem o ciclo de vida das transações do Ethereum. O ciclo de vida de uma transação inicia com o estado "Esperando" quando ela é observada pela primeira vez na *mempool*. Um usuário oferece a taxa para que o minerador processe sua transação e espera que algum minerador a inclua para ser processada em um bloco, entrando assim no estado "Minerada".

Figura 2 – Ciclo de vida de uma transação. Pα é a probabilidade de uma transação na *mempool* ser minerada e Pβ é a probabildiade de uma transação minerada ser aprovada.



Fonte: Elaborado pelo autor. (2020).

A aprovação de transações depende da quantidade de *gas* oferecido pelo usuário ser o suficiente para executar as operações propostas pelo usuário emissor. Quando o usuário oferece uma quantidade suficiente de *gas*, mineradores podem processar a transação e, como consequência, o usuário terá sua transação aprovada. Fazendo uma analogia de *gas* com gasolina, esta situação torna-se semelhante ao funcionamento de um carro, onde quantidades suficientes de *gas* permitem ao carro alcançar seu destino final com sucesso. Caso contrário, o combustível (*gas*) finaliza e o carro pára no caminho sem obter êxito. Retornando ao processamento de transações no Ethereum, se um minerador usa todo o *gas* e ainda não foi possível processar a transação, esta não tem efeito na rede e seu status é definido como reprovado.

Algumas transações não são mineradas ao longo do tempo. De fato, podemos observar algumas transações na *mempool* e, mesmo depois de um longo tempo depois da primeira observação, não foi possível decidir se a transação é considerada como aprovada ou reprovada. Quando tal situação ocorre, a transação é considerada como expirada. Existem outros casos em que transações podem expirar, como a seguinte: o *nonce* da

transação[6], neste caso, refere-se ao contador de transações de um usuário com o sistema. Por exemplo, a primeira transação de um usuário com a rede terá *nonce* 1, a segunda terá o valor 2. Logo, uma transação sai do estado "Esperando" para o estado "Expirada", caso outra transação desse mesmo usuário com *nonce* maior seja aprovada no sistema.

Alguns autores consideram também a existência de um estado entre "Esperando" e "Expirada" [Weber et al. 2017], remetendo a um estado de ignoração por parte dos mineradores da rede, que deliberadamente removem a transação da *mempool*, havendo ainda a possibilidade de retorno à *mempool*. Uma vez que existem usuários com *mempool* infinita, ou seja, não eliminam nenhuma transação de sua fila de transações pendentes, foi optado por não utilizar este estado, visto que uma transação pode não seguir o ciclo definido por este autor, enquanto a Figura 2 aplica-se a todos os casos possíveis de transações.

Existem outros campos nas transações do Ethereum tais como o *hash*, uma forma única de identificar a transação, aplicando uma função de *hash* SHA-3 sobre o conteúdo da transação [Xu et al. 2020], obtendo um valor alfanumérico. Na mudança de estado de "Esperando" para "Minerada" a transação, que agora é contida em um bloco, possui uma marcação de tempo, idêntica ao bloco ao qual ela pertence, remetendo ao momento ao qual esta informação passou a fazer parte da *blockchain*. Além disso, uma transação pode conter um conteúdo, que é anexado ao seu corpo, podendo ser uma mensagem, ou o conteúdo para utilização por um contrato inteligente.

Em suma, as transações e *smart contracts* contém vários atributos, podendo destacar entre eles: *hash*, número do bloco ao qual pertencem, um remetente, um destinatário, *gas*, *gas price*, carimbo do tempo herdado do bloco a qual ela pertence, *nonce*, valor e conteúdo. O hash é um campo de texto com tamanho fixo, o qual é usado para dar um endereço único a uma transação, sendo seu meio de identificação. O *gas* é a quantidade de esforço computacional necessário para executar e armazenar a transação. O *gas price* refere-se a quantidade paga por unidade de processamento (*gas*), e tem sua unidade medida em Ether. Os remetentes e destinatários são endereços únicos na rede, ou carteiras, e referem-se a usuários ou *smart contracts*. O carimbo do tempo define o momento em que a transação foi confirmada na *blockchain*, ou seja, o momento exato no qual o bloco que ela pertence foi minerado com sucesso. O valor contido na transação refere-se a uma quantia em Ether que será enviada do endereço destinatário para o remetente. O campo de conteúdo pode conter qualquer tipo de informação, desde simples mensagens, ou códigos para criação e execução de *smart contracts*.

[6] Não confundir com *nonce* do bloco.

3 TRABALHOS RELACIONADOS

Existem vários trabalhos os quais focam seus esforços na análise e caracterização da taxa na criptomoeda com maior valor de mercado: o Bitcoin. Nos primeiros anos do Bitcoin, a recompensa dos mineradores foi obtida com os blocos de transações minerados, sendo esta recompensa o número de *bitcoins* (BTC) (unidade monetária do Bitcoin). Sendo assim, no início da rede, o primeiro usuário a resolver o desafio criptográfico pelo PoW, era recompensando com 50 BTC [Houy 2014]. No entanto, a taxa paga por processamento de transação tornou-se gradualmente a recompensa mais importante dos mineradores devido a dificuldade crescente de prova de trabalho originalmente proposta para a segurança do protocolo Bitcoin, juntamente com aumento no volume de transações, dada a popularidade do Bitcoin, o qual aumenta a oferta de gorjetas a mineradores para incluírem suas transações. Alguns autores [Houy 2014] conduziram um estudo teórico sobre a economia das taxas pagas por processamento de transação do Bitcoin e propuseram uma taxa fixa imposta como uma medida para manter o blockchain do Bitcoin seguro e estável. Outros autores [Easley, O'Hara e Basu 2019] investigaram o impacto das taxas de transação na participação dos usuários no Bitcoin por meio de uma abordagem de teoria dos jogos e concluíram que a ausência de taxas acabaria por tornar o Bitcoin financeiramente inviável para mineradores.

Diferentemente dos estudos mencionados acima baseados em abordagens teóricas, Möser e Böhme conduziram o estudo empírico pioneiro sobre taxas pagas por processamento de transação no Bitcoin [Möser e Böhme 2015]. Seu estudo foi baseado em uma amostra de 9.000 transações das 45,7 milhões de transações registradas no blockchain público do Bitcoin até o final de agosto de 2014. Eles mostram evidências de que taxas para processamento de transações mais altas levam a uma confirmação mais rápida, levando em consideração, principalmente, um longo período de tempo para confirmar as transações com taxa zero. Os autores, no entanto, não mediram a relação entre esse tempo e as taxas por meio de correlações, mas através da comparação do valor absoluto do tempo de espera de diferentes transações, agrupadas por taxa de processamento. Além disso, o número de transações utilizado para esta conclusão é pequeno perto do volume total de transações da rede, sendo utilizadas 9143 transações somente.

O tempo de espera atraiu a atenção da comunidade financeira desde que o Bitcoin se tornou uma moeda virtual valiosa [Böhme et al. 2015]. Modelos de fila foram adotados pelos pesquisadores para prever esse tempo pendente. Por exemplo, o modelo de filas proposto por Ricci et al. [Ricci et al. 2019] faz uso de classificadores supervisionados os quais incluem atributos e características de transações, enquanto o modelo proposto em outro estudo [Li, Ma e Chang 2018] envolveu métodos geométricos matriciais. Em outro trabalho um modelo para prever se uma transação será incluída no próximo bloco Bitcoin, ou seja, um tempo de espera inferior a dez segundos, foi proposto [Tedeschi et al.

2019] com base no histórico de transações anteriores e técnicas de *deep learning*. O acesso a possível correlação entre o tempo pendente e os recursos relacionados a taxas é uma etapa importante para aprimorar os métodos mencionados, pois mais recursos correlatos melhoram o desempenho das previsões de tempo de espera.

A análise do tempo de espera no Ethereum é de suma importância dado a popularidade da plataforma e a vasta gama de aplicações que usam os *smart contracts* como base. Dois outros estudos [Pierro e Rocha 2019, Antonio Pierro et al. 2020] investigaram os efeitos de causalidade por fatores externos e a precisão da recomendação de taxa dada por *EthGasStation oracle*[1]. Este serviço recomenda aos usuários o *gas price* que eles devem pagar para aguardar um certo tempo de espera. Os autores observaram que a variação no volume de transações pendentes na *mempool* e o número de mineradores influenciam a recomendação de taxas desse serviço, além de uma alta margem de erro do valor de *gas price* que chega a 28%. Os autores concluíram que as recomendações existentes do *gas price* para os usuários não são precisas. Essa afirmação está alinhada com os resultados deste trabalho, e demonstram a necessidade de análises da importância de atributos de transações do Ethereum no tempo de espera.

Vale ressaltar que a crescente popularidade do Ethereum atraiu a atenção de pesquisadores interessados em entender a dinâmica desse novo ecossistema blockchain. No entanto, a grande maioria dos estudos baseados em grandes conjuntos de dados Ethereum concentra-se nas tendências de preços do Ether [Lamon, Nielsen e Redondo 2017, Hitam e Ismail 2018, Chen, Narwal e Schultz 2019, Kumar e Rath 2020], e baseiam-se em recursos do mercado externo, como séries temporais do preço do Ether em dólares ou até mesmo comentários em mídias sociais, para prever o valor do Ether. Nesses estudos, os recursos das transações do Ethereum que emergem do comportamento do usuário não são explorados como é feito nesta dissertação.

Payette et al. propuseram uma categorização de usuários do Ethereum [Payette, Schwager e Murphy 2017], observando o comportamento de suas transações a partir de endereços anônimos. Para isso foram obtidas informações sobre o Ethereum tais como detalhes de transações e usuários através da API do Etherscan[2], coletando mais de 250.000 endereços únicos, com seus respectivos saldos e históricos de transação. A partir dos dados, foram propostos 4 agrupamentos diferentes, dado as características de cada usuário, utilizando K-Means como forma de agrupamento.

Existem trabalhos os quais usaram técnicas complexas de rede para analisar a centralidade dos participantes na rede do Ethereum [Mascarenhas, Vieira e Ziviani 2018]. Transações financeiras foram caracterizadas [Chen et al. 2018], considerando a relação entre usuário e usuário, usuário com contrato inteligente e contrato inteligente com contrato

[1] https://ethgasstation.info/
[2] https://etherscan.io/

inteligente usando análise de gráficos de fluxo múltiplo. Em [Motamed e Bahrak 2019], as transações no Ethereum e em outras quatro criptomoedas foram analisadas como um gráfico em evolução que permite estimar a densidade do gráfico de transações em função do preço da criptomoeda e classificar os padrões mais comuns de transações com base em no grau dos usuários.

[Weber et al. 2017] estuda este tema por um espectro semelhante, analisando o tempo de confirmação de uma transação, ou seja, o tempo que uma transação leva para ter uma quantidade suficiente de blocos para poder descrever a transação como confirmada na rede, o que pode variar de plataforma para plataforma, segundo o autor. No trabalho em questão são definidos os valores de 6 blocos para o Bitcoin, visto que o seu tempo entre blocos é maior do que no Ethereum [Kiffer, Levin e Mislove 2017, Decker e Wattenhofer 2013], e de 12 blocos para a plataforma foco deste estudo como valores aceitáveis para confirmação. Apesar disso, são analisados diferentes quantidades de blocos confirmando a transação. Posteriormente são analisadas também diferentes faixas de *gas price*, tal como a subseção 5.2.1 desta dissertação.

Pode-se observar que existe uma vasta gama de estudos relacionados ao Ethereum em diferentes aspectos. Apesar das copiosas pesquisas envolvendo as diversas características e atributos da plataforma do Ethereum, não há estudos os quais focam na análise empírica sobre a correlação do tempo de espera e atributos relacionados a taxa das transações e *smart contracts* no Ethereum, sendo esta correlação tomada como verdadeira nos estudos sem uma análise científica. Este fato abre precedentes para pesquisas neste tema, sendo o foco desta dissertação.

Outro tema que concerne o tempo de espera no Ethereum é a segurança em sistemas que usam *blockchain*. A lentidão da rede causada por um grande volume de transações é estudada desde 2017, quando Gerard et. al. abordou a inundação do Bitcoin por transações maliciosas [Gerard 2017]. Soluções para ataques desse tipo compreendem desde medidas simplistas, como aumento do tamanho do bloco, até o estabelecimento de *blockchains* paralelas, que mantém a rede principal somente com o resumo do fluxo de transações. Por exemplo, alguns autores [Chen et al. 2017] propõe uma mudança no mecanismo de custo e armazenamento computacional da rede para ajustar ataques desse tipo, com maior foco em contratos inteligentes. Nesse trabalho, são discutidas modalidades de ataque que exploram operações de baixo custo dos *smart contracts* para gerar lentidão no processamento e, consequentemente, reduzir o tamanho dos blocos, afetando a vazão da rede. Para sanar este problema, os autores então avaliam medidas que amenizam o efeito baseadas no ajuste de tarefas podem minimizar os efeitos dos ataques.

Existem trabalhos os quais estudam ataques distribuídos de negação de serviço no Bitcoin [Saad et al. 2019], a criptomoeda com o maior número de usuários e o maior valor de mercado. Os autores identificam as contas que participaram do DDoS ocorrido

em 11 de novembro de 2017, além de destacar a metodologia usada pelos atacantes para materializar os danos à rede. A forma como o DDoS à blockchain é definido no trabalho parte do mesmo princípio do *Penny Attack*: transações são estabelecidas como forma de afetar a rede destino do ataque, envolvendo um curto período e um volume maior do que o convencional da plataforma em questão. Nesse capítulo, também são propostas contramedidas no nível de fila de transações, avaliadas através de simulações de eventos discretos. Essas soluções priorizam as transações em relação às taxas de recompensa ao minerador na primeira heurística (as mais altas são prioritárias), o que remove o poder dos mineradores de escolher qual método de mineração eles desejam. A segunda heurística é um adendo à primeira, buscando a existência de transações pais das atuais (reinserção de transações na fila). Sendo assim, transações mais antigas tem preferência na fila.

Uma outra linha de trabalho agrupa as transações ocorridas no Bitcoin e, a partir da definição dos grupos, tentam identificar aquelas que participaram de ataques maliciosos a partir de suas respectivas características. Por exemplo, foi utilizado o K-Means, para identificar transações maliciosas realizadas na rede do Bitcoin [Baqer et al. 2016]. Os autores também estimam o valor monetário em dólares que o ataque custou, com base na recompensa aos mineradores das transações no momento da execução e no volume de transações necessárias para estabelecer o ataque.

Ataques de negação de serviço, tal qual o *Penny Attack*, não são exclusivos da rede Ethereum. Por exemplo, Chervinski et al. avaliaram o *Penny Attack* na rede Monero [Chervinski, Kreutz e Yu 2019] [3]. Alguns aspectos são bem diferentes dos encontrados atualmente na literatura sobre o Ethereum, como o sucesso do ataque de acordo com o tempo de planejamento. Este ponto, no entanto, está intrinsecamente relacionado à maneira como essa blockchain funciona, não se aplicando a outras redes.

Apesar da ampla gama de pesquisas focadas na segurança e até diretamente no *Penny Attack*, nenhum trabalho tem como objetivo identificá-las usando diversos métodos de aprendizado de máquina, incluindo comitês de classificadores, que abre espaço para a pesquisa conduzida neste trabalho.

Em suma, este capítulo apresentou diversos trabalhos relacionados a criptomoedas, tais como Bitcoin, Ethereum e Monero, os quais fazem estudos como por exemplo análises de características da rede, características de transações, uma vasta gama de aspectos de segurança na *blockchain*. Apesar do volumoso número de estudos, ainda há oportunidades de pesquisa em temas tais como o tempo de espera na rede Ethereum e sua correlação com os demais atributos relacionados a uma transação. Outro ponto que se refere ao tempo de espera na plataforma do Ethereum é o aspecto de segurança, o qual existem limitados estudos que aplicam técnicas de aprendizado de máquina para identificar comportamentos maliciosos na rede, mais especificamente, com características de ataque de negação de

[3] https://www.getmonero.org

serviço DDoS. Este trabalho difere-se dos trabalhos supracitados por apresentar analises de correlação entre os atributos de transação no Ethereum e o tempo de espera, além do uso de técnicas de aprendizado de máquina para identificar um tipo de ataque de negação de serviço e, quanto antes o ataque for detectado, permitir a tomada de decisões para mitigar os efeitos do ataque.

4 COLETA DE DADOS DA REDE ETHEREUM

O presente capítulo tem por objetivo apresentar uma coleta de dados do Etehreum através de diferentes APIs públicas, obtendo dados da *mempool* e dos blocos do Ethereum.

4.1 CAPTURA DE DADOS

Existem dois principais meios de coletar dados da rede do Ethereum e sua *blockchain*: usando um cliente nativo do Ethereum ou por meio de uma API existente. Nesta dissertação, foram adotadas duas APIs bem conhecidas e amplamente usadas para coletar os dados do Ethereum: (i) a API do Etherscan [Etherscan 2020] e (ii) a API do Etherchain [Etherchain 2020]. Ambas APIs permitem um fácil acesso, confirmação e validação das transações que adentraram a *blockchain* do Ethereum. Além disso, os serviços de API podem ser usados para tanto construir aplicações descentralizadas ou servir como fonte de dados para informações da *blockchain* do Ethereum. A principal motivação para a utilização de duas APIs ao invés de centralizar a coleta de dados em somente um serviço é a limitação da frequência de requisições, imposta por parte dos provedores destes serviços, sendo assim, paralelizando as coletas consegue-se fazer mais requisições no mesmo período de tempo.

A Tabela 1 externa os *endpoints* utilizados para a captura das transações contidas tanto na *mempool* quanto as pertencentes a *blockchain*:

Tabela 1 – Endpoints utilizados para capturar dados.

Recurso	URL	API
Transações pendentes	www.etherchain.org/txs/pending/data?draw=&start=0&length=100.	Etherchain
Número do ultimo bloco	api.etherscan.io/api?module=proxy&action=eth_blockNumber	Etherscan
Último bloco com transações	api.etherscan.io/api?module=proxy&action=eth_getBlockByNumber&tag=0x10d4f&boolean=true	Etherscan

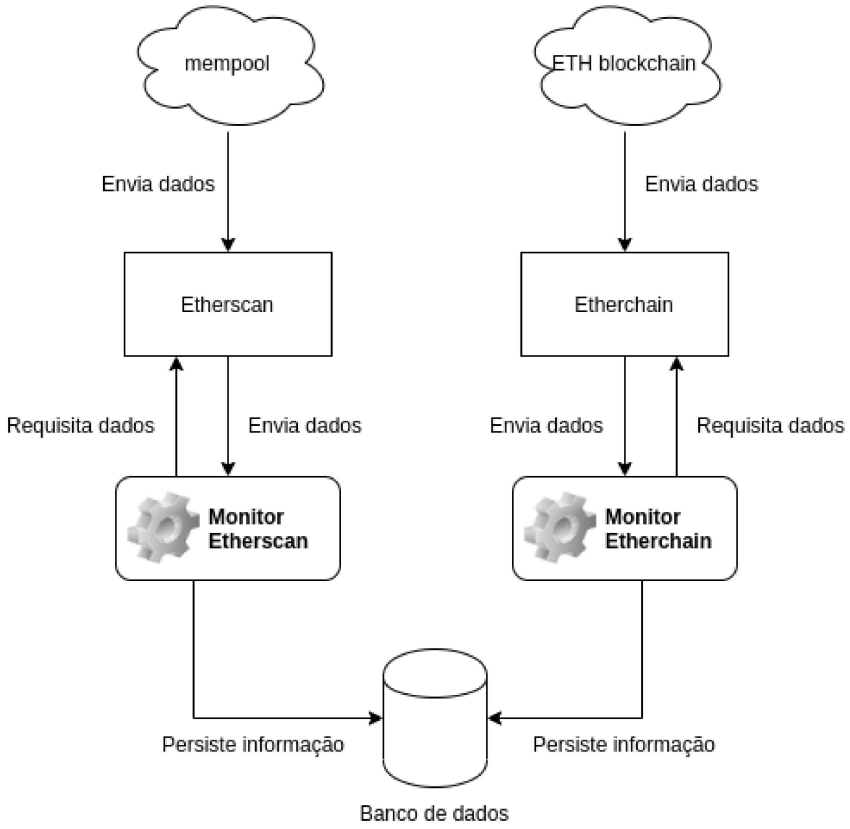
Fonte: Elaborado pelo autor. (2020).

Para utilizar as APIs é necessário fazer uma requisição HTTP para o *endpoint* da API REST em questão. Sendo assim, foram utilizados algoritmos em Python[1] para automatizar as requisições. Como são utilizadas duas APIs neste processo, foi utilizado um algoritmo para monitorar o Etherscan e um outro para monitorar o Etherchain. Em caso de sucesso da requisição, é obtido uma resposta em formato de JSON contendo os recursos solicitados, o qual o seu respectivo monitor é delegado para persistir as informações obtidas em um banco de dados relacional.

A metodologia de coleta contém dois passos principais. Primeiramente são coletadas todas as transações pendentes (transações e *smart contracts* esperando para serem incluídos na *blockchain* do Ethereum) na *mempool* do Ethereum. Então, são coletadas todas as transações da *blockchain* do Ethereum.

[1] https://www.python.org/

Figura 3 – Fluxo de coleta de dados do Ethereum.



Fonte: Elaborado pelo autor. (2020).

A Figura 3 mostra como é feito o procedimento de coleta dos dados. Periodicamente, são coletadas todas as transações pendentes (transações e *smart contracts* aguardando para serem incluídas na *blockchain*) na *mempool* do Ethereum com uma granularidade de um segundo. No momento em que é identificada uma nova transação (transação cujo *hash* não fazia parte da nossa base de dados), esta é inserida no banco de dados local. É usado data e hora atual para registrar o momento de nascimento da transação.

Outra coleta ocorre de forma paralela a esta. A cada cinco segundos é obtido o número do último bloco, o qual então são obtidas as transações pertencentes a ele. Todas as transações pertencentes à esse bloco em questão, fornecidas pela API do Etherscan, vide Tabela 1, são persistidas no banco de dados, obtendo todas as informações fornecidas pela API, as quais são descritas na Tabela 2.

As transações coletadas contém *hash*, número do bloco, destinatário, remetente, *gas*, *gas price*, carimbo do tempo, *nonce* e dados. Foram coletados também campos especiais, como as taxas de recompensa pelo minerador. A Tabela 2 apresenta todos os atributos que foram coletados das transações do Ethereum. A grande maioria é provida diretamente pela *blockchain*, enquanto atributos tais como o tempo de espera, *fee* e *paid* são calculados usando outros atributos ou valores externos tais como o valor de mercado do Ether, como

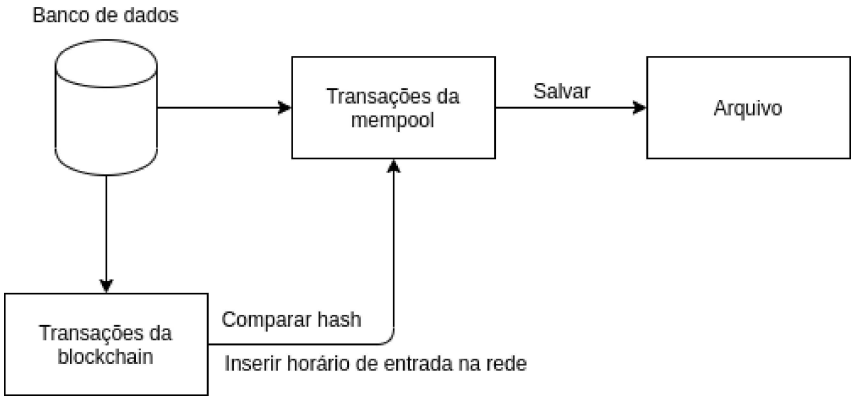
exposto na Tabela 2.

Tabela 2 – Atributos obtidos de transações no Ethereum.

Atributo	Descrição	Calculado?
hash	Texto único de tamanho fixo criado usando todas informações de uma transação.	Não
gas	Quantidade de *gas* oferecida para processar uma transação.	Não
gas price	Quantidade (em Ether) pago por unidade de *gas*.	Não
value	Quantia em Ether transferida de uma conta para outra.	Não
ether price	Valor monterário do Ether no momento em que o bloco foi minerado.	Não
fee	Taxa paga pela transação ou invocação de contrato em Ether (gas × gas price).	Sim
paid	Taxa paga pela transação ou invocação de contrato em dólares (fee × ether price).	Sim
pending time	Tempo entre a criação de uma transação e aparição na *blockchain* (tempo de espera).	Sim

Fonte: Elaborado pelo autor. (2020).

Figura 4 – Extração dos dados do banco de dados.



Fonte: Elaborado pelo autor. (2020).

A Figura 4 mostra o processo final para obter os dados do banco de dados. Após um longo período de coleta são pré-processados os dados obtidos. Cada transação contida na listagem da *mempool* é buscada na base de dados que contém os dados da *blockchain* através do *hash* da transação, onde foram contabilizadas 20.857.783 transações. É calculado então o tempo de espera baseado no tempo em que a transação adentrou a *mempool* e o tempo que a mesma iniciou sua participação na *blockchain*, sendo calculado como o segundo atributo subtraído do primeiro, obtendo um intervalo, o qual usamos o termo "tempo de espera" para referenciá-lo. Este atributo é uma unidade medida em segundos.

Ao final da coleta e extração dos dados é necessário um processo de limpeza dos dados. Note que a informação de cada API pode variar, visto que os sistemas do Etherscan e Etherchain não são sincronizados entre si. Em outras palavras, algumas transações podem aparecer na *blockchain* com carimbo do tempo anterior ao da Etherchain. Neste sentido, para ser conservador, não foram usadas as transações as quais o tempo de espera é menor que zero. Após este processo de limpeza, obteve-se 7.242.771 transação.

Com esta coleta, pode-se definir claramente alguns atributos que são calculados a partir de atributos coletados diretamente da rede do Ethereum:

1. Tempo de espera: O tempo de espera aqui é definido como o a diferença de tempo entre a primeira vez que uma transação é observada na rede do Ethereum e o tempo que a transação é devidamente adicionada a *blockchain*. Em outras palavras, o tempo de espera é o período que a rede do Ethereum demanda para aprovar ou reprovar uma transação/*smart contract*. Como mostrado na Figura 2, é o tempo gasto entre o estado inicial e o estado de aprovado ou reprovado. A fila de transações pendentes pode ser vista por um usuário, dependendo da configuração de seu cliente, oscilando de acordo com o nó utilizado para coletar os dados.

2. Taxa: A taxa do Ethereum é o pagamento efetuado ao minerador para processar uma transação/*smart contract*. A taxa é uma métrica direta, no entanto, neste trabalho, é preprocessada para avaliar sua forma indireta. O valor da taxa é calculado pelo produto de *gas* e *gas price*. Gas é o preço para executar com sucesso uma transação ou *smart contract* no Ethereum. O limit é proporcional ao número de instruções executadas na *blockchain*. No entanto, o usuário deve prover *gas* suficiente para cobrir os esforços computacionais necessários. Caso contrário, a transação ou *smart contract* é reprovado devido a falta de *gas*.

4.2 CARACTERÍSTICAS DOS DADOS

Foram coletados 20.857.783 transações entre 15 de abril de 2019 e 30 de agosto de 2019. No entanto, nós consideramos 7.242.771 destas transações para nossas análises, buscando maior precisão em nossas métricas de tempo de espera. É interessante ressaltar que este descarte de transações visando maior precisão das métricas de tempo de espera também ocorre em proporções semelhantes em cenários onde a coleta é feita em somente uma API (Etherscan ou Etherchain), favorecendo o cenário previamente descrito de coleta, onde duas APIs distintas são utilizadas como fonte de informação. A Tabela 3 resume os dados aqui analisados.

Tabela 3 – Resumo do dataset utilizado.

Dados completos	Dados após filtro	Transações	Smart Contracts	Período de coleta
20.857.783	7.242.771	2.800.733	4.442.038	04/2019-08/2019

Fonte: Elaborado pelo autor. (2020).

A maioria dos dados que são analisados neste trabalho são *smart contracts*. De fato, 61,33% dos dados que consideramos em nossa anaíse, depois do processo de limpeza, referem-se a contratos do Ethereum. A Tabela 4 resume o ciclo de vida das transações

usadas neste trabalho. Não foi possível inferir o estado de somente 4,09% dos casos. Por outro lado, quase 96% das transações foram propriamente mineradas. Quando minerada, praticamente todas as transações são aprovadas. Neste caso, 98,93% das transações alcançam o estado de aprovada, enquanto 1,07% das transações foram provadas.

Tabela 4 – Resumo dos dados coletados.

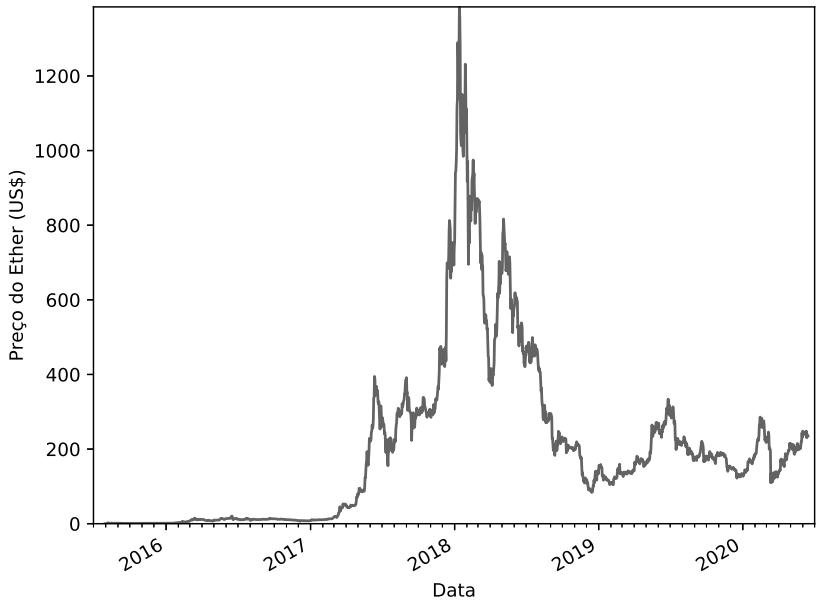
	Esperando	Minerado	Aprovado	Reprovado
Quantidade	296.229	6.946.542	6.872.214	74.328
%	4,09	95,91	94,88	1,02

Fonte: Elaborado pelo autor. (2020).

A Figura 5 mostra o valor de mercado da moeda Ether por dia. Este dado foi obtido diretamente pela API do Etherscan[2], desde o início da rede do Ethereum até janeiro de 2020. Este valor é utilizado no trabalho, sendo referido como *ether_price*, sendo utilizado, vide Tabela 2, para cálculo do atributo *paid*.

A Figura 6 ilustra a atividade do Ethereum por dia. Primeiro, como esperado, a atividade é baixa no período inicial do Ethereum. Percebe-se um grande crescimento tanto na Figura 5 quanto na Figura 6 até o final de 2017 e início de 2018, quando esta tendência muda rapidamente. Desde esta data, o valor do Ether e sua taxa de transações se mantém praticamente estáveis.

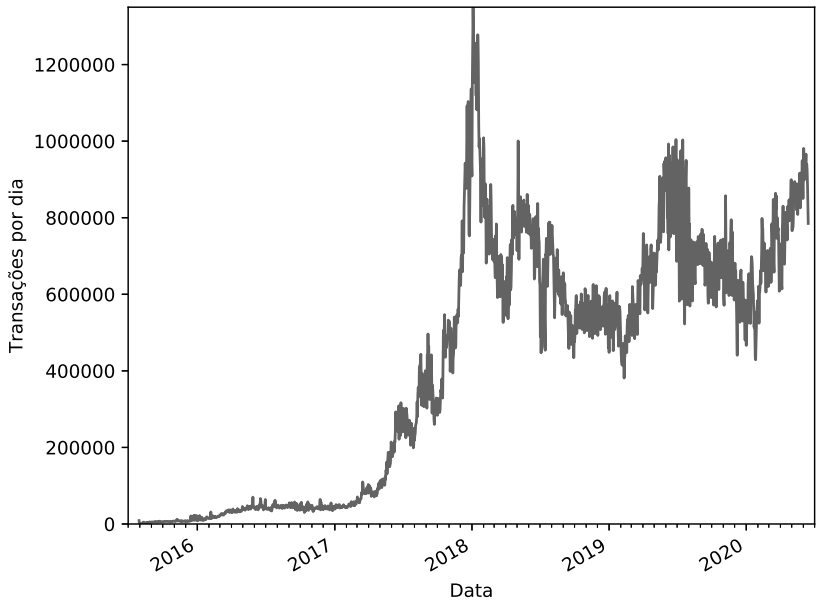
Figura 5 – Preço do Ether por dia.



Fonte: Elaborado pelo autor. (2020).

[2] https://etherscan.io/chart/etherprice

Figura 6 – Quantidade de transações por dia no Ethereum.

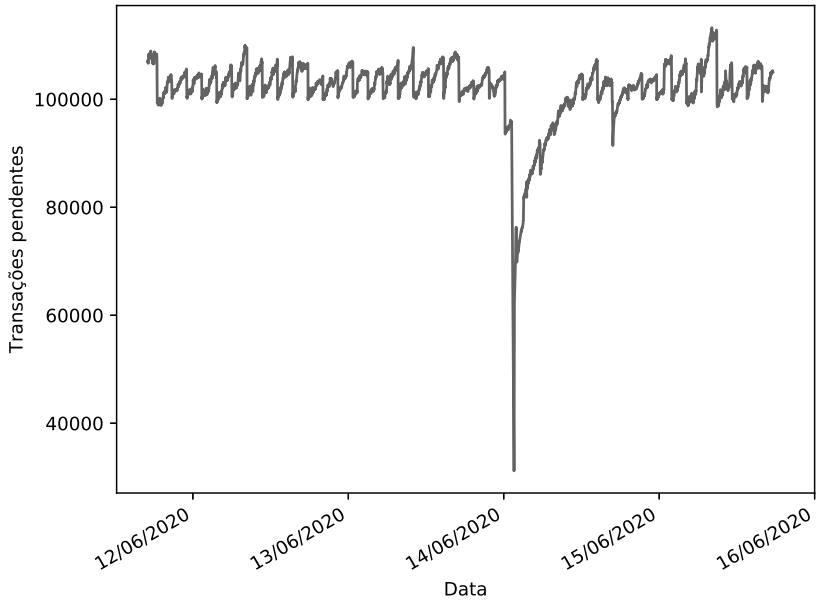


Fonte: Elaborado pelo autor. (2020).

A explosão da taxa de adesão a rede do Ethereum e o grande aumento do preço do Ether, em 2017, é bastante ligado ao aumento do preço de mercado do Bitcoin. O preço do Bitcoin em 2017 teme um crescimento máximo de aproximadamente 2.700% e no mesmo ano, algumas plataformas tais como o Ethereum, tiveram crescimento bem maior do que o do Bitcoin. Durante o colapso das criptomoedas em 2018, moedas tais como o Ether tiveram queda de aproximadamente 80% do seu pico em janeiro de 2018.

Apesar da grande variação no valor e taxa de transações do Ethereum, o número de transações pendentes é considerado estável. De fato, de acordo com a Figura 7, a *mempool* tem um número substancial de transações, sempre próximo a 40.000. Também, uma pequena redução ocorre no número de transações pendentes nos fins de semana, o que pode ser reflexo do uso do Ethereum em dias de trabalho.

Figura 7 – Quantidade de transações pendentes no Ethereum por data.



Fonte: Elaborado pelo autor. (2020).

5 ANÁLISE DO TEMPO DE ESPERA

Neste capítulo são avaliadas a correlação de atributos relacionados a taxa paga por processamento de transação, que por questões de simplificação chamamos de taxa, e o tempo de espera no Ethereum. O principal objetivo deste capítulo é confirmar ou refutar o senso comum de que usuários que pagam taxas maiores terão suas transações mineradas mais rapidamente. Neste sentido, é conduzida uma análise sobre as diferenças entre transações e *smart contracts* para verificar a possibilidade de conduzir de forma conjunta as análises destes. Em seguida são estudados os impactos das taxas no tempo de espera no Ethereum sobre diferentes perspectivas.

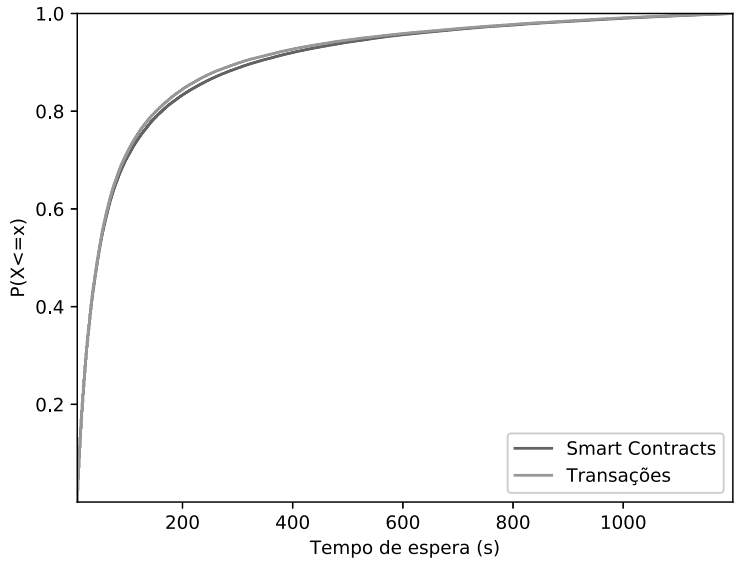
5.1 TRANSAÇÕES VERSUS CONTRATOS NO ETHEREUM

Nesta subseção são discutidas as similaridades e diferenças entre transações monetárias (o qual nos referimentos simplesmente como transações) e *smart contracts*. São comparadas transações e *smart contracts* obtidas através do método de coleta do Capítulo 4, entre o período de 04/2019 até 08/2019. Esta comparação é feita para averiguar se estas operações demonstram diferenças relevantes no tempo de espera e *gas price*.

Primeiro, a Figura 8 mostra a função distribuição acumulada do tempo de espera para *smart contracts* e transações separadamente. O eixo x mostra o tempo de espera em segundas, enquanto o eixo y mostra a respectiva probabilidade $P(X \leq x)$, ou seja, a probabilidade de uma transação ou *smart contract* ser aprovada com um tempo menor ou igual a x. Visualmente, ambas distribuições apresentam medianas próximas. No entanto, a média de tempo de espera para *smart contracts* é ligeiramente menor do que o valor médio para transações. Enquanto o intervalo de confiança do tempo de espera para *smart contracts* é de 122,61-123,31 s, o intervalo de confiança para transações vai de 117,54-118,32 s com um nível de confiança de 95%. *Smart contracts* e transações apresentam tempo de espera estatisticamente diferentes, porém bastante próximos em valores absolutos. Essa proximidade entre os valores do tempo de espera também ocorre nos diferentes percentis da amostra, sendo o 25 percentil igual para ambos (21 s). Além disso, a maioria do tempo de espera para ambas distribuições (até 95%) são iguais ou menor que 600 segundos. Em outras palavras, remetentes e destinatários tem alta probabilidade de esperar não mais do que 10 minutos para confirmar suas transações ou *smart contracts* no Ethereum. Pode-se observar que o tempo de espera é maior do que um usuário comum estaria disposto a esperar em uma transação financeira cotidiana, como o uso de seu cartão de crédito.

Outrossim, é avaliada a diferença do *gas price* entre transações e *smart contracts*, como mostra a Figura 9. Novamente, ambas distribuições apresentam padrão visual similar. De fato, a mediana, o *gas price* mínimo e máximo tem valores bem próximos para ambos. Neste caso, a mediana do *gas price* é proximo a 7 GWei. Diferentemente do tempo de

Figura 8 – Função distribuição acumulada do tempo de espera.



Fonte: Elaborado pelo autor. (2020).

espera, neste caso, ambas distribuições apresentam valores de média de *gas price* similares. Mais precisamente, o intervalo de confiança da média do valor de *gas price* é $7,26 \pm 0.1$ GWei para *smart contracts* e $7,27 \pm 0.1$ GWei para transações.

Em suma, apesar da diferença do valor médio de tempo de espera para transações e *smart contracts*, a distribuição para transações e *smart contracts* apresentam padrões similares, tais como o primeiro e segundo quartis. Além disso, ambos são estatisticamente equivalentes em termos de *gas price*. Então, nesta dissertação, a análise é conduzida considerando transações e *smart contracts* conjuntamente nos conteúdos seguintes.

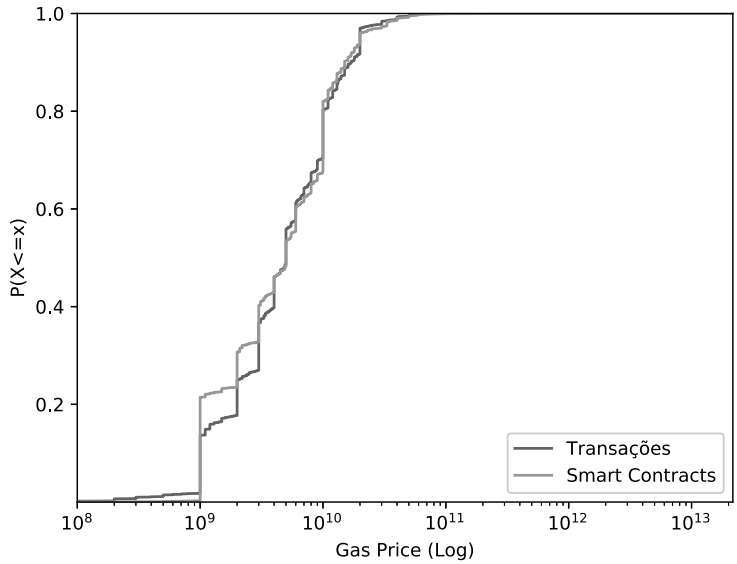
5.2 CORRELAÇÕES ENTRE ATRIBUTOS DE TRANSAÇÕES DO ETHEREUM

Nesta subseção são estudados os impactos das taxas no tempo de espera de transações do Ethereum. Além disso, é avaliada a relação entre os demais atributos possíveis de se extrair dos dados obtidos no Capítulo 4.

Como previamente mencionado, são consideradas transações e *smart contracts* conjuntamente por questões de simplicidade. De fato, como previamente demonstrado, as transações em criptomoedas e *smart contracts* apresentam tempo atributos bem próximos, tais como tempo de espera e atributos relacionados a taxa.

São utilizados diferentes métodos estatísticos para quantificar a correlação entre todos os atributos. Os coeficientes de correlação de Pearson e Spearman [Jain 1990] foram calculados para capturar as correlações lineares e não lineares, respectivamente.

Figura 9 – Função distribuição acumulada do *gas price*



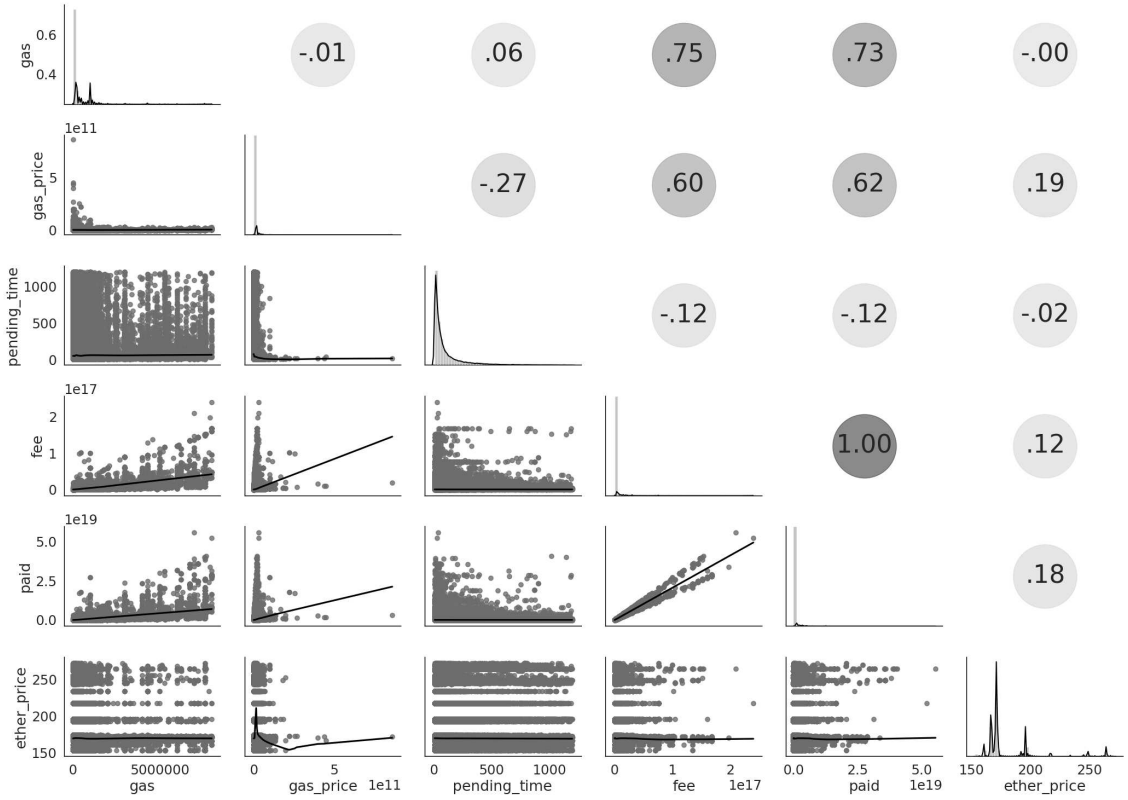
Fonte: Elaborado pelo autor. (2020).

No restante desta seção, os resultados apresentados referem-se a correlação de Pearson, visto que foi dado preferencia para mostrar os melhores resultados de correlação. Apesar disso, mesmo sendo utilizado o maior coeficiente de correlação, as análises empíricas mostraram evidências fortes de que não há uma clara correlação entre atributos das transações relacionados a taxas e o tempo de espera.

A Figura 10 mostra os coeficientes de correlação de Pearson entre todos os atributos que foram extraídos das transações do Ethereum. Esta figura apresenta uma matriz onde são descritas três informações importantes: (i) a diagonal principal da matriz apresenta a distribuição dos dados; (ii) o triângulo inferior da matriz exibe o gráfico de dispersão para cada par de atributos (*gas price* versus *gas*, por exemplo) e, (iii) o triângulo superior da matriz mostra o coeficiente de correlação de Pearson entre os pares de atributos. Adicionalmente, são usados mapas de calor para distinguir o quão próximos são os coeficientes de correlação do máximo e mínimo da escala usada que são -1 e 1. Cores vermelhas representam um coeficiente de correlação positivo, enquanto cores azuis representam uma correlação negativa. O mais próximo uma cor é de tons escuros, mais alto é o modulo do seu valor. Para ambos os casos, um coeficiente de correlação próximo a zero, que é definido por cores claras, indica que não há correlação entre o par de atributos.

De acordo com os resultados apresentados na Figura 10, não há uma correlação clara entre o tempo de espera e as taxas, tal como com qualquer outro atributo. Em especial, não há uma correlação clara entre o tempo de espera e outro atributo relacionado a taxa, como *gas* e *gas price*. Além disso, não é possível observar uma clara correlação

Figura 10 – Matriz do Índice de correlação de Pearson para todos os dados coletados, onde, (i) a distribuição dos dados é exibida na diagonal principal; (ii) o gráfico de dispersão na porção trinagular inferior; e (iii) o coeficiente de correlação de Pearson na porção triangular superior.



Fonte: Elaborado pelo autor. (2020).

entre o tempo de espera e o preço do Ether, ou valor final da transação, ambos possuindo unidade em dólares. Mais precisamente, estes coeficientes de correlação de Pearson variam entre -0.27 e 0.06, o qual provê uma forte evidência de que não há uma relação entre o tempo de espera e outros atributos analisados.

No entanto, observa-se uma forte correlação somente entre os pares de atributos que representam taxas e valores pagos, onde o coeficiente de correlação de Pearson varia entre 0.60 e 1.00. Tais valores de correlação altos são esperados visto que a maioria destes atributos tem alguns componentes em comum, como mencionado no Capítulo 4.

Estudos prévios que analisam taxas em criptomoedas, especialmente o Bitcoin, concordam que os valores de taxa levam a diferentes níveis de prioridade para executar transações [Easley, O'Hara e Basu 2019, Narayanan et al. 2016]. Isto também é um senso comum para a maioria dos serviços e especialistas de criptomoedas [Buterin 2017]. Por exemplo, MyEtherWallet, um serviço popular para usuários no Ethereum, recomenda 21000 *gas* para transações usuais no Ethereum, onde o *gas price* pode variar entre 2 e

40 GWei para curtos e longos períodos de espera, respectivamente[1]. Vale lembrar que usuários podem oferecer diferentes valroes de taxas no Ethereum graças ao *gas price* e *gas* que são disponíveis para pagar por transação. Esta análise, no entanto, mostra que taxas altas ou baixas não estão relacionadas com transações que são mineradas rapidamente ou com grande atraso. Então, pagar altas taxas pode levar os usuários a crer que esta estratégia é eficiente. Essa é uma questão importante para os usuários do Ethereum que merece mais atenção; em seguida, realizamos uma análise de correlação mais detalhada.

5.2.1 O impacto do *gas* e *gas price* em correlações

Nesta subseção, foca-se na correlação entre grupos selecionados de transações que são levadas em consideração os valores de *gas* e *gas price*, os quais são atributos de taxas definidos pelos usuários no Ethereum. A intenção aqui é analisar se valores incomuns de *gas* e *gas price*, também chamados de pontos fora da curva, podem conter correlações fortes entre estes atributos e o tempo de espera. Estes pontos foras da curta podem ser causados principalmente por usuários que não entendem do mecanismo de taxas no Ethereum e definem valores claramente não usuais, possivelmente ineficazes para os atributos de taxa.

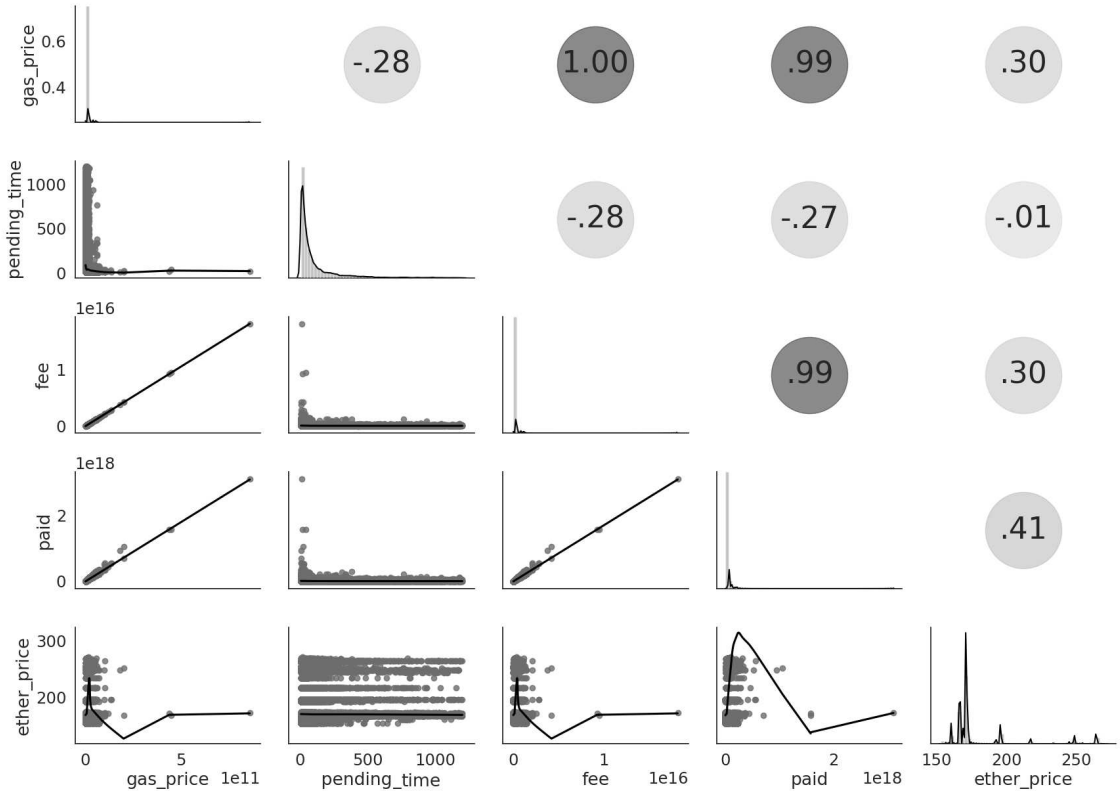
Primeiramente são selecionadas transações onde os usuários definem *gas* igual a 21000. Este valor é normalmente adotado para transações rotineiras pelos serviços mais populares que colaboram com os usuários do Ethereum durante o tempo da escrita desta dissertação[1]. Observa-se que 21000 é o valor mais comum (a moda) para este atributo entre os nossos dados, e é usada em 15,08% de todas as transações. Uma vez selecionado este valor, são obtidas as correlações entre todos os atributos extraídos dos dados obtidos para analisar como esses atributos estão relacionados entre si.

A Figura 11 mostra os resultados para transações cujo *gas* é igual a 21000. Pode-se observar que estes resultados são bem parecidos com os apresentados na Figura 10. Sendo assim, não são observadas mudanças relevantes, se comparadas as correlações já existentes discutidas previamente. Apesar disso, valores maiores de correlação são encontrados entre o tempo de espera e atributos relacionados a taxa (0,28 versus 0,27 da figura anterior). Estas novas correlações, no entanto, ainda não representam clara evidencia sobre a relação entre o tempo de espera e estes atributos. Pode-se observar pelo gráfico de dispersão para estes pares de atributos que a grande maioria dos pontos (transações) não seguem um senso comum entre os especialistas de criptomoedas [Buterin 2017], onde tempos de espera curtos são tipicamente associados com altas taxas. Por contraste, não há padrões claros que possam ser observados para ambas relações.

A Figura 12 mostra como atributos correlacionam entre si, para metade das transações contidas nos dados, e que contém *gas price* intermediário, incluindo a primeira e segunda moda. Pode-se observar que estes resultados não mostram mudanças relevantes, se

[1] https://kb.myetherwallet.com/en/transactions/what-is-gas

Figura 11 – Matriz de coeficiente de correlação de Pearson para transações cujo *gas* é igual a 21000.

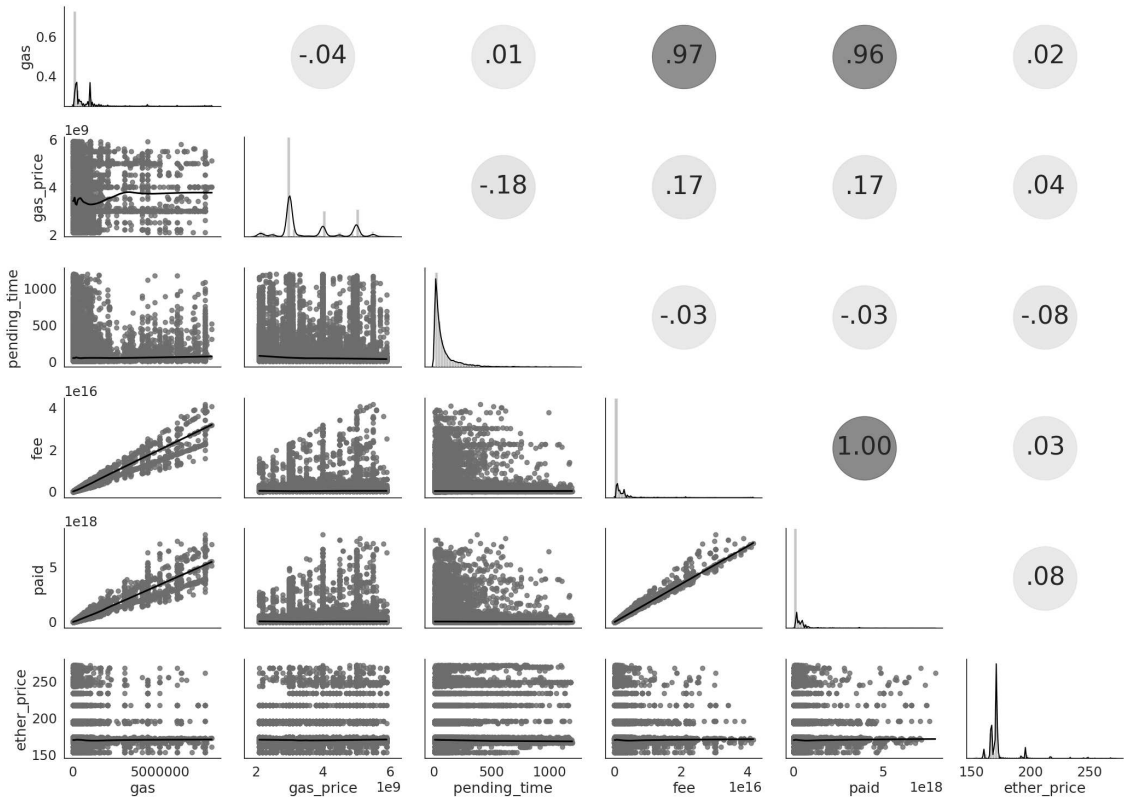


Fonte: Elaborado pelo autor. (2020).

comparadas a Figura 10, onde são usados todos os dados. Neste caso, onde são analisados os *gas prices* entre o 25 e 75 percentil, o coeficiente de correlação de Pearson entre o tempo de espera e a taxa, bem como tempo de espera e o valor pago, são mais baixos até mesmo que os valores presentes nas análises anteriores. Em suma, um coeficiente de correlação de Pearson próximo a -0.05 é uma clara evidência de que não há correlação entre estes atributos. Como previamente comentado, os resultados utilizando o coeficiente de Spearman tem valores em módulo menor dos que os apresentados , neste caso, estes resultados foram omitidos deste trabalho.

Foram feitas análises extras selecionando transações com *gas* maior que 21000 (visto que *gas* menor que este valor não é permitido no Ethereum) e *gas price* diferente dos valores intermediários, como por exemplo, valores abaixo do 25 percentil e acima do 75 percentil. Também foi analisado o valor da segunda moda para o *gas*, o que significa 11,97% de todas as transações de nossos dados. Em suma, percebe-se um baixo valor do coeficiente de correlação de Pearson entre o tempo de espera e a taxa, como também o tempo de espera e o valor pago pela transação. Por exemplo, o valor da correlação de Pearson entre o tempo de espera e outros atributos variam entre -0,11 e -0,01, ao usar

Figura 12 – Matriz de coeficiente de correlação de Pearson para transações com *gas prices* está entre o 25 e 75 percentil.



Fonte: Elaborado pelo autor. (2020).

o valor da segunda moda. Como os resultados para estes valores são qualitativamente similares aos resultados previamente mostrados, são omitidos desta seção por brevidade.

5.3 CORRELAÇÃO POR AGRUPAMENTO AUTOMÁTICO

Previamente foi analisado a correlação de atributos dos dados obtidos. Adicionalmente, a correlação foi calculada agrupando transações manualmente usando valores de *gas* e *gas price*. Em todas as situações, nenhuma correlação representativa foi observado. Entao, nesta seção é analisada a correlação do tempo de espera com atributos relacionados a taxas de transações no Ethereum por meios de técnicas de agrupamento.

Existem vários métodos de agrupamento de dados e cada uma destas técnicas apresenta seus pontos fortes e aplicabilidades. De mesmo modo, existem várias outras métricas para avaliar a qualidade dos grupos gerados, por exemplos, a similaridade dos grupos. Nesta seção é adotado o *Ordering Points To Identify the Clustering Structure* (OPTICS) [Ankerst et al. 1999], uma abordagem baseada em densidade. Este tipo de abordagem foi adotada visto que o algoritmo gera grupos de formato arbitrário ou não-esféricos, e regiões do espaço de dados com uma menor densidade não são associadas a

nenhum grupo.

Neste sentido, esperamos avaliar o impacto das taxas no tempo de espera no Ethereum com porções de dados menores, porém, com características similares. A seguir são descritas a metodologia de agrupamento, são externados os grupos encontrados e, para cada grupo, apresentado e discutido os resultados obtidos.

5.3.1 Metodologia de agrupamento

Aqui são apresentadas e discutidas as metodologias usadas para agrupar os dados. A ideia principal é agrupar transações com características parecidas de forma automática, usando como meio de distingui-las os atributos de *gas*, *gas price* e tempo de espera.

No geral, os métodos de agrupamento podem ser classificados como [Han, Kamber e Pei 2011]: particionada, hierárquica, baseada em densidade e baseada em grade. Não foi claramente identificado uma hierarquia ou organização em grade, visto que transações não possuem estrutura hierárquica. Além disso, os dados aqui apresentados podem conter transações não representativas (ruídos). Como consequência, os métodos de particionamento, hierárquicos e baseados em grade podem não identificar instâncias que não pertencem a nenhum grupo, visto que estes métodos sempre classificam dados em um grupo. Então, nesta seção foi adotada uma técnica baseada em densidade. Tal técnica de agrupamento agrupa dados de forma não-esférica ou com formato arbitrário. Regiões com menor densidade não são associadas a nenhum agrupamento e, então, pode-se dizer se um dado pertence a um grupo ou não. Por exemplo, uma transação incomum, ou um ponto fora da curva nos dados não será agrupado a nenhum grupo. De fato, estes dados podem ser considerados ruídos e descartados da análise, considerando-os como uma transação de comportamento não representativo, ao ser comparado ao conjunto de dados.

O *Density-based Spation Clustering Applications with Noise* (DBSCAN) [Ester et al. 1996] é uma das abordagens de agrupamento por densidade mais populares. No DBSCAN, a métrica de distância entre instâncias (neste caso, a transação) é usada para definir as conexões. Para esta pesquisa foram utilizados o *gas*, *gas price* e o tempo de espera para cada calcular a distância de cada instância. Em outras palavras, dado um grupo, transações terão valores próximos (pequena distância) de *gas*, *gas price* e tempo de espera. O limite para definir uma pequena distância é um parâmetro definido pelo usuário. O método tenta agrupar instâncias próximas e, cada vez que o número de instâncias com baixa distância for maior que o limite definido pelo usuário, o método cria um novo grupo e tais transações passam a pertencer a tal grupo.

Apesar da popularidade e bons resultados reportados na literatura, a definição adequada dos parâmetros dificultou a aplicação do DBSCAN. Então, foi adotado aqui o Ordering Points To Identify Structure (OPTICS) [Ankerst et al. 1999]. No OPTICS, é criada uma lista ordenada na qual as instâncias de um grupo mais denso são listadas mais

próximas umas das outras. A lista criada é semelhante à aplicação de um grupo baseado em densidade usando uma ampla variedade de configurações de parâmetros [Han, Kamber e Pei 2011] tais como distância mínima e limite inferior de membros em um grupo.

Foi usado o Scikit-learn[2] e o PyClustering[3] para implementar os métodos e automaticamente agrupar as transações utilizando o OPTICS. Para reduzir consideravelmente o esforço computacional para agrupar as transações, foi aplicado o OPTICS em uma subamostragem de 40.000 instâncias aleatoriamente selecionadas.

Os atributos utilizados foram normalizados (*gas*, *gas price* e tempo de espera) entre $[0, 1]$ utilizando `MinMaxScaler`. Neste sentido, atributos se tornam admensionais e igualmente importantes para o procedimento de agrupamento. A implementação do OPTICS feita pelo PyClustering foi usada com um máximo de raio entre instâncias (`eps`) igual a 10, o número mínimo de transação pertencentes a cada grupo (`minpts`) equivalente a 15, e o número de grupo esperado (`amount_clusters`) foi definido como 4. Estes parâmetros foram empiricamente definidos.

A Figura 13 represente os dados normalizados usados no procedimento de agrupamento, juntamente com os grupos que foram encontrados. As cores azul-escuro, rosa e vermelho representam os grupos, enquanto as instâncias que não foram agrupadas estão representadas em azul-claro. Estas instâncias plotadas em azul-claro são consideradas transações não representativas, visto que não são similares (dado um raio mínimo) as várias outras (em termos de número mínimo de vizinhos). Então estas transações não representativas não foram analisadas aqui. Então, são analisados os grupos, por meio das transações pertencentes a ele.

5.3.2 Análise das transações agrupadas

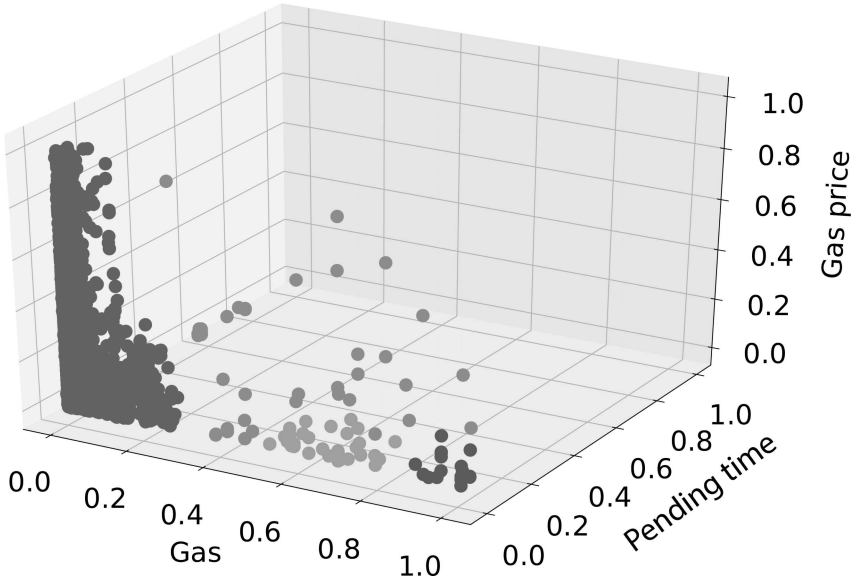
Aqui, a amostragem de transações é analisada considerando os grupos criados. O *gas* é o recurso que pode explicar principalmente os grupos. De fato, percebe-se grupos com valores baixos, médios e altos de *gas*. No entanto, esse recurso não é suficiente para dividir todas as transações em grupos, pois existem vários dados não representativos. De acordo com a Figura 13, a maior parte das transações pertencem ao grupo azul-escuro. Neste caso, as transações apresentam baixo valor de *gas* e baixos valores de tempo de espera. Além disso, o *gas price* se espalha por praticamente todos os possíveis valores.

O tempo de espera parece ser irrelevante para definir grupos na Figura 13. A maioria das transações pertencentes a qualquer grupo tem curto tempo de espera. A seguir, são estudados os impactos das taxas no tempo de espera em transações do Ethereum considerando os três grupos encontrados durante o processo de agrupamento.

As Figuras 14, 15 e 16 mostram as matrizes de correlação de todos os atributos

[2] `https://scikit-learn.org`
[3] `https://pypi.org/project/pyclustering`

Figura 13 – Transações do Ethereum agrupadas. Os três grupos encontrados são destacados em azul escuro, rosa e vermelho. Os pontos em azul claro representam as instâncias que não pertencem a nenhum grupo.



Fonte: Elaborado pelo autor. (2020).

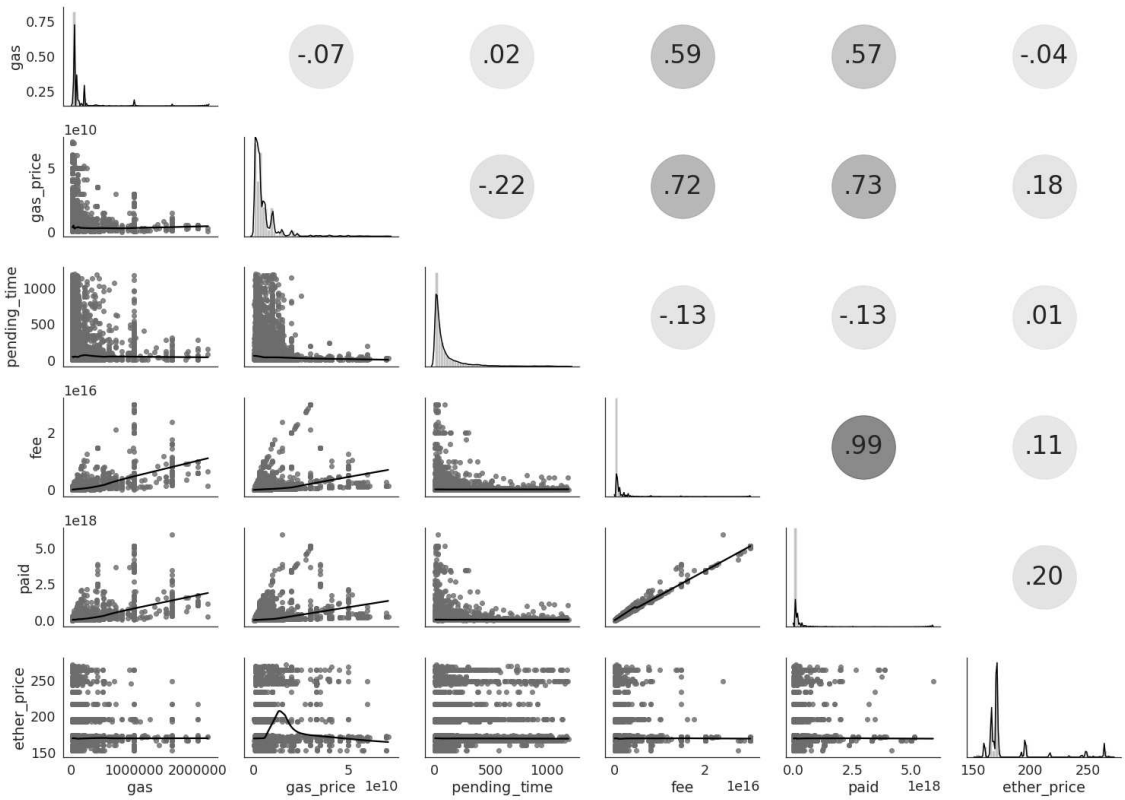
que foram extraídos das transações do Ethereum, para cada grupo encontrado. Foram omitidas as instâncias não representativas (transações que não foram inseridas em algum agrupamento, e portanto, apresentam quantidade negligenciável).

Novamente, como observamos na análise considerando todo o conjunto de dados, não foi possível observar uma clara correlação entre o tempo de espera e outros atributos relacionados a taxa, tais como *gas* e *gas price*. De fato, para todos os três grupos, a correlação entre o tempo de espera e valores das taxas é insignificante. Semelhante à análise anterior, também observamos uma forte correlação apenas entre um par de recursos de taxa e *paid*, uma vez que esses recursos têm alguns componentes comuns, conforme mostrado na Tabela 2.

Porém, diferentemente da análise anterior, os dois grupos com maiores valores de *gas*, quando comparados ao azul escuro, apresentam alta correlação entre o *gas price* e o valor pago, e entre o *gas price* e valor da taxa. Esses coeficientes de correlação de Pearson variam de 0,93 a 0,97, o que indica uma forte relação positiva entre eles.

A baixa correlação entre a taxa fornecida nas transações e o tempo de espera das mesmas, observada nas diversas análises conduzidas neste capítulo permite concluir que usuários da rede os quais possuem menor disponibilidade de aporte financeiro para executar suas transações também têm capacidade de participar ativamente da rede, tendo suas transações mineradas em tempos comparáveis com as transações que proveem grandes valores para que as suas transações sejam mineradas.

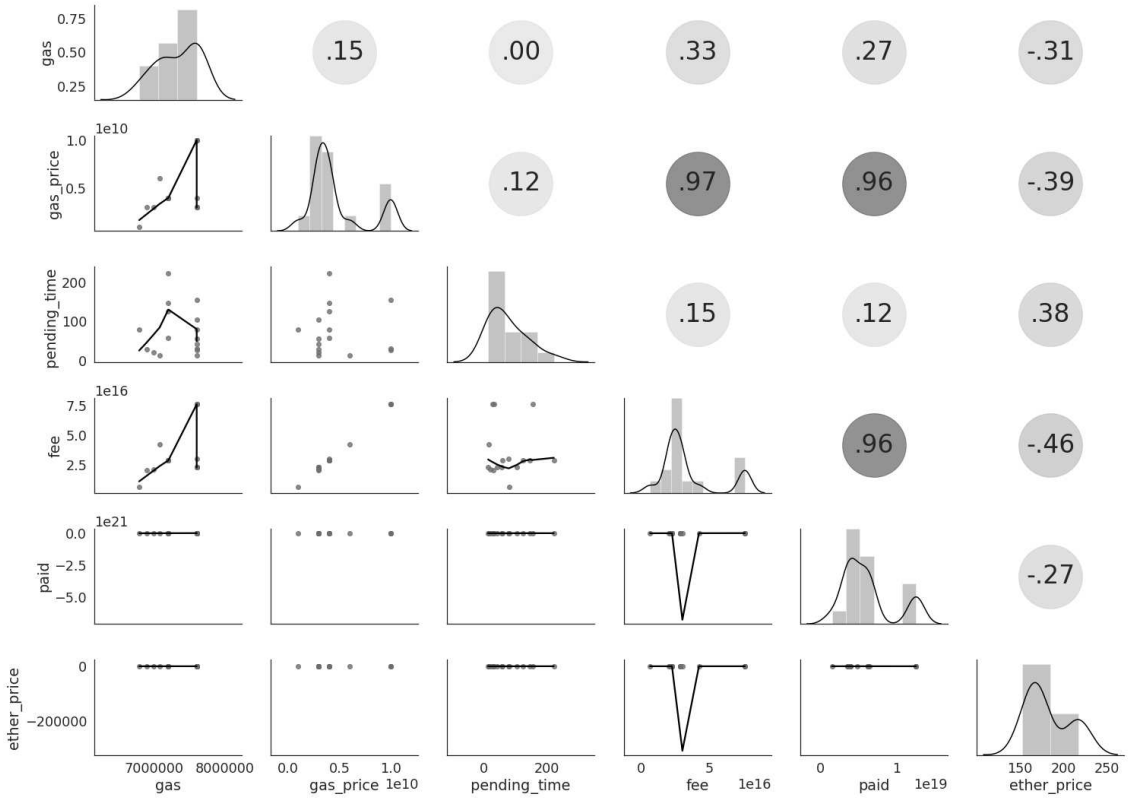
Figura 14 – Mapa de calor da matriz de correlação de Pearson para as transações no Grupo 1 (grupo azul escuro na Figura 13).



Fonte: Elaborado pelo autor. (2020).

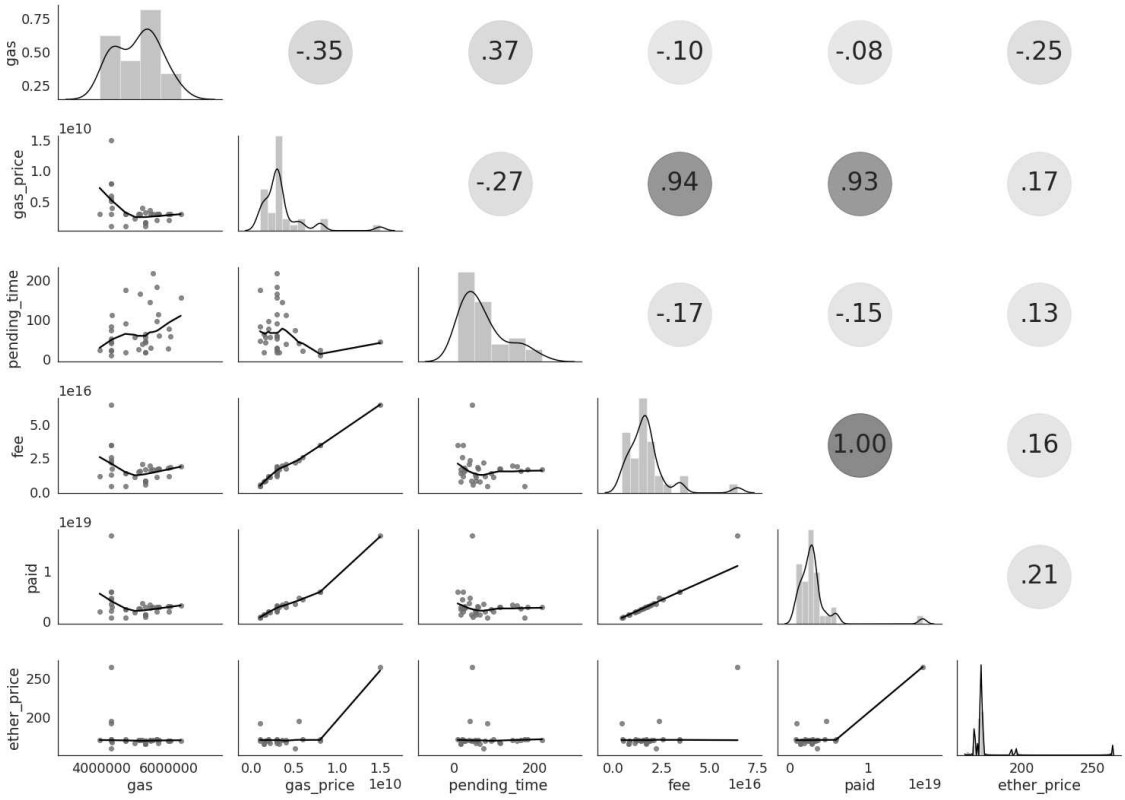
Em suma, as análises de correlação mostram que taxas altas ou baixas não estão relacionadas às transações, mesmo as pertencentes ao mesmo grupo de transações. Por outro lado, o mecanismo de agrupamento automático que foi adotado agrupou principalmente as transações de acordo com o *gas*. O *gas price*, dependendo da faixa de valor, apresenta uma alta correlação com a taxa e também com o valor pago. Podemos pensar que, caso o preço seja alto, os usuários gastam uma quantidade maior de Ether (valor pago) e também pagam uma taxa mais alta, desejando que suas transações sejam minadas mais rapidamente.

Figura 15 – Mapa de calor da matriz de correlação de Pearson para as transações no Grupo 2 (grupo rosa na Figura 13).



Fonte: Elaborado pelo autor. (2020).

Figura 16 – Mapa de calor da matriz de correlação de Pearson para as transações no Grupo 3 (grupo vermelho na Figura 13).



Fonte: Elaborado pelo autor. (2020).

6 AVALIAÇÃO DE PENNY ATTACK NO ETHEREUM

Neste capítulo, discute-se sobre ataques DDoS na rede Ethereum. Mais especificamente, definimos um ataque com características particulares, o qual denominamos *Penny Attack*. A partir da definição, é avaliado o ataque em uma rede que simula o Ethereum. São então propostas contra-medidas para reduzir os danos deste tipo de ataque, dispondo de técnicas de aprendizado de máquina e classificadores para identificação precoce do *Penny Attack*. Inicialmente define-se o *Penny Attack*. Na subseção 6.1, apresenta-se as ferramentas utilizadas para o desenvolvimento do ambiente experimental. Na subseção 6.2, apresenta-se a metodologia de avaliação do Ethereum sob um ataque desse tipo. Na subseção 6.3, são discutidos os resultados alcançados pela simulação e na seção 6.4 são avaliadas as abordagens adotadas para detectar o ataque. Por último, é apresentada uma conclusão sobre o capítulo atual.

O crescente interesse no Ethereum também incita ameaças à segurança e ataques à sua plataforma de criptomoeda. Ataques de negação de serviço (DoS) são um dos maiores desafios que os sistemas de criptomoedas enfrentam [Chen et al. 2017], uma vez que esses ataques afetam a experiência do usuário. Nesse tipo de ataque, diversos usuários mal-intencionados acionam o sistema, mantendo-o ocupado. Nesse sentido, o sistema de criptomoeda não é capaz de atender adequadamente seus clientes genuínos. Apesar do pagamento de taxas para realizar uma transação na maioria das criptomoedas, usuários mal-intencionados já exploraram este mecanismo para aplicar ataques DoS [Chen et al. 2020]. A plataforma Ethereum, em particular, também sofreu ataques que exploravam instruções de baixo custo e falhas na máquina virtual (EVM), causando congestionamento na rede e perdas de contratos e transações [Gautham 2016]. Além disso, apesar do impacto nos sistemas de criptomoedas, ainda faltam estudos acadêmicos que analisem os ataques a essas plataformas e seus impactos.

A segurança em *blockchains* se tornou um tópico popular em diversos setores [Bradbury 2015, Ulusoy 2018, Atzei, Bartoletti e Cimoli 2017, Wüst e Gervais 2016]. Em especial, há três tópicos abordados: (1) a quantidade de esforço necessário para burlar o sistema de consenso distribuído [Takefuji 2020, Bagay 2020]; (2) brechas nas linguagens de contrato, o que já levou, por exemplo, a dividir a rede Ethereum em duas (i.e., *hard fork*) [Vujičić, Jagodić e Ranjić 2018, Mehar et al. 2019] e, (3) comportamentos que causam lentidão na rede.

Em 2016, ataques que utilizavam de instruções para programação de contratos inteligentes no Ethereum tais como *exitcodesize* [Buterin 2016] e *suicide* [Buterin 2017] foram descobertos [Atzei, Bartoletti e Cimoli 2017]. Devido as características dessas instruções, que anteriormente ao ataque apresentavam baixo custo financeiro e alto custo computacional, elas foram utilizadas demasiadamente de forma proposital com intuito de

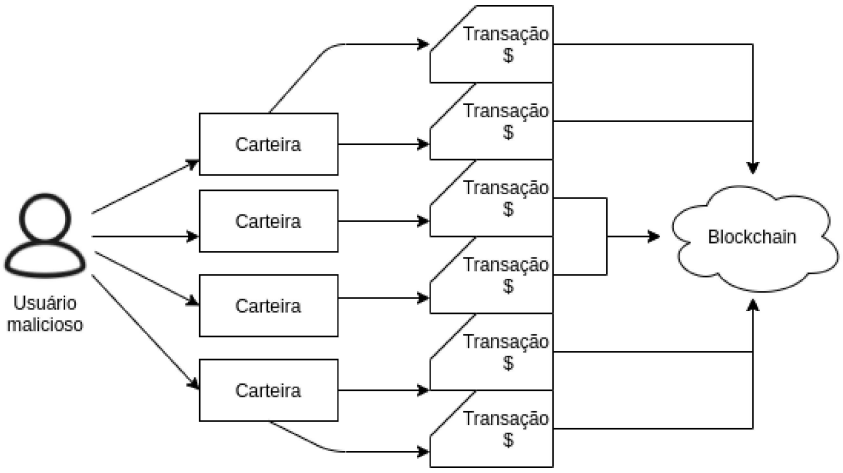
gerar lentidão na rede e desperdício de recursos computacionais. Em um desses ataques, vários contratos inteligentes que continham 50.000 execuções da operação *exitcodesize* foram criados. Essa instrução tem baixa tarifação, ou seja, utiliza pouco *gas*, mas leva à ocupação dos recursos de entrada e saída nos clientes Ethereum (mineradores) quando executada com alta frequência [Buterin 2016]. Em outro ataque, foram criados contratos inicializados com estruturas de repetição contendo a operação *suicide*, que também tem baixa tarifação e serve para abortar a transação em caso de erros, transferindo o valor restante para uma determinada conta. Essa instrução, no entanto, foi utilizada maliciosamente para gerar contas falsas, ocupando espaço em disco dos clientes [Buterin 2017]. Todos esses ataques implicaram em lentidão no sistema, o qual é acumulado com as novas transações, cascateando para toda a rede Ethereum um maior tempo de espera, visto que as transações da *blockchain*, dado o protocolo de consenso corrente, precisam ser replicadas e executadas em todos os mineradores da rede.

A plataforma Ethereum continua susceptível a esses tipos de ataques pois as soluções utilizadas para mitigá-los não foram definitivas [Chen et al. 2017]. A maioria das propostas existentes [Chen et al. 2017, Saad et al. 2019, Chervinski, Kreutz e Yu 2019] dependem de medidas que alteram a arquitetura do sistema blockchain. Por exemplo, esses trabalhos propõem aumentar o tamanho do bloco, o valor da taxa ou o custo computacional do mecanismo de consenso para minimizar o impacto dos ataques ao blockchain. Na abordagem desta dissertação, como alguns outros autores [Baqer et al. 2016], tenta-se detectar transações suspeitas e, então, pode-se aplicar efetivamente qualquer medição de contador existente. Porém, enquanto em [Baqer et al. 2016] os autores agrupam as transações, neste trabalho, conta-se com um conjunto de técnicas de aprendizado de máquina para poder identificar transações suspeitas.

Neste capítulo, é abordado um tipo de DDoS, o qual é aqui denominado de *Penny Attack*. Suas características são de um ataque *Sybil*, onde uma entidade mal intencionada subverte o sistema de reputação de um serviço da rede criando um grande número de identidades pseudônimas e as usa para obter uma influência desproporcionalmente grande [Zhang e Lee 2019]. Em específico no *Penny Attack*, o ataque é estabelecido com várias transações em um curto período em que o custo total (valor enviado com taxas) é insignificante [LeMahieu 2018]. Este tipo de ataque já é previsto na literatura, divergindo de nome entre autores, sendo identificado também como *mempool flooding*, *penny flood attack*, *under-priced DoS attack*, entre outros [Chen et al. 2017, Saad et al. 2019, Kruglik et al. 2019, Saad et al. 2020].

A Figura 17 ilustra o estabelecimento de um *Penny Attack*, onde um ou mais usuários fazem a criação de várias interações com a rede, o que pode ser uma transação financeira ou *smart contract*, e fazem sua divulgação na *mempool*, que tem a função de um repositório de transações não confirmadas. A partir da criação destas interações em um local, elas são divulgadas para a rede, de forma que os nós passam a contê-la na *mempool*,

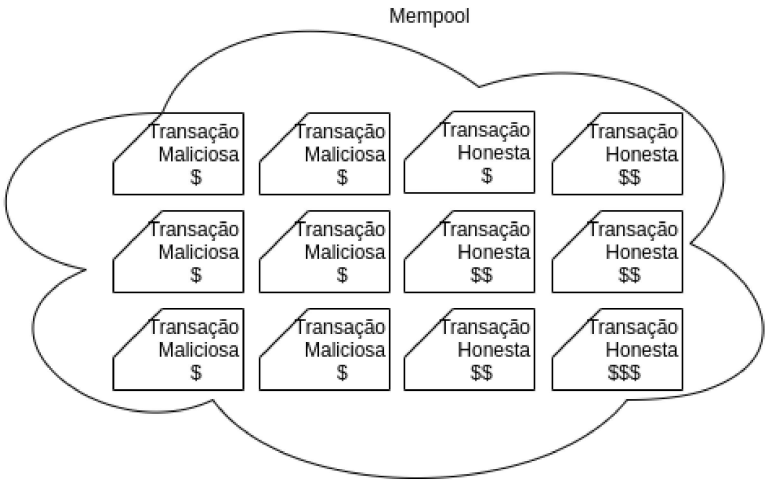
Figura 17 – Exemplo do estabelecimento de um *Penny Attack*.



Fonte: Elaborado pelo autor. (2020).

sendo passível de mineração. Caso a quantidade de transações divulgadas pela rede como um todo seja menor do que a vazão de mineração de todo Ethereum, a *mempool* torna-se vazia novamente, não obtendo transações atrasadas. A ideia do *Penny Attack* é que esta fila de transações pendentes esteja com um volume suficientemente grande, vide Figura 18, para que transações legítimas tenham seu tempo de mineração incrementado.

Figura 18 – *Mempool* contaminada com transações maliciosas.



Fonte: Elaborado pelo autor. (2020).

Dado a necessidade de estabelecer um grande número de transações para alcançar o objetivo final, que é a geração de lentidão na rede para inserir novas transações honestas, caso fosse mantido o padrão de taxas convencional da rede, ou seja, repetindo os valores de taxas utilizados pelos usuários para enviarem suas transações, o ataque tornar-se-ia custoso. O *Penny Attack*, no entanto, tem uma restrição orçamentária para sua execução,

sempre buscando o menor custo para o sucesso do ataque, utilizando-se das menores taxas praticadas por usuários não maliciosos da rede. Sendo assim, para a execução do *Penny Attack* é necessário que as contas envolvidas tenham saldo suficiente para executar as ações planejadas para o processo.

6.1 FERRAMENTAS PARA AVALIAÇÃO EXPERIMENTAL

Nesta seção, são descritas as ferramentas utilizadas para o desenvolvimento do ambiente experimental de avaliação da plataforma Ethereum.

6.1.1 Ganache

Ganache[1] é uma ferramenta destinada a facilitar o estabelecimento de uma rede Ethereum privada, atuando como uma forma de emular a plataforma usando a Clique como protocolo de consenso [Thakur et al. 2019]. O Ganache inicia rapidamente uma *blockchain* pessoal do Ethereum que você pode executar testes, comandos e inspecionar o estado enquanto controla como a *blockchain* opera. Transações e aplicações distribuídas podem ser implementadas em caráter de teste, sem implicações para a rede principal. Em resumo, permite executar simulações, além de verificar o estado da rede enquanto ela é controlada por um administrador.

É possível usar ferramentas convenientes para o controle avançado da forma de mineração e de um explorador de blocos, facilitando a visualização da estrutura de uma maneira mais amigável, como a forma visual do Ganache. No entanto, por questões de desempenho foi optado por seu formato em texto, interagindo diretamente com a ferramenta por meio de solicitações HTTP ou WebSocket.

Uma das grandes limitações do Ganache em seu estado atual é a falta de suporte ao serviço "TxPool" da Ethereum, que permite o monitoramento em tempo real da fila de transações pendentes na rede (*memory pool*). Apesar do fato mencionado, a ferramenta em questão se destaca da implementação tradicional da rede Ethereum devido à sua agilidade e maior possibilidade de controle e gerenciamento da estrutura da rede. Existem outras ferramentas com comportamento semelhante, como Puppeth[2], que permite a configuração de um protocolo de consenso personalizado, a visualização do *TxPool*, além da comunicação entre nós. Esta no entanto não foi adotada, pois o objetivo deste trabalho não é estabelecer uma rede distribuída, mas simular a *blockchain* de forma realista.

6.1.2 Web3.py

Os protocolos de comunicação utilizados pela plataforma Ethereum são complexos, portanto, foi adotado uma ferramenta para medeia-la, de forma a alcançar um nível de

[1] https://www.trufflesuite.com/ganache
[2] https://github.com/ethereum/go-ethereum/tree/master/cmd/puppeth

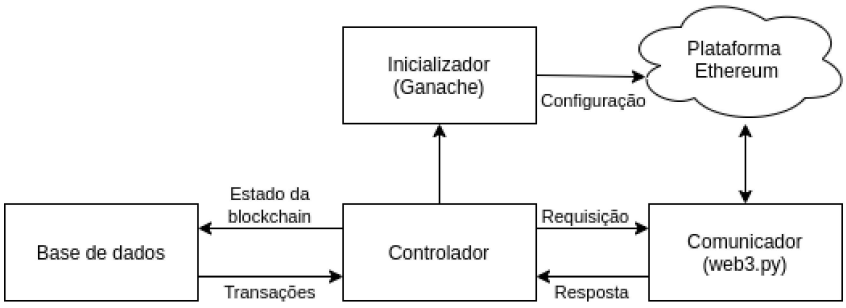
abstração maior para essa tarefa. Dentre as várias maneiras de se comunicar com a rede Ethereum, optou-se por fazê-lo através da interface de comunicação ou *driver* web3.py,[3] uma biblioteca que usa a linguagem Python e é composta por uma coleção de funções que permitem a interação com um nó Ethereum local ou remoto por meio de uma conexão HTTP ou IPC [Lee 2019].

Esta biblioteca foi inspirada no *web3.js*, um pacote disponibilizado via npm[4]. Para fazer uso do web3.py é necessário a instalação prévia por meio de um repositório de softwares do Python, o PyPi[5]. Com o estabelecimento de uma conexão com uma *blockchain* Ethereum através desta biblioteca, é possível de forma simples monitorar o comportamento e estado atual da rede. É permitido a execução de diversos comandos para interagir com a rede, tais como obter informações sobre o último bloco ou um bloco específico, buscar as transações que se encontram na *mempool*, a criação de *smart contracts*, obtenção de recibo de transações, verificar a velocidade de mineração executada pelo próprio nó, entre várias outras opções.

6.1.3 Arquitetura da simulação

Para executar a simulação proposta, é necessário organizar as ferramentas e entidades, estabelecendo uma arquitetura para seu funcionamento. A Figura 19 ilustra a arquitetura geral que foi projetada para a simulação da rede ETH. Essa arquitetura está organizada em quatro componentes: *(1) base de dados, (2) inicialização da rede, (3) controlador e (4) comunicador.* Esses componentes se comunicam para execução de transações na rede ETH. As setas indicam a dinâmica de funcionamento da arquitetura, dado o fluxo de dados entre os componentes e a rede.

Figura 19 – Visão geral da arquitetura para executar os experimentos.



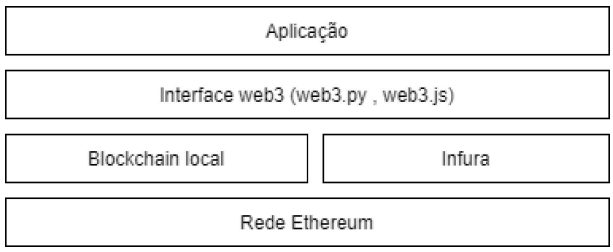
Fonte: Elaborado pelo autor. (2020).

A Figura 20 demonstra o modelo de camadas proposto nesta dissertação para simular a rede do Ethereum. Cada camada comunica somente com sua porção inferior e

[3] https://web3py.readthedocs.io/
[4] https://www.npmjs.com/package/web3
[5] https://pypi.org/project/web3/

superior. Sendo assim, pode-se observar que a camada de aplicação aqui proposta refere-se ao ”Controlador“ da Figura 19, o qual se comunica através de requisições e respostas com a interface web3. Esta interface, por sua vez, é capaz de fazer requisições à *blockchain*. No caso desta simulação, é utilizado uma *blockchain* local para simular o comportamento da plataforma Ethereum. Pode ser utilizado também para este fim o Infura[6], que consiste em uma série de ferramentas para facilitar a comunicação com a rede Ethereum, servindo como um serviço de ponte, caso o usuário não tenha interesse de manter uma cópia local da *blockchain*. A última camada, por sua vez, tem responsabilidades e regras de negócio particulares da rede, como por exemplo os protocolos de consenso.

Figura 20 – Modelo de camadas proposto para simular a plataforma Ethereum.



Fonte: Elaborado pelo autor. (2020).

6.1.4 Transações utilizadas

A primeira etapa da simulação ocorre com o componente base de dados, que fornece as transações a serem processadas nessa rede. O banco de dados pode prover transações reais do Ethereum, assim como transações sintéticas em que características como tarifa, valor, origem e destino são definidas de forma personalizadas. Para a base de transações, foram utilizados dois grupos: (i) transações reais contemporâneas ao momento da pesquisa para refletir o comportamento real da rede ETH, a partir da coleta descrita no Capítulo 4 e (2) transações sintéticas como forma de produzir comportamento condizente a um ataque à rede. Para as transações sintéticas foram configurados valores encontrados nas transações reais como forma de evitar previsibilidade de comportamento malicioso, modificando apenas o *gas price* para minimizar o custo do ataque em termos de tarifa. Especificamente, o *gas price* foi estabelecido em 1 GWei (10^{-9} Ether), que equivale ao 17o. percentil dos valores de *gas price* observados nas transações reais. Note que, dado a definição do ataque, seu estabelecimento envolve estabelecer um número de transações superior ao que normalmente ocorre na rede.

Como forma de oscilar o custo do ataque, mais uma vez visando evitar a previsibilidade do ataque, o *gas* foi definido de forma variável, isto é, um *gas* médio de $2,36 \times 10^5$ $\pm 0,25$. Além disso têm-se um desvio padrão de $5,66 \times 10^5$, valor moda de $7,91 \times 10^4$

[6] https://infura.io/

e máximo de $7,60 \times 10^6$. Como valor mínimo para este atributo é usado o menor valor usado no sistema, definido por $2,1 \times 10^4$.

Em suma, para a simulação foram utilizadas transações que aparecem na rede real no dia 16/05/2019 entre 08:22 e 08:37, onde são replicados seus valores de *gas* e valor transacionado, juntamente com transações geradas sinteticamente. Além disso cada endereço de usuário remetente e destinatário é mapeado em um novo usuário pertencente à rede do ambiente experimental. É importante perceber que, na simulação, foram incluídas transações na *mempool* de acordo com o atributo de tempo observado nos dados, enquanto o tempo de espera é definido pela dinâmica da simulação.

6.1.5 Configuração inicial

Dadas as transações, a próxima etapa utiliza a ferramenta Ganache, a qual é responsável por prover o serviço do Ethereum. Assim, foi configurada nesta ferramenta uma *blockchain* com tempo entre blocos de 13 segundos, semelhante ao comportamento atual da rede ETH[7]. Foram inicializadas 2000 contas nesta nova rede, para que cada endereço único na base de dados possa ser mapeada em um endereço único na simulação. Cada conta inicia-se contendo um saldo de 100 Ether, de forma a não limitar a execução de transações, independentemente das transações escolhidas para serem replicadas na rede privada. Foi configurado também um valor máximo de *gas* para o bloco de 1×10^6, de forma a replicar as características da *blockchain* real[8]. Dadas essas configurações, o controle dispara um comando para que o Ganache gere uma nova instância de rede Ethereum privada.

Como pode ser observado na Figura 19, o componente controle é central na arquitetura, e nos permite monitorar todas as etapas das simulações. Em particular, o controle faz o mapeamento de usuários extraídos das transações reais em usuários da rede inicializada na simulação, além de definir uma frequência de injeção de transações por período, monitoramento da fila de transações (mempool) e a solicitação da criação de novas transações e contratos inteligentes. O componente Comunicador, por sua vez, utiliza a interface *web3.py* para requisitar a execução de transações à rede, assim como fornecer a resposta dessas execuções.

Dada a visão geral da arquitetura e transações, é descrito a seguir o comportamento configurado no controlador para as simulações. Foi executada a simulação em um período de 15 minutos da rede para simular o ataque, onde o ataque ocorre entre o quinto e o décimo minuto. A injeção de carga na rede foi dívida em três partes de 5 minutos. A primeira parte consiste em injetar dados na rede acima da média de transações por segundo para obter, além de transações suficientes para estabelecer um bloco, transações

[7] https://etherscan.io/chart/blocktime
[8] https://etherscan.io/chart/gaslimit

que podem compor a fila de transações pendentes. Isso equivale a 16,7% da taxa de chegada das transações reais. Em uma segunda etapa, as transações de comportamento normal serão inseridas simultaneamente com as transações de comportamento malicioso geradas artificialmente, seguindo as definições estabelecidas aqui sobre o *Penny Attack*. Em um terceiro momento, a rede terá apenas a inserção de transações da rede principal em fluxo proporcional ao convencional, para verificar como a rede se comporta após o estabelecimento do *Penny Attack*.

Para executar experimentos sobre o *Penny Attack* foram utilizados dados de transações da rede ETH que foram apresentados previamente nesta dissertação no Capítulo 4. Os atributos mostrados na Tabela 2 são os adotados aqui, que foram suficientes para reproduzirmos o comportamento da rede ETH no experimento. Com a simulação, calculamos o momento de criação da transação e o momento em que um bloco foi inserido contendo esta transação. A diferença entre esses valores é referida aqui como tempo de espera.

Foram reproduzidos três cenários em nossas simulações da rede: (i) um fluxo normal com 2000 transações Ethereum, (ii) *Penny Attack* com as mesmas transações do fluxo normal, mas adicionando 10% do fluxo de transações mal-intencionadas e (iii) transações de 20 de dezembro de 2017, onde uma alta aderência à rede foi observado, aqui denominada de *Ethereum Boom*.

6.2 ABORDAGENS PARA DETECTAR O ATAQUE

A partir da definição do *Penny Attack*, percebe-se que há necessidade de compreender diversas variáveis que concernem a rede da blockchain para distinguir comportamentos idôneos na rede de comportamentos maliciosos. Este aprendizado seria altamente complexo para ser feito por uma pessoa, visto que há um grande volume de dados, o que impactaria também no tempo de resposta em momentos de ataque. Uma tecnologia capaz de reconhecer diferentes padrões em grandes volumes de dados é o aprendizado de máquina, que já é aplicado em diversos setores [Paolanti e Frontoni 2020] tais como de saúde [Kaur e Kumari 2020], engenharia [Rohilla, Chakraborty e Kumar 2020], entre outros. Sendo assim, foi optado pela aplicação de diversas técnicas de aprendizado de máquina em busca de uma forma distinguir transações maliciosas das demais.

Classificadores foram adotados neste trabalho para determinar se uma transação tem comportamento malicioso ou não. Em aprendizado de máquina, a classificação é considerada como um aprendizado supervisionado para inferir uma função que relacione variáveis independentes a rótulos pré-definidos (supervisão/classe) de um conjunto de dados de treinamento rotulados. Um modelo de classificação é inferido a partir de dados de treinamento e pode ser usado para predizer o rótulo de dados não observados durante o treino. Existem vários classificadores, e aqui foram utilizados alguns comumente encontrados na literatura: Árvore de Decisão, *Random Forest* [Breiman 2001], KNN [Liao e

Tabela 5 – Definição de métricas usadas para avaliação dos modelos.

Acurácia	Precisão (P)	Recall (R)	F_β-score	F_1-score
$\frac{TP+TN}{TP+TN+FP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{(1+\beta^2)\cdot R\cdot P}{\beta^2\cdot R+P}$	$\frac{TP}{TP+0.5\cdot(FP+FN)}$

Fonte: Elaborado pelo autor. (2020).

Vemuri 2002], SVM [Lorena e Carvalho 2007] e Naïve Bayes [Rish et al. 2001]. Os métodos aqui escolhidos são populares em aprendizado de máquina e amplamente usados em diversos temas, não somente na ciência da computação [Udaiyakumar et al. 2020, Princy et al. 2020, Sambasivam, Amudhavel e Sathya 2020]. Além disso, foi considerado também o uso de comitês de classificadores, os quais definem conjuntamente a classificação, baseado na decisão da comunidade definida.

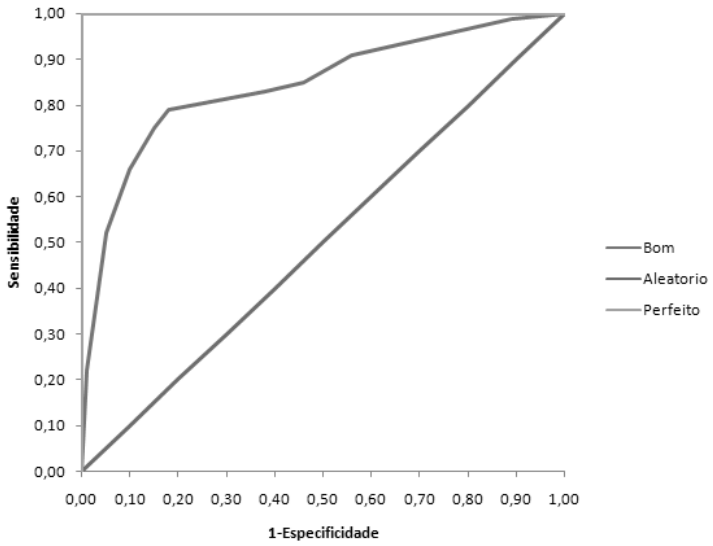
Os dados empregados nos classificadores foram divididos em dois grupos: um grupo de treino e um grupo de testes. Os dados de teste (70% das transações) são utilizados para criar os modelos classificadores. Por outro lado, os modelos gerados são avaliados utilizando as 30% outras transações. Os modelos são avaliados utilizando de métricas como acurácia, precisão, *recall*, F_1-score, F_β-score com $\beta = 2$, número de verdadeiros positivos (TP) e negativos (TN), e área da curva ROC (AUC-ROC), definidos na Tabela 5.

F-scores são a combinação da precisão e *recall*. Em particular, F_1-score é a média harmônica entre eles. O valor de β pode ser usado para ponderar a importância da precisão e do *recall*. A importância da precisão é aumentada quando $0 < \beta < 1$. No entanto, estamos interessados em valores maiores de *recall* ($\beta > 1$) e, então, aqui usamos $\beta = 2$. O *recall* é a proporção de instâncias realmente positivas as quais são corretamente classificadas como positivas. A precisão denota a proporção de instâncias classificadas como positivas que são de fato positivas [Powers 2011]. A acurácia, por sua vez, é a proporção do número de predições corretas e o número total de predições feitas.

A curva ROC, como exemplificada na Figura 21, representa a taxa de verdadeiros positivos em oposição a taxa de falsos positivos considerando diferentes limites. Sendo assim, temos no eixo horizontal a especificidade, que representa a proporção de verdadeiros negativos, ou seja, a capacidade do classificador em predizer corretamente instâncias negativas. Já no eixo vertical encontra-se a sensibilidade, que é a capacidade do classificador em predizer corretamente instâncias positivas. Como resultado, a métrica AUC-ROC pode representar a probabilidade de um classificador classificar uma instância positiva escolhida aleatoriamente mais alta que uma negativa selecionada aleatoriamente. Esta é uma das métricas mais comumente usadas para avaliar o desempenho dos algoritmos de aprendizado de máquina em casos de classes desbalanceadas.

A árvore de decisão é uma ferramenta de suporte a decisão que usa uma árvore, ou modelo de decisões, e suas possíveis consequências. Cada nó interno representa o teste de

Figura 21 – Exemplo de curva ROC.



Fonte: Elaborado pelo autor. (2020).

um atributo, e cada ramo representa o resultado deste teste, e cada nó folha representa a decisão por um rótulo, ou seja, a decisão feita após computar alguns dos atributos. O caminho da raiz até a folha representa as regras de classificação estabelecidas pelo modelo em questão. O processo de geração do modelo envolve a criação da árvore, decidindo quais atributos escolher e quais condições a serem utilizadas para os ramos. Devido ao comportamento de crescimento arbitrário da árvore, é comum a necessidade de poda da mesma, ou seja, redução da quantidade de níveis para melhor divisão dos grupos e garantir a inteligibilidade do modelo.

Outra técnica empregada neste trabalho é o *Random Forest*, o qual consiste em um grande número de árvores de decisão que operam em formato cooperativo. Cada árvore no *Random Forest* utiliza dos atributos definidos como relevantes para tomada de decisão para mapeá-los em uma classe. Uma das possíveis decisões do modelo como um todo é baseado na classe mais frequente entre as decisões das árvores envolvidas no modelo. Para este modelo foi predefinido a altura máxima das árvores em 3, além de estabelecer o número de árvores a serem geradas em 100. O critério de otimização escolhido para este modelo foi o "gini".

O KNN, do inglês *K-Nearest Neighbors*, é um modelo no qual é utilizado a ideia de similaridade, sendo esta similaridade calculada com diferentes funções, tais como de distância entre dois pontos. Essa distância pode ser a distância euclidiana, distância de *Manhattan*, distância de *Minkowski* ou distância de *Hamming*, por exemplo. A similaridade é utilizada para realizar a classificação. A letra "K" do nome, refere-se ao número de vizinhos, calculados pela função escolhida, necessários para realizar a classificação. Sendo assim, assumindo $k = 1$, por exemplo, a classe a ser escolhida pelo método é a mesma de

seu vizinho mais próximo. Uma das principais desvantagens deste método esta exatamente na definição do modelo, visto que há necessidade de calcular a distância entre todos os pontos para poder realizar a classificação, logo, sua complexidade é alta, impactando na escalabilidade para grandes volumes de dados.

O SVM, do inglês *Support Vector Machine*, que significa Máquina de vetores de suporte. O objetivo deste algoritmo é encontrar um hiperplano em um espaço n-dimensional, sendo n o número de atributos utilizados, que consegue classificar os dados, que são representados por um ponto neste espaço. Para separar duas ou mais classes de dados existem várias possibilidades de hiperplanos que podem ser escolhidos, sendo o objetivo deste modelo encontrar o hiperplano que maximiza a distância entre as classes.

Por último temos o Naïve Bayes, um classificador probabilístico o qual é baseado no teorema de Bayes

$$P(\theta|e) = P(\theta)\frac{P(e|\theta)}{P(e)}, \tag{6.1}$$

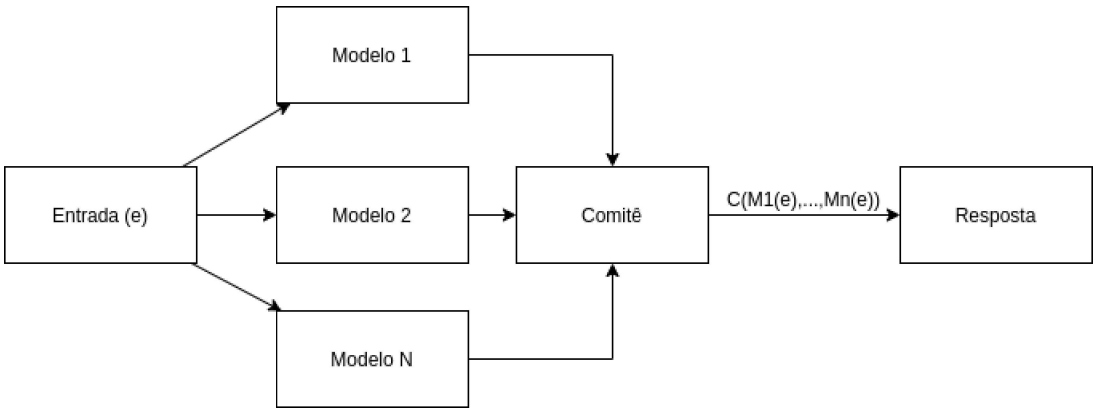
onde θ é a hipótese a ser testada, e é a nova instância de dados e P($\theta|e$) é a probabilidade da hipótese θ, dada a informação e. O classificador Naïve Bayes combina este modelo probabilístico com uma regra de decisão. Uma regra comum é escolher a hipótese mais provável, sendo assim escolhendo qual classe é mais provável de uma instância a ser classificada pertencer. Uma das limitações deste modelo é a suposição de que os atributos utilizados são independentes, sendo assim o Naïve Bayes assume que a presença de uma característica particular em uma classe não está relacionada com a presença de qualquer outro recurso, logo todas os atributos considerados pelo classificador contribuem de forma independente para a probabilidade de pertencer a uma classe. Pode-se dividir o modelo entre alguns tipos, tais como o Naïve Bayes multinomial, o Bernoulli Naïve Bayes, que torna-se útil quando a classificação é binária e o Naïve Bayes gaussiano, que assume uma distribuição normal dos dados.

Além dos modelos indicados, foram empregados também um conjunto de diferentes classificadores, definindo a classificação não apenas com base em um único algoritmo mas com base na decisão conjunta de um comitê de classificadores [Niranjan et al. 2019]. Essa abordagem permite que as vantagens de diversos tipos de classificadores sejam combinadas, buscando eliminar seus pontos fracos e, em seguida, levar a uma solução mais robusta. Dado um conjunto de modelos de classificação $\{M_1, M_2, ..., M_n\}$, para uma matriz de características em que cada linha representa uma transação, o qual daremos o nome de e, e cada coluna representa um atributo de uma transação qualquer, cada classificador classifica individualmente cada linha desta matriz, gerando uma resposta binária: transação maliciosa ou não, como ilustrado na Figura 22.

As predições para cada instância são feitas com base num critério de votação definido ao criar este comitê, o qual pode ser definido como: (i) voto majoritário (*hard*) e (ii) média de confiança (*soft*). O primeiro modo parte do princípio simples de que a classe

de resposta mais votada pelos modelos participantes do comitê será a classe escolhida como resposta. A segunda opção tem um funcionamento diferente, partindo sua decisão da certeza que os modelos têm de que uma transação é pertencente a uma classe e, assim, apresenta uma resposta com base numa média. Todos os classificadores supracitados (Árvore de Decisão, *Random Forest*, KNN, SVM e Naïve Bayes) foram utilizados para o estabelecimento dos classificadores *hard* e *soft*. Para a geração do modelo de votação por média de confiança é possível definir pesos diferentes para que um classificador tenha, além do peso por confiança, mudanças no impacto de seu voto. No modelo gerado nesta dissertação foram utilizados valores iguais para os classificadores participantes do comitê.

Figura 22 – Modelo do comitê de classificadores.



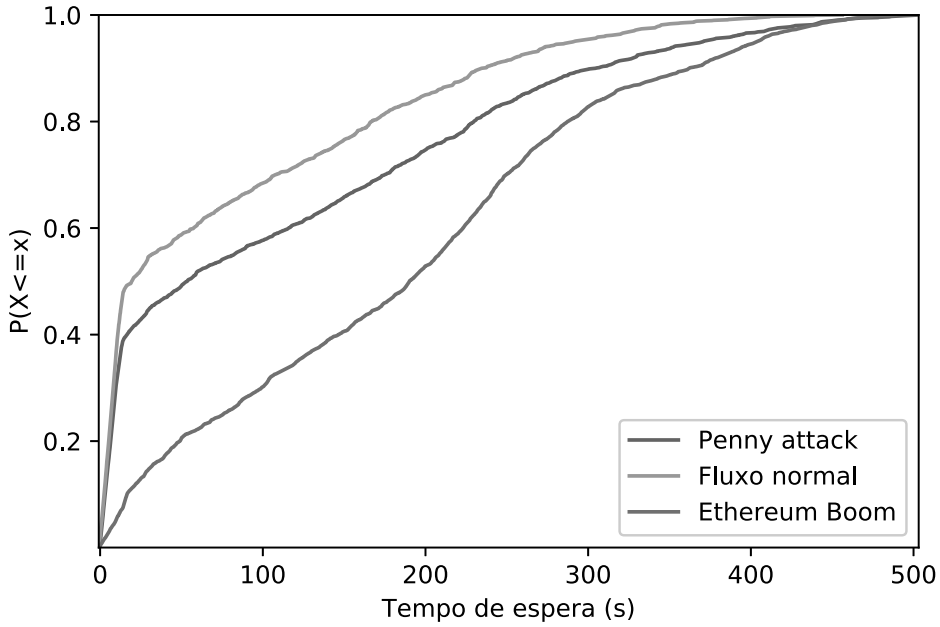
Fonte: Elaborado pelo autor. (2020).

6.3 RESULTADOS DA SIMULAÇÃO

Nesta subseção são avaliados os impactos do *Penny Attack* em uma rede Ethereum simulada. São comparados os tempos de espera das transações para os 3 cenários previamente descritos.

A Figura 23 mostra a distribuição de probabilidades acumulada desses tempos para cada cenário. As curvas que crescem mais lentamente indicam os cenários onde o tempo de espera é maior. À primeira vista, o cenário Ethereum Boom apresenta maior tempo de espera, quando comparado aos outros cenários. Por outro lado, a rede Ethereum em fluxo normal apresenta os menores tempos de pendência. Por exemplo, 60% de todas as transações de um sistema Ethereum em funcionamento normal (onde não notamos um ataque) não apresentam tempos de espera superiores a 19,5 segundos no 50º percentil. Quando o sistema está sob ataque, praticamente 50% de todas as transações apresentam tempos pendentes superiores a 50 segundos. Nessa mesma situação, no cenário Boom, os tempos pendentes são praticamente 4 vezes maiores e praticamente 50% de todas as transações experimentam tempos pendentes superiores a 200 segundos.

Figura 23 – Função de distribuição acumulada do tempo de espera na simulação.



Fonte: Elaborado pelo autor. (2020).

Adicionalmente, a Tabela 6 mostra as estatísticas relacionadas ao tempo de espera nos três cenários. Pode-se observar que o ataque aumentou o tempo médio de espera em 42,16%, enquanto o período de maior uso da rede aumentou esse valor em 131,90%. Outro ponto observado é que, usando um intervalo de confiança de 95%, a média máxima do fluxo normal e a média mínima do *Penny Attack* não se encontram. Foi utilizado o teste de Kruskal-Wallis para medir a significância da distinção entre o tempo de espera em momentos de fluxo normal e momentos de ataque, que resultaram em um p-valor muito próximo de zero ($1,14108 \cdot 10^{-59}$). Isso nos permite concluir que o ataque realmente afetou o tempo de espera pelas transações na rede simulada neste trabalho.

Em suma, fica evidente a partir dos cenários simulados que o tempo de espera é maior tanto para cenários com *Penny Attack* quanto para momentos de alta demanda de uso da rede. Mas curiosamente, o ataque não obteve impacto no tempo de espera superior ao observado na rede durante o Boom do Ethereum. Logo, observa-se que é um desafio para os mantenedores e administradores da rede distinguir um *Penny Attack* de uma situação onde a rede realisticamente tem um aumento no número de transações.

6.4 ANÁLISE DOS MODELOS DE APRENDIZADO DE MÁQUINA

Nesta subseção é avaliada a eficiência das técnicas de aprendizado de máquina para detectar o *Penny Attack*. Para isso, são apresentados os resultados dos diferentes modelos

Tabela 6 – Detalhes do tempo de espera.

	Média	Intervalo de Confiança (95%)	Desvio Padrão	Moda
Normal	79,56	75,15 - 83,96	100,37	19,55
Penny attack	113,11	107,96 - 118,27	126,06	53,54
Ethereum boom	184,50	180,69 - 188,32	122,98	190,09

Fonte: Elaborado pelo autor. (2020).

classificadores e posteriormente as métricas obtidas pelos classificadores são discutidas.

Tabela 7 – Resultados obtidos pelos métodos de aprendizado de máquina usando diferentes abordagens para balancear os dados.

Balanceamento	Classificador	Acurácia	Precisão	Recall	F_1-score	F_β-score	TP	TN	AUC-ROC
Undersampling	Árvore de Decisão	0,891818	0,454756	**0,984925**	0,622222	0,798696	**196**	1766	0,933742
	Random Forest	**0,915000**	**0,515957**	0,974874	**0,674783**	**0,827645**	194	**1819**	**0,941960**
	KNN	0,686364	0,215527	0,934673	0,350282	0,560579	186	1324	0,798171
	SVM Gaussiano	0,614091	0,178854	0,909548	0,298927	0,500553	181	1170	0,747128
	Naïve Bayes	0,871364	0,411392	0,979899	0,579495	0,767717	195	1722	0,920235
	Comitê *Soft*	0,907273	0,493606	0,969849	0,654237	0,812974	193	1803	0,935449
	Comitê *Hard*	0,910000	0,501292	0,974874	0,662116	0,819949	194	1808	0,939211
Oversampling	Árvore de Decisão	0,911818	0,506460	**0,984925**	0,668942	**0,828402**	**196**	1810	**0,944736**
	Random Forest	**0,959545**	**0,757009**	0,814070	**0,784504**	0,801980	162	**1949**	0,894042
	KNN	0,872727	0,398496	0,798995	0,531773	0,665272	159	1761	0,839527
	SVM Gaussiano	0,685909	0,216590	0,944724	0,352390	0,564904	188	1321	0,802447
	Naïve Bayes	0,860909	0,392354	0,979899	0,560345	0,754060	195	1699	0,914487
	Comitê *Soft*	0,915909	0,518919	0,964824	0,674868	0,823328	192	1823	0,937934
	Comitê *Hard*	0,913182	0,510582	0,969849	0,668977	0,821976	193	1816	0,938698
SMOTE	Árvore de Decisão	0,912727	0,509091	**0,984925**	0,671233	**0,829805**	**196**	1812	**0,945236**
	Random Forest	**0,953182**	**0,703390**	0,834171	**0,763218**	0,804264	166	**1931**	0,899594
	KNN	0,868182	0,388206	0,793970	0,521452	0,656692	158	1752	0,834766
	SVM Gaussiano	0,690455	0,218458	0,939698	0,354502	0,565981	187	1332	0,802683
	Naïve Bayes	0,860909	0,391039	0,964824	0,556522	0,745921	192	1702	0,907699
	Comitê *Soft*	0,913636	0,512064	0,959799	0,667832	0,816938	191	1819	0,934422
	Comitê *Hard*	0,913182	0,510695	0,959799	0,666667	0,816239	191	1818	0,934172

Fonte: Elaborado pelo autor. (2020).

A Tabela 7 mostra o resultado da aplicação de diferentes técnicas de aprendizado de máquina sem balanceamento e com subamostragem, sobre-amostragem, e a técnica de SMOTE como forma de balanceamento.

O *Random Forest* teve melhor desempenho do que as outras técnicas aplicadas para todas estratégias de balanceamento de dados utilizadas em relação a acurácia, precisão, F_1-score e valores de TN. Por exemplo, quando a técnica de *oversampling* e utilizada, o *Random Forest* consegue acurácia, precisão, F_1-score e TN com valores maiores que 0,95, 0,75, 0,78 e 1949, respectivamente.

Por outro lado, a Árvore de Decisão apresentou resultados melhores para todas as estratégias de balanceamento de dados empregadas no que diz respeito ao *recall*, F_β-score TP e AUC-ROC. Ocorreram exceções para F_β-score e AUC-ROC, quando o *undersampling* é aplicado, onde o *Random Forest* alcançou valores melhores nessas métricas. O melhor valor de *recall*, F_β-score, TP e AUC-ROC foram obtidos pela Árvore de Decisão e são

maiores que 0,98, 0,82, 196 e 0,94 respectivamente. No geral, pode-se concluir que a Árvore de Decisão pode identificar transações maliciosas melhor do que outras abordagens.

No geral, tanto o comitê *soft* quanto *hard* tiveram bons resultados quando comparados a outros classificadores. Apesar desse bom desempenho, nenhum dos dois comitês obteve o melhor resultado em uma métrica específica. Outro classificador, no entanto, teve os piores resultados gerais, em todos os cenários: o SVM.

Dentre os atributos identificados como relevante para a tomada de decisão nos modelos destacam-se o *gas price* e posteriormente o *gas*. Este tipo de decisão é esperada, tendo em vista que são os principais requisitos para o estabelecimento do ataque. Fatores como remetente e destinatário não tiveram relevância para a classificação, tendo em vista que, dado a atual tecnologia, torna-se fácil criar um ataque suficientemente distribuído, de forma a não concentrar o ataque em poucos endereços. O valor transacionado também não teve grande relevância para a classificação.

Em suma, observa-se que os modelos classificadores *Random Forest* e Árvore de Decisão se destacam dentre os classificadores apresentados nesta dissertação. Ambos conseguiram uma AUC-ROC maior que 0,94, que aliado com *recall* maior que 0,98 mostra que os modelos são eficazes para detectar este tipo de comportamento. Esta abordagem, no entanto, não especifica qual solução será tomada pela rede para sanar o problema, tendo seu foco somente em detectar o *Penny Attack*. Uma das possibilidades de utilização dos modelos classificadores na rede Ethereum é o constante monitoramento da fila de transações (*mempool*), de forma a identificar precocemente transações maliciosas, e em caso de concretização do ataque, alertar sobre a detecção para que possam ser tomadas medidas de governança como um *soft fork* [Kiffer, Levin e Mislove 2017], alterando manualmente o estado da rede de forma a que os comportamentos maliciosos não sejam incluídos no novo estado gerado da *blockchain*.

Neste capítulo foram criadas simulações da rede Ethereum com intuito de avaliar o comportamento da plataforma durante um *Penny Attack*. Foi proposto o uso de técnicas de aprendizado de máquina para detectar o ataque. A partir das simulações baseadas em dados reais, é mostrado que o *Penny Attack* tem capacidade de aumentar o tempo de espera no Ethereum. Além disso, percebe-se que a alta demanda de transações por segundo, como no *Ethereum Boom* pode acarretar em tempo de espera maior do que em ataques DoS. Por último, é externado que a aplicação de aprendizado de máquina é possível para detectar o *Penny Attack* com desempenho considerável. A Árvore de Decisão e *Random Forest*, em particular, alcançaram os melhores desempenhos na maioria das métricas utilizadas para avaliar os classificadores.

7 CONCLUSÕES

A presente dissertação explorou a plataforma Ethereum, a segunda maior criptomoeda em atividade, no que tange a capitalização de mercado. Foram analisados dados da rede ETH do Ethereum sob a ótica da tarifação de transações e *smart contracts*. Pode-se dividir as contribuições desta dissertação em duas análises dessa rede. A primeira análise aqui conduzida busca a verificação de um senso comum, no qual acredita-se que transações com maiores taxas serão mineradas com maior prioridade na rede.

Para poder conduzir essa pesquisa, tem-se como contribuição uma metodologia para captura de dados da rede a partir de duas APIs que permitem a consulta de dados Ethereum. Essa consulta nos permite obter tanto transações pendentes (*mempool*) quanto transações e *smart contracts* que adentraram a blockchain.

As análises da primeira parte desse trabalho foram realizadas sob todos os dados coletados e também subconjunto dos dados, baseado na seleção de transações com valores de atributos específicos. Como mencionado, o objetivo dessa primeira parte foi analisar a relação entre valor das taxas e o tempo em que transações ficam pendentes. Logo, métodos quantitativos como coeficientes de correlações de Pearson e Spearman foram utilizados, além do algoritmo de clusterização OPTICS para realizar essa análise em subconjuntos de dados. Baseado nas evidências dessas análises foram obtidas as seguintes conclusões:

- transações e *smart contracts* possuem comportamento semelhante em relação ao tempo de espera e valores de tarifação, especificamente o *gas price*;
- não há evidências de correlação entre atributos derivados de taxas e o tempo de espera. Portanto, o pagamento de taxas maiores não leva necessariamente à confirmação mais rápida de transações na rede Ethereum, o que torna a predição do tempo de espera baseado apenas no valor das taxas inadequada;
- o algoritmo de agrupamento OPTICS consegue identificar diferentes comportamentos dos usuários na rede ETH, o que permite uma análise específica para cada caso.

A segunda parte deste trabalho foca na segurança da rede Ethereum. Foram analisados os efeitos, em termos de atraso na aprovação de transações, de um tipo de DDoS, aqui denominado de *Penny Attack*. Adicionalmente foi proposto e avaliado a identificação desse ataque baseado em reconhecimento de padrões via técnicas de aprendizagem de máquina. Para isto, foi estabelecido uma arquitetura para possibilitar a simulação da rede Ethereum de forma realista para então injetar transações maliciosas como ocorreria em um momento onde ocorre *Penny Attack* na plataforma.

Um *Penny Attack* com 10% de transações maliciosas tem um impacto em aumentar o tempo pendente das transações, com significância estatística, sendo utilizado o teste

de Kruskal-Wallis. Contudo, momentos de grande movimento na rede, como no final de 2017, tiveram maior impacto de lentidão do que o ataque aqui estabelecido. O que torna desafiante diferenciar um *Penny Attack* de picos de transações verdadeiras na rede.

Métodos de aprendizagem de máquina são capazes de identificar as transações adequadamente com métricas de AUR-ROC maior que 0,945 e F_β maior que 0,82, o que mostra que é possível empregar esta tecnologia como forma de contingência de danos a rede Ethereum ao detectar precocemente o ataque.

As contribuições desse estudo foram publicadas em congressos internacionais [Sousa et al. 2019, Sousa et al. 2021, Oliveira et al. 2021] (LANOMS e SOCCA), um periódico internacional [Sousa et al. 2020a] (IJNM), e um *workshop* de referência da comunidade brasileira de pesquisadores blockchain [Sousa et al. 2020b] (WBLOCKCHAIN), as quais deram base para esta obra. Como continuação do trabalho, pretende-se abranger o uso de técnicas de aprendizado de máquina para a detecção de outros tipos de ataque à principal rede do Ethereum. Outra possibilidade de extensão deste estudo é a utilização de *Deep learning* para identificação de diversos ataques de forma simultânea.

REFERÊNCIAS

[Alonso 2020]ALONSO, K. M. *Zero to Monero*. 2020.

[Angelis et al. 2018]ANGELIS, S. D. et al. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. 2018.

[Ankerst et al. 1999]ANKERST, M. et al. OPTICS: Ordering points to identify the clustering structure. In: . [S.l.]: ACM Press, 1999. p. 49–60.

[Antonio Pierro et al. 2020]Antonio Pierro, G. et al. Are the gas prices oracle reliable? a case study using the ethgasstation. In: *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. [S.l.: s.n.], 2020. p. 1–8.

[Atzei, Bartoletti e Cimoli 2017]ATZEI, N.; BARTOLETTI, M.; CIMOLI, T. A survey of attacks on ethereum smart contracts (sok). In: SPRINGER. *International Conference on Principles of Security and Trust*. [S.l.], 2017. p. 164–186.

[Bagay 2020]BAGAY, D. x. *Procedia Computer Science*, Elsevier, v. 169, p. 187–191, 2020.

[Bano et al. 2017]BANO, S. et al. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[Baqer et al. 2016]BAQER, K. et al. Stressing out: Bitcoin "stress testing". In: SPRINGER. *International Conference on Financial Cryptography and Data Security*. [S.l.], 2016. p. 3–18.

[Bareis, Angelo e Salzer 2020]BAREIS, M.; ANGELO, M. D.; SALZER, G. Functional differences of neo and ethereum as smart contract platforms. In: SPRINGER. *International Congress on Blockchain and Applications*. [S.l.], 2020. p. 13–23.

[Bartoletti et al. 2020]BARTOLETTI, M. et al. Dissecting ponzi schemes on ethereum: identification, analysis, and impact. *Future Generation Computer Systems*, Elsevier, v. 102, p. 259–277, 2020.

[Berg, Davidson e Potts 2020]BERG, C.; DAVIDSON, S.; POTTS, J. Proof of work as a three-sided market. *Frontiers in Blockchain*, Frontiers, v. 3, p. 2, 2020.

[Böhme et al. 2015]BÖHME, R. et al. Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, v. 29, n. 2, p. 213–38, 2015.

[Bradbury 2015]BRADBURY, D. In blocks we trust [bitcoin security]. *Engineering & Technology*, IET, v. 10, n. 2, p. 68–71, 2015.

[Breiman 2001]BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

[Buterin 2016]BUTERIN, V. Gas cost changes for io-heavy operations. *URL https://github.com/ethereum/EIPs/blob/master/EIPS/eip-150.md*, n. 150, 2016.

[Buterin 2016]BUTERIN, V. Transaction spam attack: Next steps. *URL https://blog.ethereum.org/2016/09/22/transaction-spam-attack-next-steps*, 2016.

[Buterin 2017]BUTERIN, V. *A Quick Gasprice Market Analysis*. 2017. `https://vitalik.ca/general/2017/12/14/gas_analysis.html`. Último acesso: 12/05/2020.

[Buterin 2017]BUTERIN, V. A state clearing faq. *URL https://www.reddit.com/r/ethereum/comments/5es5g4/a_state_clearing_faq/?st=iw2e1mwo&sh=fa* n. 150, 2017.

[Buterin et al. 2013]BUTERIN, V. et al. Ethereum white paper. *GitHub repository*, p. 22–23, 2013.

[Chan e Olmsted 2017]CHAN, W.; OLMSTED, A. Ethereum transaction graph analysis. In: *International Conference for Internet Technology and Secured Transactions (ICITST). IEEE*. [S.l.: s.n.], 2017. p. 498–500.

[Chen et al. 2020]CHEN, H. et al. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 53, n. 3, p. 1–43, 2020.

[Chen et al. 2020]CHEN, H. et al. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, v. 53, n. 3, p. 1–43, 2020.

[Chen, Narwal e Schultz 2019]CHEN, M.; NARWAL, N.; SCHULTZ, M. Predicting price changes in ethereum. *International Journal on Computer Science and Engineering (IJCSE) ISSN*, p. 0975–3397, 2019.

[Chen et al. 2017]CHEN, T. et al. Under-optimized smart contracts devour your money. In: *International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE*. [S.l.: s.n.], 2017. p. 442–446.

[Chen et al. 2017]CHEN, T. et al. An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks. In: SPRINGER. *International Conference on Information Security Practice and Experience*. [S.l.], 2017. p. 3–24.

[Chen et al. 2018]CHEN, T. et al. Understanding ethereum via graph analysis. In: *IEEE INFOCOM*. [S.l.: s.n.], 2018. p. 1484–1492.

[Chervinski, Kreutz e Yu 2019]CHERVINSKI, J. O. M.; KREUTZ, D.; YU, J. Floodxmr: Low-cost transaction flooding attack with monero's bulletproof protocol. *IACR Cryptology ePrint Archive*, v. 2019, p. 455, 2019.

[Decker e Wattenhofer 2013]DECKER, C.; WATTENHOFER, R. Information propagation in the bitcoin network. In: IEEE. *IEEE P2P 2013 Proceedings*. [S.l.], 2013. p. 1–10.

[Denisova, Mikhaylov e Lopatin 2019]DENISOVA, V.; MIKHAYLOV, A.; LOPATIN, E. Blockchain infrastructure and growth of global power consumption. *International Journal of Energy Economics and Policy*, Econjournals, v. 9, n. 4, p. 22, 2019.

[Dwork e Naor 1992]DWORK, C.; NAOR, M. Pricing via processing or combatting junk mail. In: SPRINGER. *Annual International Cryptology Conference*. [S.l.], 1992. p. 139–147.

[Easley, O'Hara e Basu 2019]EASLEY, D.; O'HARA, M.; BASU, S. From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics*, Elsevier, 2019.

[Ester et al. 1996]ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (KDD). AAAI Press*. [S.l.: s.n.], 1996. p. 226–231.

[Etherchain 2020]ETHERCHAIN. *Etherchain - The Ethereum Blockchain Explorer*. 2020. `https://etherchain.org`.

[Etherscan 2020]ETHERSCAN. *What is Etherscan?* 2020. `https://etherscancom.freshdesk.com/support/solutions/articles/35000022140-what-is-etherscan-`.

[Eyal e Sirer 2018]EYAL, I.; SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, ACM, v. 61, n. 7, p. 95–102, 2018.

[Gautham 2016]GAUTHAM. Ethereum network comes across yet another dos attack. *URL https://www.newsbtc.com/2016/09/23/ethereum-dao-attack-attack-platforms-credibility*, 2016.

[Gencer et al. 2018]GENCER, A. E. et al. Decentralization in bitcoin and ethereum networks. In: SPRINGER. *International Conference on Financial Cryptography and Data Security*. [S.l.], 2018. p. 439–457.

[Gerard 2017]GERARD, D. *Attack of the 50 foot blockchain: Bitcoin, blockchain, Ethereum & smart contracts*. [S.l.]: David Gerard, 2017.

[Gramoli 2017]GRAMOLI, V. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, Elsevier, 2017.

[Han, Kamber e Pei 2011]HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[Hao et al. 2020]HAO, W. et al. Towards a trust-enhanced blockchain p2p topology for enabling fast and reliable broadcast. *IEEE Transactions on Network and Service Management*, IEEE, 2020.

[Hitam e Ismail 2018]HITAM, N.; ISMAIL, A. Comparative performance of machine learning algorithms for cryptocurrency forecasting. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 11, n. 3, p. 1121–1128, 2018.

[Houy 2014]HOUY, N. The bitcoin mining game. *Available at SSRN 2407834*, 2014.

[Houy 2014]HOUY, N. The economics of bitcoin transaction fees. *GATE WP*, v. 1407, 2014.

[Jain 1990]JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990.

[Jakobsson e Juels 1999]JAKOBSSON, M.; JUELS, A. Proofs of work and bread pudding protocols. In: *Secure information networks*. [S.l.]: Springer, 1999. p. 258–272.

[Kaur e Kumari 2020]KAUR, H.; KUMARI, V. Predictive modelling and analytics for diabetes using a machine learning approach. *Applied computing and informatics*, Emerald Publishing Limited, 2020.

[Kepser 2004]KEPSER, S. A simple proof for the turing-completeness of xslt and xquery. In: CITESEER. *Extreme Markup Languages®*. [S.l.], 2004.

[Kiayias, Miller e Zindros 2020]KIAYIAS, A.; MILLER, A.; ZINDROS, D. Non-interactive proofs of proof-of-work. In: SPRINGER. *International Conference on Financial Cryptography and Data Security*. [S.l.], 2020. p. 505–522.

[Kiffer, Levin e Mislove 2017]KIFFER, L.; LEVIN, D.; MISLOVE, A. Stick a fork in it: Analyzing the ethereum network partition. In: *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. [S.l.: s.n.], 2017. p. 94–100.

[Kruglik et al. 2019]KRUGLIK, S. et al. Fractional reservation based mempool processing in blockchains. In: *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*. [S.l.: s.n.], 2019. p. 26–30.

[Kumar e Rath 2020]KUMAR, D.; RATH, S. Predicting the trends of price for ethereum using deep learning techniques. In: *Artificial Intelligence and Evolutionary Computations in Engineering Systems*. [S.l.]: Springer, 2020. p. 103–114.

[Lamon, Nielsen e Redondo 2017]LAMON, C.; NIELSEN, E.; REDONDO, E. Cryptocurrency price prediction using news and social media sentiment. *SMU Data Sci. Rev*, v. 1, n. 3, p. 1–22, 2017.

[Lee 2019]LEE, W.-M. Using the web3. js apis. In: *Beginning Ethereum Smart Contracts Programming*. [S.l.]: Springer, 2019. p. 169–198.

[LeMahieu 2018]LEMAHIEU, C. Nano: A feeless distributed cryptocurrency network. *URL: https://nano. org/en/whitepaper*, 2018.

[Li, Ma e Chang 2018]LI, Q.-L.; MA, J.-Y.; CHANG, Y.-X. Blockchain queue theory. In: SPRINGER. *International Conference on Computational Social Networks*. [S.l.], 2018. p. 25–40.

[Li et al. 2017]LI, X. et al. A survey on the security of blockchain systems. *Future Generation Computer Systems*, Elsevier, 2017.

[Liao e Vemuri 2002]LIAO, Y.; VEMURI, V. R. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, Elsevier, v. 21, n. 5, p. 439–448, 2002.

[Lorena e Carvalho 2007]LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

[Mascarenhas, Vieira e Ziviani 2018]MASCARENHAS, J. Z.; VIEIRA, A. B.; ZIVIANI, A. Análise da rede de transações do ethereum. In: SBC. *Anais do I Workshop em Blockchain: Teoria, Tecnologias e Aplicações (WBlockchain-SBRC 2018)*. [S.l.], 2018. v. 1.

[Mehar et al. 2019]MEHAR, M. I. et al. Understanding a revolutionary and flawed grand experiment in blockchain: the dao attack. *Journal of Cases on Information Technology (JCIT)*, IGI Global, v. 21, n. 1, p. 19–32, 2019.

[Meneghetti, Sala e Taufer 2020]MENEGHETTI, A.; SALA, M.; TAUFER, D. A survey on pow-based consensus. *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, p. 2516–0281, 2020.

[Meshcheryakov, Ivanov et al. 2020]MESHCHERYAKOV, A.; IVANOV, S. et al. Ethereum as a hedge: The intraday analysis. *Economics Bulletin*, AccessEcon, v. 40, n. 1, p. 101–108, 2020.

[Möser e Böhme 2015]MÖSER, M.; BÖHME, R. Trends, tips, tolls: A longitudinal study of bitcoin transaction fees. In: SPRINGER. *International Conference on Financial Cryptography and Data Security*. [S.l.], 2015. p. 19–33.

[Motamed e Bahrak 2019]MOTAMED, A. P.; BAHRAK, B. Quantitative analysis of cryptocurrencies transaction graph. *Applied Network Science*, v. 4, n. 1, p. 131, Dec 2019. ISSN 2364-8228. Disponível em: <https://doi.org/10.1007/s41109-019-0249-6>.

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

[Narayanan et al. 2016]NARAYANAN, A. et al. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. [S.l.]: Princeton University Press, 2016.

[Niranjan et al. 2019]NIRANJAN, A. et al. Ekrv: Ensemble of knn and random committee using voting for efficient classification of phishing. In: *Progress in Advanced Computing and Intelligent Engineering*. [S.l.]: Springer, 2019. p. 403–414.

[Oliveira et al. 2021]OLIVEIRA, V. et al. Analyzing transaction confirmation in ethereum using machine learning techniques. *SIGMETRICS Perform. Eval. Rev.*, 2021.

[Paar e Pelzl 2010]PAAR, C.; PELZL, J. Sha-3 and the hash function keccak. *Understanding Cryptography-A Textbook for Students and Practitioners*, Springer Berlin Heidelberg, 2010.

[Paolanti e Frontoni 2020]PAOLANTI, M.; FRONTONI, E. Multidisciplinary pattern recognition applications: A review. *Computer Science Review*, Elsevier, v. 37, p. 100276, 2020.

[Payette, Schwager e Murphy 2017]PAYETTE, J.; SCHWAGER, S.; MURPHY, J. *Characterizing the ethereum address space*. 2017. `https://pdfs.semanticscholar.org/db53/a0281ea25f0041ca4aa812be5c9013f33f26.pdf`.

[Pierro e Rocha 2019]PIERRO, G. A.; ROCHA, H. The influence factors on ethereum transaction fees. In: IEEE. *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. [S.l.], 2019. p. 24–31.

[Powers 2011]POWERS, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, Bioinfo Publications, 2011.

[Princy et al. 2020]PRINCY, R. J. P. et al. Prediction of cardiac disease using supervised machine learning algorithms. In: IEEE. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. [S.l.], 2020. p. 570–575.

[Raynal 2010]RAYNAL, M. Communication and agreement abstractions for fault-tolerant asynchronous distributed systems. *Synthesis Lectures on Distributed Computing Theory*, Morgan & Claypool Publishers, v. 1, n. 1, p. 1–273, 2010.

[Ren et al. 2020]REN, W. et al. A flexible method to defend against computationally resourceful miners in blockchain proof of work. *Information Sciences*, Elsevier, v. 507, p. 161–171, 2020.

[Ricci et al. 2019]RICCI, S. et al. Learning blockchain delays: A queueing theory approach. *SIGMETRICS Perform. Eval. Rev.*, ACM, v. 46, n. 3, p. 122–125, 2019.

[Rish et al. 2001]RISH, I. et al. An empirical study of the naive bayes classifier. In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. [S.l.: s.n.], 2001. v. 3, n. 22, p. 41–46.

[Rohilla, Chakraborty e Kumar 2020]ROHILLA, V.; CHAKRABORTY, S.; KUMAR, R. Car auomation simulator using machine learning. *Available at SSRN 3566915*, 2020.

[Saad et al. 2020]SAAD, M. et al. Contra-*: Mechanisms for countering spam attacks on blockchain memory pools. *arXiv preprint arXiv:2005.04842*, 2020.

[Saad et al. 2019]SAAD, M. et al. Mempool optimization for defending against ddos attacks in pow-based blockchain systems. In: IEEE. *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. [S.l.], 2019. p. 285–292.

[Saad et al. 2019]SAAD, M. et al. Shocking blockchain's memory with unconfirmed transactions: New ddos attacks and countermeasures. *Blockchain for Distributed Systems Security*, John Wiley & Sons, p. 205, 2019.

[Sambasivam, Amudhavel e Sathya 2020]SAMBASIVAM, G.; AMUDHAVEL, J.; SATHYA, G. A predictive performance analysis of vitamin d deficiency severity using machine learning methods. *IEEE Access*, IEEE, v. 8, p. 109492–109507, 2020.

[Schneider 1990]SCHNEIDER, F. B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 22, n. 4, p. 299–319, 1990.

[Sousa et al. 2019]SOUSA, J. E. d. A. et al. An analysis of the fees and pending time correlation in ethereum. In: *IFIP LANOMS, 9th Latin American Network Operations and Management Symposium*. [S.l.: s.n.], 2019.

[Sousa et al. 2020a]SOUSA, J. E. d. A. et al. An analysis of the fees and pending time correlation in ethereum. *International Journal of Network Management nem.2113*, 2020.

[Sousa et al. 2020b]SOUSA, J. E. d. A. et al. Simulação de penny attack no ethereum e sua identificação usando classificadores. In: *Anais do III Workshop em Blockchain: Teoria, Tecnologia e Aplicações*. [S.l.: s.n.], 2020.

[Sousa et al. 2021]SOUSA, J. E. d. A. et al. Fighting under-price dos attack in ethereum with machine learning techniques. *SIGMETRICS Perform. Eval. Rev.*, 2021.

[Szilágyi 2017]SZILÁGYI, P. Clique poa protocol & rinkeby poa testnet. *URL https://github. com/ethereum/EIPs*, n. 225, 2017.

[Takefuji 2020]TAKEFUJI, Y. Security protection mechanisms must be embedded in blockchain applications. *Journal of Chemical Education*, ACS Publications, 2020.

[Tavares et al. 2018]TAVARES, M. et al. Wallid: Secure your id in an ethereum wallet. In: IEEE. *2018 International Conference on Intelligent Systems (IS)*. [S.l.], 2018. p. 714–721.

[Tedeschi et al. 2019]TEDESCHI, E. et al. Predicting transaction latency with deep learning in proof-of-work blockchains. In: IEEE. *2019 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2019. p. 4223–4231.

[Thakur et al. 2019]THAKUR, S. et al. Code sharing application on ethereum. Jaypee University of Information Technology; Solan; HP, 2019.

[Udaiyakumar et al. 2020]UDAIYAKUMAR, R. et al. A comparative study on machine learning and artificial neural networking algorithms. In: IEEE. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. [S.l.], 2020. p. 516–517.

[Ulusoy 2018]ULUSOY, G. *Analysis of Cryptocurrencies*. Dissertação (Mestrado) — Başkent Üniversitesi Fen Bilimleri Enstitüsü, 2018.

[Vujičić, Jagodić e Ranjić 2018]VUJIČIĆ, D.; JAGODIĆ, D.; RANJIĆ, S. Blockchain technology, bitcoin, and ethereum: A brief overview. In: IEEE. *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.], 2018. p. 1–6.

[Vukolić 2015]VUKOLIĆ, M. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In: SPRINGER. *International workshop on open problems in network security*. [S.l.], 2015. p. 112–125.

[Wang et al. 2019]WANG, W. et al. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, IEEE, v. 7, p. 22328–22370, 2019.

[Weber et al. 2017]WEBER, I. et al. On availability for blockchain-based systems. In: IEEE. *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. [S.l.], 2017. p. 64–73.

[Wood et al. 2014]WOOD, G. et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, v. 151, n. 2014, p. 1–32, 2014.

[Wüst e Gervais 2016]WÜST, K.; GERVAIS, A. *Ethereum eclipse attacks*. [S.l.], 2016.

[Xu et al. 2020]XU, G. et al. Am i eclipsed? a smart detector of eclipse attacks for ethereum. *Computers & Security*, Elsevier, v. 88, p. 101604, 2020.

[Xu e Huang 2020]XU, Y.; HUANG, Y. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access*, IEEE, v. 8, p. 17434–17441, 2020.

[Zhang e Lee 2019]ZHANG, S.; LEE, J.-H. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Transactions on Industrial Informatics*, IEEE, v. 15, n. 10, p. 5715–5722, 2019.

[Zheng et al. 2017]ZHENG, Z. et al. An overview of blockchain technology: Architecture, consensus, and future trends. In: IEEE. *2017 IEEE international congress on big data (BigData congress)*. [S.l.], 2017. p. 557–564.

[Zheng et al. 2018]ZHENG, Z. et al. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, Inderscience Publishers (IEL), v. 14, n. 4, p. 352–375, 2018.

[Zoican et al. 2018]ZOICAN, S. et al. Blockchain and consensus algorithms in internet of things. In: IEEE. *2018 International Symposium on Electronics and Telecommunications (ISETC)*. [S.l.], 2018. p. 1–4.