UNIVERSIDADE FEDERAL DO PAMPA

João Otávio Massari Chervinski

Análise da rastreabilidade das transações da criptomoeda Monero

Alegrete
2018

João Otávio Massari Chervinski

Análise da rastreabilidade das transações da criptomoeda Monero

Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Diego Luis Kreutz

Coorientador: Prof. Dr. Jiangshan Yu

Alegrete
2018

João Otávio Massari Chervinski

Análise da rastreabilidade das transações da criptomoeda Monero

Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Projeto de Trabalho de Conclusão de Curso defendido e aprovado em 05 de DEZ de 2018

Banca examinadora:

Prof. Me. Diego Luis Kreutz
Orientador
UNIPAMPA

Prof. Dr. Marcelo Caggiani Luizelli
UNIPAMPA

Prof. Dr. Marcelo Resende Thielo
UNIPAMPA

Pesquisador Dr. Roger Kreutz Immich
IC/UNICAMP

RESUMO

A adoção de criptomoedas como forma de pagamento vem crescendo expressivamente desde o lançamento da primeira moeda digital descentralizada, a Bitcoin. Diferentemente do dinheiro tradicional, criptomoedas utilizam apenas pseudônimos para a identificação dos usuários, permitindo que transações sejam realizadas sem que informações pessoais sejam reveladas. Mas, apesar da privacidade fornecida pelo uso de pseudônimos, alguns trabalhos mostram que através da análise dos dados gerados pelas transações é possível identificar os usuários envolvidos. Com a intenção de proteger as identidades dos usuários, foram criadas criptomoedas cujo foco é a privacidade das transações. Entretanto, até mesmo essas moedas tem sido alvo de ataques de rastreio de dados bem sucedidos. O objetivo deste trabalho é realizar um estudo sobre os ataques de rastreio de dados de transações da criptomoeda Monero, cujo foco é a privacidade dos usuários. Resultados obtidos através da implementação de ataques existentes na literatura são apresentados e um novo ataque, denominado ataque de manipulação de *mixins*, é proposto. Os resultados obtidos mostram que um atacante com controle sobre 25% das chaves geradas em um período de um ano é capaz de rastrear até 85,59% de todas as entradas de transações deste mesmo período. As análises apresentadas enfatizam a existência de vulnerabilidades no sistema Monero e mostram a importância da detecção e correção de falhas de segurança.

Palavras-chave: Criptomoedas. Monero. Rastreabilidade. Privacidade.

ABSTRACT

The adoption of cryptocurrencies as a form of payment has been growing significantly since the launch of the first decentralized digital currency, Bitcoin. Unlike fiat money, cryptocurrencies require only the use of pseudonyms to identify users, allowing them to participate in transactions without the need to disclose any personal information. Despite providing privacy, the use of pseudonyms is not enough, as many studies show that it is possible to identify users through the analysis of transaction data. Some cryptocurrencies were created with the intent of providing better privacy for users, but even those have been shown to be vulnerable to traceability attacks. This work conducts a study on traceability attacks on Monero, a privacy-centered cryptocurrency. We present results obtained through the replication of existing strategies and propose a novel traceability attack, based on flooding the blockchain with attacker's keys. Simulation results show that an attacker who controls 25% of the keys created within a one year interval is able to trace up to 85,59% of all the input keys used in the same interval. Our findings indicate the existence of vulnerabilities in Monero's privacy mechanisms. The results shown emphasize the importance of detecting and fixing security issues.

Key-words: Cryptocurrencies. Monero. Traceability. Privacy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ASIC *Application Specific Integrated Circuit*

BTC Unidade da moeda do sistema Bitcoin

CSV *Comma-separated values*

ECDSA *Elliptic Curve Digital Signature Algorithm*

GPU *Graphics Processing Unit*

I2P *Invisible Internet Project*

IP *Internet Protocol*

JSON *Javascript Object Notation*

PoS *Proof-of-Stake*

PoW *Proof-of-Work*

RingCT *Ring Confidential Transaction*

RIPEMD *RACE Integrity Primitives Evaluation Message Digest*

SHA *Secure Hashing Algorithm*

STXO *Spent Transaction Output*

UTXO *Unspent Transaction Output*

Wasm WebAssembly

XMR Unidade da moeda do sistema Monero

SUMÁRIO

1 INTRODUÇÃO

Desde o lançamento da criptomoeda Bitcoin (NAKAMOTO, 2008), vêm ocorrendo uma revolução nos sistemas de pagamento em todo o mundo. Um dos fatores que impulsionam a adoção deste tipo de moeda é a natureza descentralizada do sistema sobre o qual as criptomoedas operam, uma tecnologia chamada de *blockchain*. A rede Bitcoin funciona através de conexões ponto-a-ponto, onde os participantes trabalham de maneira colaborativa para validar transações, utilizando assinaturas criptográficas para garantir a segurança e a verificabilidade das mesmas, por isso o nome criptomoeda. Diferentemente do que ocorre com dinheiro tradicional, não existe nenhuma autoridade central que regula o uso da Bitcoin, ao invés disso, a moeda é mantida por colaboradores e tanto seu código-fonte quanto o histórico de todas as transações é aberto, ou seja, pode ser acessado por qualquer um. Devido a popularidade alcançada pela Bitcoin, houve o surgimento de outras criptomoedas, chamadas de *altcoins*, como Ethereum (WOOD, 2014), Zcash (SASSON et al., 2014) e Dash (DUFFIELD; DIAZ, 2014).

A proposta de criação de moedas digitais, no entanto, não é recente. Como é destacado no primeiro trabalho que trata sobre esse assunto, a privacidade é uma característica desejável neste tipo de moeda (CHAUM, 1983). Porém, no que diz respeito à Bitcoin, há trabalhos que apresentam métodos capazes de identificar usuários da rede, seja através de seus endereços IP ou de seus nomes de usuário em fóruns e serviços na *Web* (BIRYUKOV; KHOVRATOVICH; PUSTOGAROV, 2014; MEIKLEJOHN et al., 2013; FLEDER; KESTER; PILLAI, 2015).

Com o objetivo de solucionar os problemas de privacidade enfrentados pelos usuários de criptomoedas, em Abril de 2014 foi lançada uma moeda chamada Monero (The Monero Project, 2018). Monero é uma criptomoeda de código aberto, cujo foco é garantir a privacidade e a irrastreabilidade das transações. Contudo, mesmo buscando oferecer um nível de segurança superior ao Bitcoin, as versões originais da criptomoeda Monero apresentavam falhas no que diz respeito à manutenção da privacidade dos usuários, como mostram alguns trabalhos (KUMAR et al., 2017; MILLER et al., 2017). Isto demonstra que a tarefa de desenvolver uma criptomoeda que seja imune contra ataques à privacidade dos usuários é uma tarefa complexa. Os trabalhos que apresentam ataques contra o sistema Monero também evidenciam a importância da detecção e correção de vulnerabilidades em criptomoedas.

Neste trabalho são apresentadas e discutidas técnicas de rastreio de dados de usuários de criptomoedas, com foco no sistema Monero. O principal objetivo é buscar por vulnerabilidades nos mecanismos de privacidade do sistema, identificando vetores de ataque existentes e propondo novas estratégias que permitam rastrear os dados dos usuários. Neste contexto, é também proposto um novo ataque denominado ataque de manipulação de *mixins*, baseado na criação de um grande número de chaves de saídas de transações.

O primeiro passo para a realização do trabalho é estudar os ataques existentes nos sistemas das criptomoedas Bitcoin e Monero, com o intuito de comparar os níveis de segurança oferecidos por cada uma das moedas. Observando a literatura existente, alguns dos métodos de investigação para a descoberta de falhas de privacidade em criptomoedas como a Bitcoin e Monero são:

(m_1) **Análise dos grafos de transações.** O objetivo é gerar grafos que representam as transações entre os usuários do sistema, onde as arestas representam as transações e os nodos representam os usuários. Os caminhos entre os nodos são usados para rastrear a origem das criptomoedas e identificar relações entre usuários.

(m_2) **Análise do tráfego de rede.** Os dados disseminados pelos usuários na rede durante a criação das transações podem ser analisados, permitindo correlacionar endereços de rede dos usuários com as transações realizadas por eles no sistema.

(m_3) **Análise dos dados do *blockchain*.** As informações geradas por transações e recompensas pela criação de blocos são armazenadas no *blockchain*. Heurísticas de análise de dados podem ser aplicadas à essas informações, revelando as entradas reais de transações.

As principais contribuições deste trabalho podem ser resumidas em:

(c_1) Levantamento e discussão do estado da arte no que tange a ataques contra a criptomoeda Monero.

(c_2) Apresentação e discussão detalhada das tecnologias integrantes dos *blockchains* e das criptomoedas Bitcoin e Monero.

(c_3) Identificação de um novo vetor de ataque contra a privacidade das transações da criptomoeda Monero baseado na exploração do protocolo *Bulletproof*. Adicionalmente, é apresentada uma proposta para a redução dos custos do ataque através do uso de mineração via navegadores.

(c_4) Apresentação de desafios de pesquisa relacionados à privacidade da criptomoeda Monero, contribuindo para o desenvolvimento de pesquisas futuras.

O restante deste documento está organizado como segue. No Capítulo 2 são apresentadas as tecnologias que integram os sistemas de criptomoedas. Os trabalhos relacionados são discutidos no Capítulo 3, comparando as estratégias de ataque apresentadas em cada um deles. As etapas de desenvolvimento do trabalho são apresentadas no Capítulo 4. O Capítulo 5 apresenta uma análise de vulnerabilidades da criptomoeda Monero através da replicação de ataques existentes na literatura. Uma nova estratégia de ataque contra

a privacidade das transações da criptomoeda Monero é apresentada no Capítulo 6. O Capítulo 7 apresenta desafios de pesquisa relacionados a exploração de vulnerabilidades em criptomoedas com foco em privacidade e o Capítulo 8 conclui o documento com algumas considerações acerca do trabalho realizado.

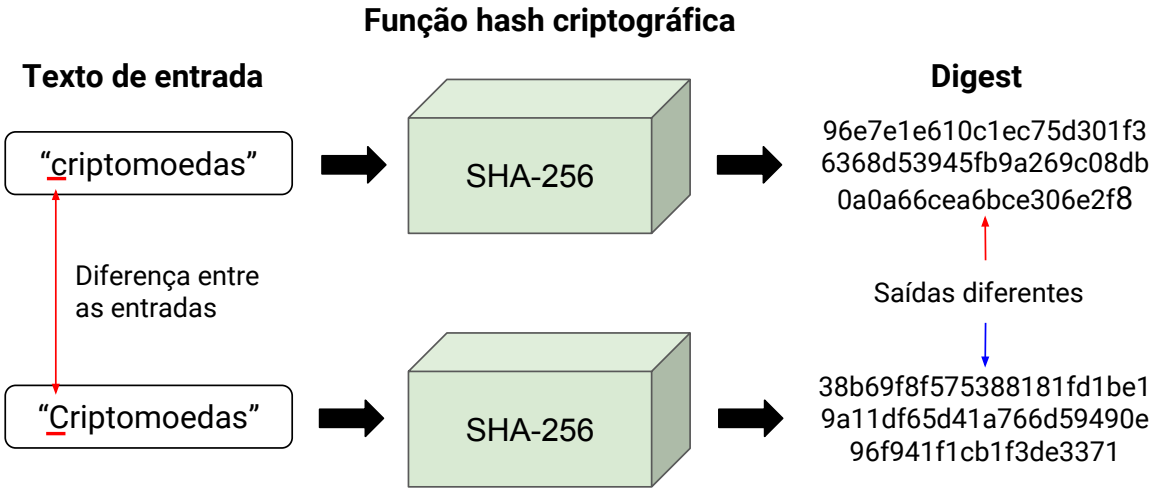
2 ESTADO DA ARTE

Neste capítulo são discutidas as tecnologias que integram os sistemas de criptomoedas, desde a estrutura distribuída de armazenamento de dados até os mecanismos que fornecem privacidade e segurança às moedas digitais. Também é apresentada uma estratégia de ataque recente baseada na injeção de códigos de mineração maliciosos em páginas da *web*.

2.1 Tecnologia Blockchain

O *blockchain* é um registro distribuído formado por uma cadeia de blocos de dados, conectados uns aos outros por um sistema que utiliza funções *hash* criptográficas. A Figura 1 apresenta o resultado da aplicação da função *hash* criptográfica SHA-256 em duas entradas distintas. Como pode ser observado, a saída da função é completamente diferente para as duas entradas aparentemente idênticas.

Figura 1 – Aplicação de uma função *hash* criptográfica.



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

- A aplicação da função sobre o mesmo dado deverá sempre retornar o mesmo resultado.
- Computar um *digest* para uma determinada entrada deve ser fácil, i.e, a operação pode ser realizada em menos de um segundo.
- Descobrir quais dados foram utilizados como entrada da função analisando somente o *digest* deve ser muito difícil. Para isso é necessária a aplicação da função em

entradas distintas até que a saída gerada pela entrada analisada seja igual ao *digest* em questão, uma operação que pode necessitar de vários anos de processamento considerando a capacidade atual dos computadores.

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

- Uma pequena mudança na entrada deve alterar o *digest* resultante de tal forma que não seja possível encontrar alguma relação entre as saídas.

Este tipo de função é utilizado para ajudar a garantir a integridade de informações, já que uma pequena mudança nos dados de entrada altera o *digest* resultante, como pode ser observado na Figura 1. Ao aplicar uma função *hash* em um arquivo e enviar o *digest* para a pessoa que irá recebê-lo, o receptor poderá aplicar a função novamente e verificar se o resultado é igual ao *digest* recebido. Desta forma, ela garante que o arquivo não foi modificado, desde que o *digest* tenha sido recebido de forma segura. Por exemplo, suponha que João trabalhe no setor financeiro de uma empresa e que tenha sido encarregado de realizar uma transferência de dinheiro da conta da empresa para a conta de algumas empresas parceiras. João recebeu de Maria o arquivo contendo os dados das contas bancárias por email.

Um usuário malicioso, realizando um ataque de interceptação de dados, pode alterar o documento contido no email antes que ele seja recebido por João, adicionando novas contas bancárias na lista, por exemplo. Para certificar-se de que João receberá o arquivo com as mesmas informações enviadas originalmente, Maria computa o *digest* do documento anexado no email. Considere que a empresa utiliza um canal seguro para comunicação auxiliar, para o envio de dados como os *digests*. Maria envia o *digest* do arquivo para João utilizando a rede privada segura. Ao fazer o *download* do arquivo com os dados das contas, João precisa certificar-se de que nada foi modificado. Para isto, ele computa o *digest* do arquivo recebido e compara-o com o *digest* enviado por Maria. Se eles forem iguais, o documento não foi modificado.

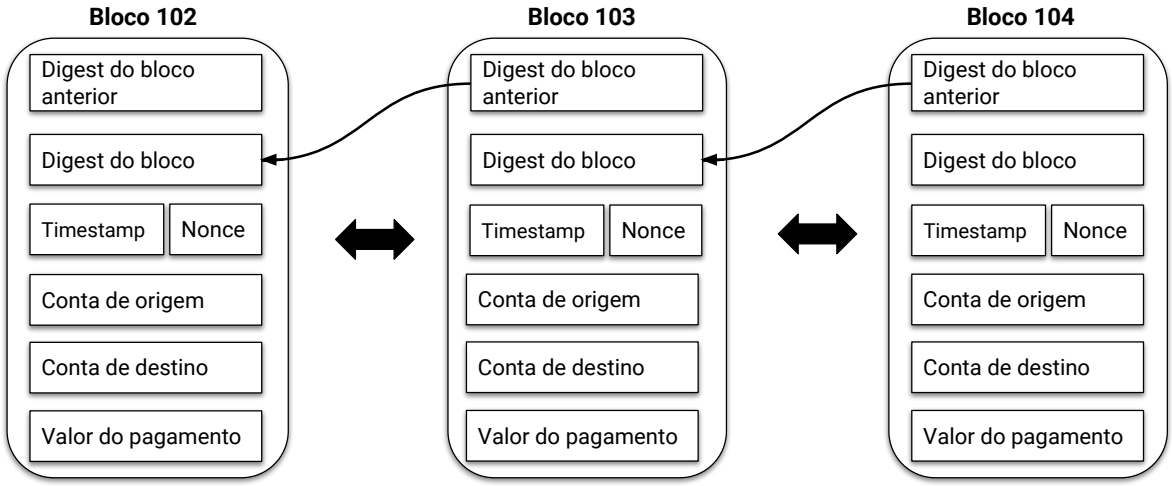
Os *blockchains* possuem uma propriedade interessante para o registro de transações: quando um bloco é adicionado ao final da cadeia de informações formada por todos os blocos já criados, torna-se uma tarefa muito difícil alterá-lo. Suponha a existência de um *blockchain* para uma aplicação bancária fictícia. Cada bloco da aplicação armazena a informação de uma transação entre duas contas. Um exemplo de um conjunto de dados armazenados em cada bloco de um *blockchain* fictício voltado para o armazenamento de transações financeiras é:

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

- *Digest* do novo bloco: Resultado da aplicação de uma função *hash* criptográfica sobre os dados do novo bloco.
- *Timestamp*: Representação da data e hora de criação do bloco.
- *Nonce*: Número auxiliar utilizado para realização do cálculo da função *hash* criptográfica.
- Conta de origem: A conta de onde o dinheiro será retirado.
- Conta de destino: A conta que receberá o dinheiro.
- Valor da transação: Quantia de dinheiro a ser transferida.

Considerando a estrutura de bloco apresentada, existem duas informações que garantem que a ordem dos blocos irá manter-se correta, o *digest* do bloco anterior na cadeia e o *digest* do próprio bloco. A Figura 2 ilustra a conexão entre uma cadeia de blocos. A utilização de uma função *hash* criptográfica garante que cada bloco irá gerar um *digest* único, permitindo estabelecer um vínculo forte e único entre os blocos da cadeia. Isso garante que exista uma única sequência válida de blocos.

Figura 2 – Representação dos blocos em um *blockchain*.



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

Como as funções *hash* criptográficas garantem que é pouco provável gerar o mesmo *digest* a partir de dados de entrada diferentes, seria mais fácil alterar o campo que contém o *digest* que aponta para o bloco anterior na cadeia em cada um dos blocos seguintes.

Bastaria que o atacante mudasse os campos nos blocos posteriores, calculasse seus *digests* e repetisse o processo até o final do *blockchain*. Um sistema utilizado para controlar a criação de novos blocos em alguns *blockchains*, chamado de Prova de Trabalho, do inglês *Proof-of-Work* (PoW), ajuda a evitar este ataque. Através deste sistema, não basta apenas calcular o *digest* do bloco para que ele seja adicionado à cadeia. O resultado da função *hash* deve obedecer à uma restrição que requer um grande esforço computacional para ser atendida. A restrição adotada na prática pela Bitcoin é encontrar um bloco cujo *digest* resultante possua os primeiros n bits iguais a zero, onde n depende da dificuldade de mineração determinada pelo sistema. Para variar a saída da função *hash* e encontrar um bloco que atenda a restrição é necessário alterar os dados de entrada, para esta finalidade há em cada bloco um campo chamado de *nonce*, que é alterado até que o resultado desejado seja obtido.

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

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

Apesar das criptomoedas serem o caso de uso mais comum dos *blockchains* atualmente, esta tecnologia está se popularizando e sua aplicação como uma estrutura de armazenamento de dados vem sendo investigada em uma grande gama de sistemas em diversas áreas (IEEE, 2017). Os *blockchains* possuem potencial para substituir plataformas digitais, não somente na área de finanças, mas também na cadeia de suprimentos, no setor de energia e no setor de agricultura (KSHETRI, 2018).

2.2 Criptomoedas

A adoção de criptomoedas como forma de pagamento vem crescendo rapidamente devido aos benefícios que elas oferecem em relação às formas de pagamento tradicio-

nais (MEDFAR87, 2018). Um usuário pode efetuar um pagamento para um receptor em qualquer lugar do mundo a qualquer momento, sem a necessidade de uma instituição intermediária, o que implica em menores taxas de transações, maior controle, e mais privacidade. Mas, apesar dos benefícios oferecidos por esse tipo de moeda, existem desvantagens como a impossibilidade de reverter transações e de obter suporte caso ocorram erros no sistema. A volatilidade dos preços da moedas é outro fator negativo. No caso da Bitcoin, o preço pode variar mais de 10% em poucas horas (ADKISSON, 2018).

2.2.1 Mineração

Para obter criptomoedas, um usuário pode realizar uma compra através de serviços especializados em vendas de criptomoedas, chamados de *cryptocurrency exchanges*. Algumas moedas, como a Bitcoin e a Monero, oferecem aos usuários a possibilidade de obtê-las diretamente através do sistema, participando de um processo comumente chamado de mineração. É importante notar que criptomoedas como a Ripple (SCHWARTZ et al., 2014) e a IOTA (POPOV, 2014) não possuem um sistema através do qual os usuários possam obter moedas ao criar blocos de transações, ou seja, não podem ser mineradas. Esta seção discute o processo de mineração que ocorre em criptomoedas similares à Bitcoin e Monero.

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

Devido ao esforço computacional necessário para a criação de blocos válidos pelo esquema de PoW, é necessário um incentivo para que os participantes da rede, ou mineradores, emprestem o seu poder de processamento para ajudar no funcionamento do sistema. Esse incentivo é dado através de uma recompensa em criptomoedas para o participante da rede que conseguir validar primeiro um determinado bloco de transações.

A primeira transação de cada bloco, chamada de *coinbase transaction*, é um tipo especial de transação que não possui entradas e cuja finalidade é enviar o valor da recompensa para o usuário que efetuou a validação do bloco. O sistema de recompensas é geralmente desenvolvido de maneira que, com o passar do tempo, a recompensa por bloco diminua. Isto é necessário para controlar a quantidade de novas moedas criadas devido ao aumento no preço da moeda e outros fatores econômicos. Na Bitcoin, essa diminuição ocorre a cada 210.000 blocos minerados, quando a recompensa é reduzida pela metade. A diminuição da recompensa estende a vida do sistema ao impedir que todo o suprimento de moedas seja emitido em um curto período de tempo, o que acabaria com a motivação para a criação de novos blocos válidos. A redução da recompensa também contribui para a valorização da moeda, que passa a valer mais quando a procura aumenta e a oferta

diminui. Em um dado momento, a validação de novos blocos não irá gerar mais recompensas e o pagamento pela criação de blocos válidos será feito somente através das taxas de transações, pagas pelos usuários. Atualmente, em 2018, a quantia recompensada por bloco na Bitcoin é de 12,5 unidades da moeda, chamada de BTC.

A Tabela 1 mostra a mudança na recompensa por bloco válido criado ao longo do tempo. O período estimado para que ocorra a criação de 210.000 novos blocos e ocorra uma diminuição no valor da recompensa por bloco é de 4 anos. Na prática este período pode ser maior ou menor, apesar do aumento no número de mineradores ao longo do tempo. Esta oscilação ocorre porque o sistema eleva automaticamente a dificuldade de criação de novos blocos quando a velocidade da rede aumenta. Esta estratégia é empregada para manter o tempo entre a criação de novos blocos por volta de dez minutos. Isto também evita a emissão de muitas moedas novas em um curto período de tempo. Quando a dificuldade de criar blocos aumenta, o retorno pela criação de blocos diminui dado o esforço necessário, causando uma redução no número de mineradores. A redução no número de mineradores diminui a capacidade de criação de blocos da rede, que ajusta a dificuldade de acordo. Essas variações causam mudanças no intervalo de tempo entre a diminuição da recompensa por bloco.

Tabela 1 – Recompensa pela validação de blocos na Bitcoin.

Número de Blocos	**Recompensa por bloco**	**Ano**
0	50 BTC	2009
210.000	25 BTC	2012
420.000	12,5 BTC	2016
630.000	6,25 BTC	2020 (estimado)

A recompensa por bloco validado no sistema Monero é regulada continuamente de acordo com a Equação 2.1. A equação considera a quantidade de moedas já emitidas até o momento da criação de um bloco para calcular o valor da recompensa pela sua validação. Os autores do protocolo chamam esse processo de emissão de moedas de *smooth emission* (SABERHAGEN, 2013). Os valores 2^{-20} e 10^{-12} foram incluídos na Equação 2.1 para substituir uma operação de *shift* de bits, com o objetivo de facilitar o entendimento da equação[1].

$$Recompensa = (M - A) \times 2^{-20} \times 10^{-12} \tag{2.1}$$

onde:

M = Suprimento máximo de Monero que pode entrar em circulação, equivalente a $2^{64} - 1$ unidades atômicas da moeda.

[1] <https://monero.stackexchange.com/questions/4253/what-is-the-mining-reward-equation>

A = Unidades atômicas da moeda emitidas até o momento presente.

É comum que mineradores organizem-se em grupos que trabalham em conjunto para criarem blocos válidos, chamados de *mining pools*. Quando um usuário cria um bloco válido, a recompensa é dividida entre todos os participantes do grupo. Este método diminui o valor da recompensa individual de cada minerador, porém garante um fluxo mais estável de renda para todos. A participação em *mining pools* é vantajosa pois podem ser necessários anos até que um usuário consiga computar sozinho um *digest* válido para a criação de um bloco antes dos outros participantes da rede.

2.2.2 Bitcoin

Introduzida através de um *white paper* em uma lista de correio eletrônico sobre criptografia em 2008 e lançada em 2009, Bitcoin foi a primeira criptomoeda (NAKAMOTO, 2008). Anos após seu lançamento, ainda permanece sendo a mais utilizada, possuindo um valor total de mercado de US$ 158 bilhões[2]. A Bitcoin baseia-se em material de pesquisas anteriores, como o esquema de PoW que controla a criação de blocos válidos no *blockchain*, esquemas de assinaturas digitais que garantem que os usuários que utilizam moedas realmente as possuem e técnicas de *timestamping* que marcam a data e a hora da realização das operações. A principal contribuição da Bitcoin foi eliminar a necessidade de uma autoridade central que regule a emissão de moedas e a confirmação de transações. Isto foi possível pela forma descentralizada pela qual o sistema funciona, utilizando uma rede ponto-a-ponto onde usuários participam do processo de validação e verificação da autenticidade das transações. A descentralização também é fruto da maneira como os dados estão armazenados. Todas as transações ocorridas estão armazenadas em um *blockchain* público, isto é, cada partipante da rede pode optar por obter uma cópia desse registro. Usuários que não desejam fazer *download* dos dados do *blockchain* da criptomoeda podem acessá-los através de clientes leves, do inglês *Light-Clients*. Clientes leves são *softwares* que permitem aos usuários acessar dados através de uma conexão com um nó remoto confiável que mantém uma cópia do *blockchain*.

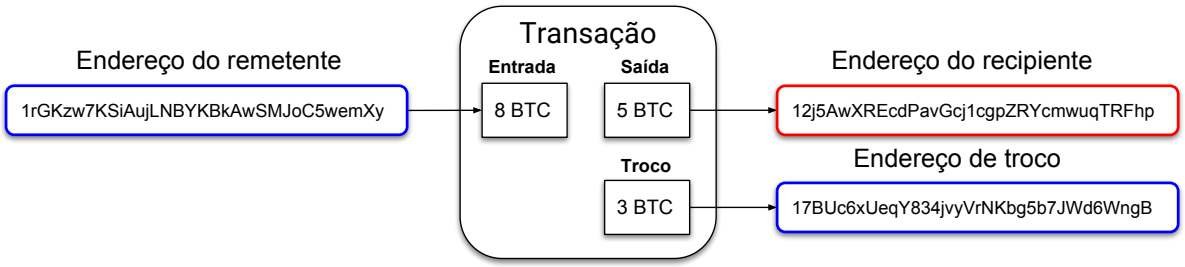
Para enviar e receber transações em Bitcoin, um usuário necessita de um par de chaves criptográficas composto por uma chave pública e uma chave privada. No caso da Bitcoin, a chave pública é utilizada como endereço de envio e recebimento de pagamentos. Um usuário pode divulgar a sua chave pública e outras pessoas podem enviar Bitcoins para esse endereço. A chave privada é utilizada para comprovar que um usuário é dono da chave pública que a acompanha, podendo assim utilizar os fundos recebidos. A chave privada não deve ser compartilhada e deve ser armazenada em segurança para que ninguém além do proprietário possa acessá-la. O endereço de um usuário é derivado de sua chave pública, ao aplicar o algoritmo de *hashing* SHA-256 e depois o algoritmo RIPEMD-160, adicionar

[2] <https://coinmarketcap.com/>

números para controle de erro e controle de versões e, por fim, codificá-lo em BASE58 (TSCHORSCH; SCHEUERMANN, 2016). O processo de derivação da chave é realizado para fornecer segurança adicional, ajudando a ocultar a verdadeira chave pública. Os usuários também podem optar pela utilização de sua chave pública original como endereço.

Em uma transação, existem endereços de entrada e endereços de saída. A Figura 3 ilustra o procedimento de uma transação de Bitcoins contendo uma única entrada. O endereço do remetente é a chave pública que corresponde ao endereço da carteira de Bitcoins do emissor do pagamento. O endereço do recipiente é o endereço da carteira do receptor do pagamento. O endereço de troco deve ser definido pelo emissor do pagamento para que ele receba o troco da operação, caso a chave de entrada utilizada exceda o valor do pagamento.

Figura 3 – Transação de Bitcoins contendo uma única entrada.

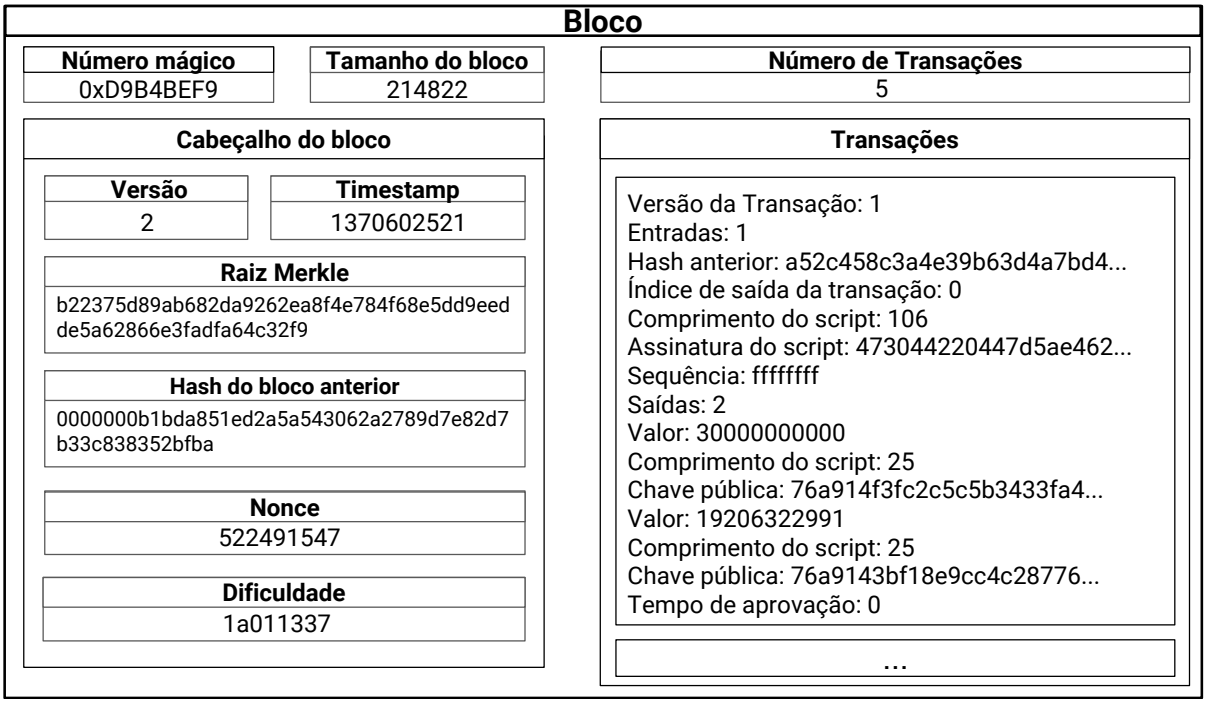


Só podem ser utilizadas como entradas de transações as chaves que foram geradas como saída em uma transação anterior. Endereços de saída de transações são chaves recebidas pelos usuários que recebem as criptomoedas. O saldo de um usuário da Bitcoin consiste na soma dos valores de todas as saídas de transações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

de gerenciar diferentes endereços de um mesmo usuário.

Um bloco é capaz de armazenar milhares de transações, desde que os dados de todas elas somados não ultrapassem o tamanho de 1 MB. A Figura 4 mostra os dados contidos um bloco do sistema Bitcoin. Os blocos criados para armazenar as transações no *blockchain* possuem a seguinte estrutura:

Figura 4 – Estrutura de um bloco do *blockchain* do sistema Bitcoin.



- Número mágico: Valor utilizado para identificar o tipo de estrutura contida nos dados. Neste caso, um bloco. Este valor é específico do protocolo Bitcoin.
- Tamanho do bloco: Especifica o tamanho em *bytes* dos dados contidos no bloco.
- Cabeçalho do bloco: Contém dados que identificam o bloco atual.
  - Versão: Especifica a versão do sistema no momento da criação do bloco.
  - *Timestamp*: Representa o momento no tempo em que o bloco foi criado.
  - Raiz Merkle: Um tipo de *hash* utilizado para verificar a validade das transações contidas nos blocos sem a necessidade de verificar todas as informações das transações. Para realizar a verificação é necessário o cabeçalho dos blocos e uma estrutura chamada de Árvore de Merkle.
  - *Nonce*: Campo cujo valor deve ser modificado até que o *digest* resultante do bloco atenda as exigências do sistema.

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

- Número de transações: Contém o número de transações presentes no bloco atual.
- Transações: Contém os dados de cada uma das transações contidas no bloco.

Como o *blockchain* da Bitcoin é um registro transparente, o histórico de transações de todos os usuários está disponível abertamente. Através da análise dos endereços e do fluxo de transações é possível efetuar ataques que correlacionam os pseudônimos com as identidades dos usuários (MEIKLEJOHN et al., 2013). Para reduzir o impacto de ataques que efetuam a análise das transações é sugerida a utilização de uma nova chave e endereço para cada transação (NAKAMOTO, 2008).

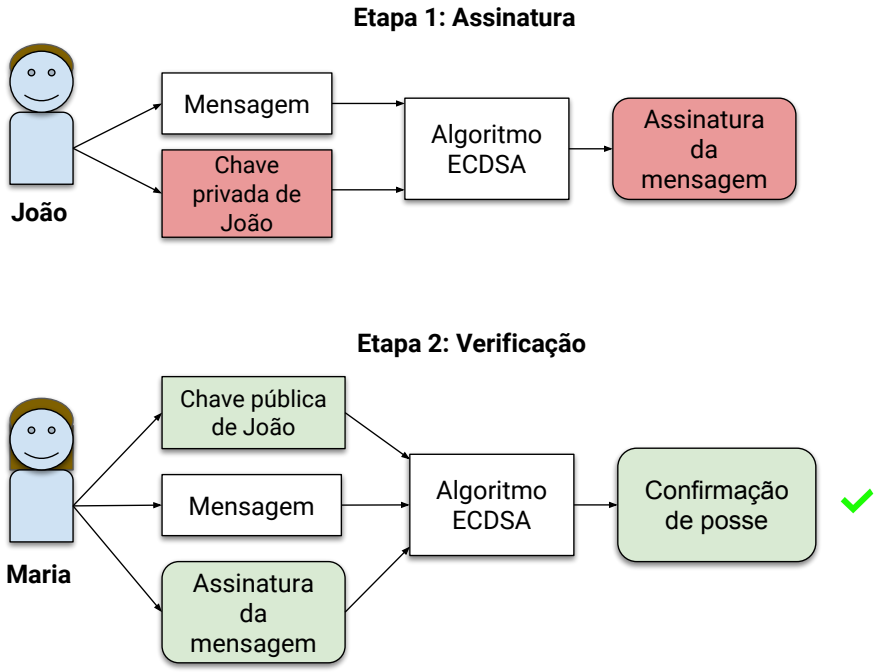
A quantidade de Bitcoins associada a cada usuário não é armazenada explicitamente nos registros. O saldo de um endereço pode ser verificado ao checar todo o seu histórico de transações, calculando quantas Bitcoins foram recebidas e quantas foram enviadas a partir do endereço. Por isso, sempre que um usuário instala pela primeira vez uma carteira de Bitcoins em seu sistema, é necessário que ele verifique todas as transações já ocorridas. A verificação é realizada para checar se os pagadores possuem de fato os valores sendo gastos.

Para garantir a segurança das transações na Bitcoin, um sistema de assinaturas baseado no algoritmo ECDSA é utilizado. A Figura 5 ilustra o processo de criação de uma assinatura digital. João precisa provar que possui um endereço para enviar dinheiro através dele. Para isso, ele cria uma mensagem contendo os dados da transação que deseja realizar. O segundo passo é criar uma assinatura digital, que servirá para provar que João é o dono do endereço do qual as moedas estão sendo enviadas. Utilizando a sua chave privada, João gera a assinatura digital da mensagem e a envia para a rede juntamente com a mensagem e sua chave pública.

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

Após a criação de uma transação, o remetente dissemina na rede uma mensagem avisando que possui uma nova transação. Os participantes interessados nos dados enviam um pedido explícito ao remetente. Os mineradores acumulam transações e as organizam em blocos antes de iniciarem o processo de tentativa de criação de um bloco válido. A dificuldade de criação dos blocos é regulada pelo sistema e é alterada com base no tempo que foi necessário para a criação dos últimos 2.016 blocos. O ajuste é realizado com

Figura 5 – Processo de assinatura de uma transação.



a intenção de manter o tempo necessário para adicionar um bloco a cada dez minutos, para que o tempo de confirmação das transações seja razoável. Levando em consideração o tempo ideal de criação de um bloco, 10 minutos, 2.016 blocos devem ser criados em exatamente duas semanas. Se o tempo necessário para a criação dos últimos 2.016 blocos exceder duas semanas, a dificuldade da criação dos blocos é reduzida, se o tempo for inferior a duas semanas, a dificuldade é elevada. O sistema ajusta a dificuldade da criação de blocos válidos de acordo com a Equação 2.2.

$$D = D_{anterior} \times \frac{T}{2016 \times 10min} \tag{2.2}$$

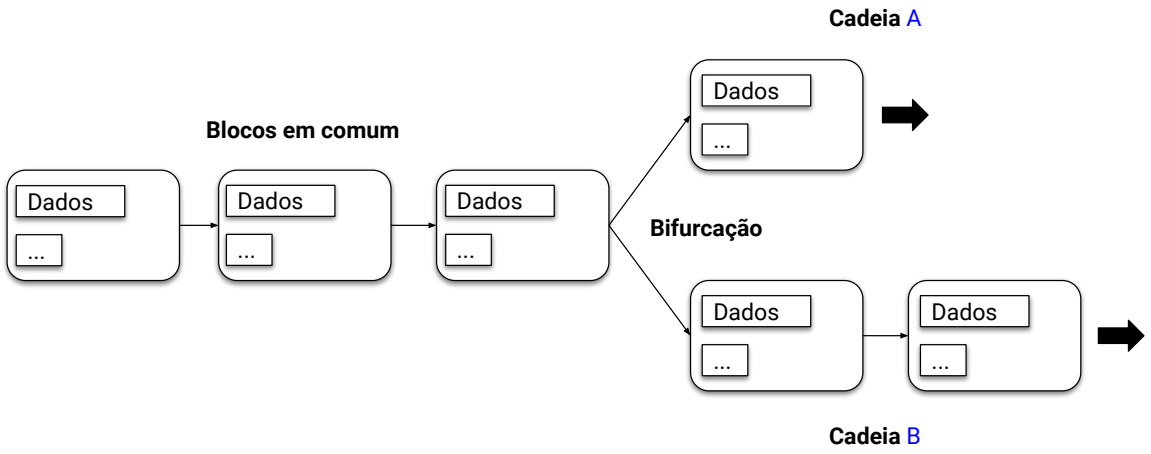
onde:
D = Dificuldade da resolução do problema de criação de um bloco válido.
T = Tempo ocorrido desde a última mudança de dificuldade em minutos.

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

Eventualmente, pode ocorrer uma bifurcação no *blockchain*, causada pelo tempo

necessário para a propagação dos blocos na rede. A Figura 6 ilustra o estado da cadeia de blocos quando ocorre uma bifurcação. O problema ocorre quando um usuário A cria um um bloco válido ao mesmo tempo em que um usuário B cria outro bloco válido contendo transações diferentes. Como é necessária uma quantia de tempo para que a informação seja propagada aos outros participantes da rede, os que recebem primeiro o bloco de A, o colocam no fim de suas cadeias, já os que recebem primeiro o bloco de B, colocam um bloco diferente no final de suas cadeias. Isso causa uma divergência momentânea no *blockchain*. A partir desse momento existem duas cadeias cuja única diferença é o ultimo bloco. Não é possível saber qual das cadeias é a correta.

Figura 6 – Bifurcação na cadeia de blocos.



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

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

2.2.3 Privacidade

O surgimento de trabalhos que relatam os problemas de privacidade na Bitcoin motivaram o desenvolvimento de novas criptomoedas com foco na segurança e privacidade dos usuários. A necessidade de privacidade nos sistemas de criptomoedas, porém, é motivo de polêmica, porque além de compras comuns como as que podem ser feitas através de dinheiro tradicional, criptomoedas são utilizadas para a compra e venda de produtos ilegais, como armas de uso restrito e drogas (TORPEY, 2018). Apesar de algumas atividades ilegais serem motivadas pela existência de criptomoedas, a privacidade continua sendo um direito dos usuários. Criptomoedas que buscam oferecer garantias de privacidade têm recebido mais atenção nos últimos anos (WILLIAMS, 2017).

Mesmo através do uso de pseudônimos como as chaves criptográficas, ainda existe a possibilidade de vincular diferentes endereços à um mesmo usuário (NAKAMOTO, 2008). Quando são efetuadas transações com múltiplas entradas, por exemplo, várias chaves públicas são utilizadas e têm seus valores somados para realizar o pagamento. Como todas as chaves devem ser adicionadas pelo usuário que cria uma transação, é possível assumir que quem efetuou o pagamento possui todas as chaves utilizadas. Dessa forma é possível vincular várias chaves ao usuário que efetua a transação. A divulgação das chaves públicas dos usuários contribui para a vinculação de pseudônimos com as suas identidades reais. Esta associação pode ser feita ao relacionar nomes e informações de contas dos usuários com as chaves públicas compartilhadas em fóruns e páginas da *web*.

Usuários legítimos podem se beneficiar da privacidade. Por exemplo, se a privacidade das transações não for protegida, empresas podem analisar os dados com o objetivo de predizer hábitos e gostos dos usuários. A partir dos dados privados de cada usuário as empresas podem exibir propagandas e ofertas dirigidas de produtos, bem como comercializar os dados de perfil do usuário para outras empresas. Como os dados sobre a utilização de serviços vêm se tornando cada vez mais valiosos, as informações geradas por cada usuário deveriam ser de sua posse somente, cabendo a cada um autorizar ou não a divulgação ou comercialização dos seus dados. Em um cenário ideal, os próprios usuários poderiam realizar a comercialização dos seus dados.

2.2.4 Monero

A Monero foi lançada em Abril de 2014 e é uma criptomoeda descentralizada. Seu código é aberto e seu foco é a privacidade dos usuários. Monero ganhou popularidade devido às suas características que fornecem um nível de privacidade mais elevado do que as chaves pseudoanônimas da Bitcoin e de outros sistemas (KUMAR et al., 2017). A atenção atraída pela criptomoeda fez com que ela subisse para a 9ª posição em termos de valor de mercado se comparada a todas as criptomoedas[3].

[3] <https://coinmarketcap.com/>

Monero utiliza um protocolo chamado de CryptoNote (SABERHAGEN, 2013). Esse protocolo é utilizado também em outras criptomoedas que focam na privacidade dos usuários, tais como Bytecoin (SABERHAGEN, 2013) e DashCoin (DUFFIELD; DIAZ, 2014). O protocolo oferece funcionalidades que são essenciais para a garantia da privacidade no uso de Monero, pois as transações do CryptoNote não podem ser rastreadas através da análise do *blockchain*. Porém, quando o protocolo é utilizado sem precauções, são criadas brechas de segurança que permitem ataques à privacidade das transações, como é mostrado em detalhes na seção 5.2. As duas principais características de privacidade oferecidas pela Monero são:

- **Irrastreabilidade das transações**: Garante que dada uma transação com várias entradas, não é possível descobrir qual entrada foi utilizada, impedindo que seu histórico seja traçado.
- **Não-vinculação de endereços**: Garante que, dadas transações diferentes, não é possível demonstrar que elas foram originadas de um mesmo usuário.

Estas duas propriedades são baseadas nas funcionalidades de chaves de uso único e *ring signatures*, oferecidas pelo protocolo CryptoNote. Além destas, o sistema Monero oferece outras garantias de segurança e privacidade: as *Ring Confidential Transaction*s (RingCTs), e o protocolo de roteamento Kovri. No lado do recipiente das transações, a identidade do usuário é protegida através da utilização dos endereços de uso único, chamadas na Monero de *stealth addresses*. Cada usuário possui dois pares de chaves pública e privada, um par de chaves de visualização e um par de chaves de utilização de fundos. Sempre que uma quantia em XMR, a unidade da moeda do sistema Monero, é enviada para um recipiente, é criado um endereço de uso único que permite que somente o recipiente saiba que recebeu um pagamento.

Suponha que o usuário João deseja enviar uma quantia em XMR para a usuária Maria. João usa as duas chaves públicas de Maria e um número aleatório para gerar um endereço de uso único para a qual o pagamento será enviado. Maria realiza uma varredura no *blockchain* verificando as saídas de transações com a sua chave privada de visualização. Dessa forma, Maria é capaz de identificar os endereços de uso único destinados a si. Isso permite ao recipiente identificar transações e evita que qualquer outra pessoa com acesso aos dados do *blockchain* consiga identificar quem recebe a saída de uma transação. Com a sua chave privada de utilização de fundos, Maria consegue provar que é a dona daquela chave de saída.

As *ring signatures* são um tipo de assinatura digital onde um grupo de possíveis remetentes são utilizados em conjunto para criar uma assinatura digital que autoriza a transação. Ao utilizar criptografia para esconder os dados da transação o remetente original mantem-se anônimo. A assinatura digital é composta pelo verdadeiro remetente juntamente com outros remetentes válidos. O verdadeiro remetente usa uma chave de

uso único para efetuar o pagamento e as chaves restantes são retiradas de transações anteriores contidas no *blockchain* e são chamadas de *mixins* ou misturas. Todas essas chaves compõem as entradas de uma transação, tornando difícil para um observador deduzir qual entrada é a verdadeira.

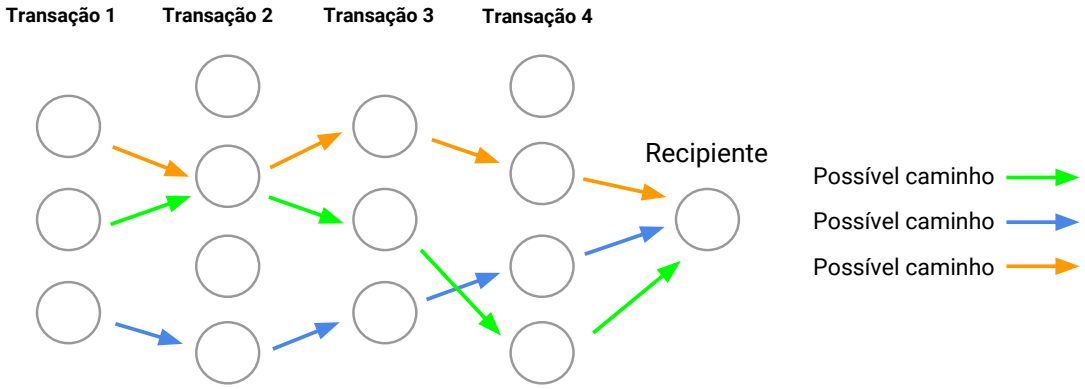
Os usuários podem escolher o número de *mixins* utilizados em uma transação. Em versões antigas do sistema, o usuário podia optar por não incluir nenhuma *mixin*, tornando a chave de entrada visível. Apesar da liberdade de escolha, não é recomendada a utilização de um número muito elevado de *mixins*, pois isso faz com que a transação se destaque entre as outras presentes no *blockchain*. Utilizar um número muito elevado de *mixins* também gera custos adicionais, pois os usuários devem pagar uma taxa para realizar as transações de acordo com o seu tamanho em *bytes*.

Para evitar que a mesma chave seja usada duas vezes para realizar pagamentos, uma vez que os outros participantes da rede não são capazes de deduzir se ela já foi utilizada, existem as imagens de chaves. Imagens de chaves são chaves criptográficas únicas que são derivadas da chave real sendo utilizada na transação. As imagens das chaves fornecem provas de que a chave sendo utilizada para o pagamento em uma entrada de transação não foi utilizada anteriormente. Isto é feito sem revelar qualquer informação sobre a chave real.

A utilização das *ring signatures* faz com que as transações não sejam transparentes, dificultando a identificação de suas origens e o rastreamento dos seus históricos. Devido à quantidade de possíveis caminhos, a análise do grafo de transações se torna inviável. A Figura 7 ilustra uma tentativa de análise das transações do sistema Monero. Cada coluna de círculos representa uma transação do sistema e as chaves de entrada são representadas pelos círculos em cada coluna. A chave utilizada na transação pode ser qualquer uma das entradas, portanto, todas as combinações de caminhos da coluna inicial até o recipiente são válidas. As setas coloridas mostram três caminhos válidos no grafo de transações. Como o *blockchain* armazena milhões de transações, um atacante precisará analisar um número de caminhos que possui dependência exponencial com o número de transações para tentar deduzir a chave correta de uma entrada.

Para aumentar ainda mais o nível de privacidade nas transações de Monero, foi criado o protocolo RingCT. A Figura 8 apresenta uma transação utilizando o protocolo RingCT, onde o valor transacionado não é visível. Antes da criação deste protocolo, os valores das transações eram visíveis no *blockchain* e precisavam ser divididos em várias partes, chamadas de denominações. Os valores de denominações variam de 10^{-12} à 10^{6} e são identificadas por um prefixo composto pelo nome da unidade (pico à mega) e o sufixo "nero". A divisão dos valores era necessária porque chaves usadas como *mixins* em uma transação precisavam possuir o mesmo valor que a chave real sendo utilizada. Devido ao grande número de valores diferentes, algumas transações não encontravam *mixins* suficientes para que alcançassem um bom nível de segurança, o que levou a várias transações

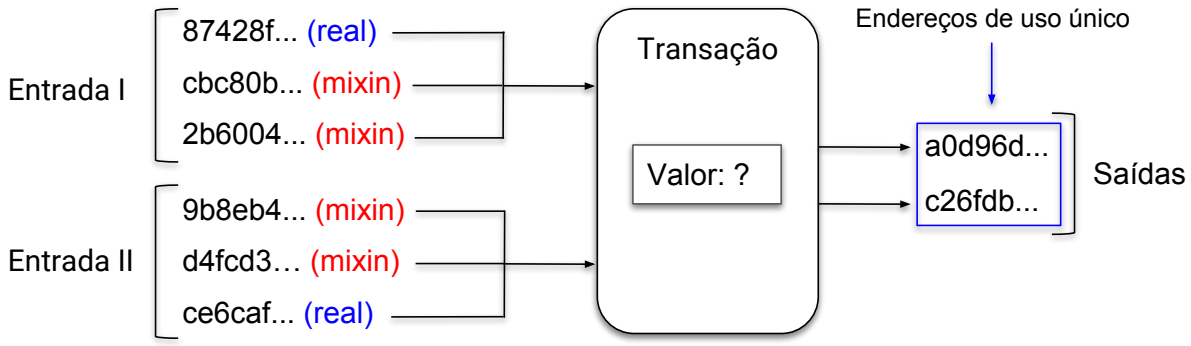
Figura 7 – Análise das transações do sistema Monero.



Adaptado: de CryptoNote (2015)

sem nenhum *mixin*. Isto gerou um problema de segurança que permite a identificação da chave real sendo utilizada nas transações. O valor da transação era constituído de diferentes tamanhos de denominações, até atingir o valor correto. Por exemplo, se um usuário precisasse enviar 16,5 XMR, a transação iria conter uma entrada de 10 XMR, seis entradas de 1 XMR e 5 entradas de 0,1 XMR, resultando no valor desejado. As RingCTs escondem o valor das transações, fazendo com que os valores de todas entradas de uma transação apareçam como 0 XMR. Uma transação pode escolher qualquer outra saída de uma transação que utilize RingCT para utilizar como *mixin*, independentemente do valor transacionado. Saídas de transações que não usam o novo protocolo não podem ser misturadas com saídas do RingCT para a realização de pagamentos.

Figura 8 – Representação de transação RingCT com duas entradas.



Apesar das garantias de privacidade oferecidas a nível de transações no sistema Monero, ainda existem maneiras pelas quais um atacante pode obter dados sobre os usuários do sistema. Uma delas é a coleta dos dados enviados pela rede durante a realização

de uma transação, que pode contribuir para a identificação dos usuários (BIRYUKOV; KHOVRATOVICH; PUSTOGAROV, 2014).

Para a resolução desse problema, foi desenvolvida a tecnologia Kovri, baseada nas especificações do *Invisible Internet Project* (I2P). Ao utilizar técnicas de roteamento e de criptografia, o protocolo Kovri estabelece uma rede sobreposta privada, permitindo que os usuários escondam suas informações geográficas e seu endereço IP. Kovri faz um tunelamento do tráfego através da rede I2P utilizando um processo chamado de *garlic routing*. As mensagens trafegam através de uma rede privada em mensagens que são criptografadas em camadas, e as únicas informações visíveis são as instruções de encaminhamento das mensagens ao longo do trajeto até o seu destino.

Além de empregar estratégias mais eficientes no que tange à privacidade dos usuários, o sistema Monero também utiliza um algoritmo de PoW chamado de CryptoNight, que objetiva ser mais igualitário do que os algoritmos utilizados em outras criptomoedas. O algoritmo CryptoNight pertence à uma classe de funções denominada *memory-bound*. Este tipo de função funciona de maneira que o tempo necessário para resolver um problema computacional depende diretamente da quantidade e da velocidade da memória disponível para armazenar os dados utilizados durante a sua resolução.

A utilização do algoritmo CryptoNight tem como motivo combater o uso de circuitos integrados de aplicação específica, do inglês *Application Specific Integrated Circuit*s (ASICs), que tornaram-se quase essenciais para usuários que desejam participar do processo de PoW da Bitcoin. Esse tipo de *hardware* é desenvolvido especificamente para permitir que um minerador compute *digests* de blocos com uma velocidade muito superior à velocidade alcançável em *hardware* de propósito geral, presente em computadores comuns. Alguns dos ASICs mais poderosos destinados ao sistema Bitcoin, como o *ANTMINER S9 Hydro* da fabricante Bitmain, possuem uma capacidade de processamento de até 18.000.000.000 (18 trilhões) de *hashes* por segundo, enquanto um processador Intel Core i7-3930k consegue computar cerca de 98.000 *hashes* por segundo (COINTOPPER, 2018).

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

O objetivo da utilização de um algoritmo *memory-bound* para computar os *digests* durante o processo de criação dos blocos é evitar que usuários precisem adquirir *hardware* especializado para participar do processo de PoW. O algoritmo CryptoNight baseia-se em acessos aleatórios à memória e possui ênfase na latência de acesso. Além disso, o algoritmo

requer cerca de 2 Megabytes de armazenamento por instância, o que significa que os dados cabem em uma memória *cache* L3 em processadores modernos (CRYPTONOTE, 2015). A velocidade das memórias presentes em GPUs e ASICs é muito inferior à velocidade das memórias *cache* de um processador e isso diminui a eficiência desses dispositivos ao executar algoritmos *memory-bound*. Essas características fazem com que processadores comuns sejam os dispositivos ideais para computar *hashes* utilizando o algoritmo CryptoNight, permitindo que usuários participem de maneira competitiva do processo de mineração da Monero sem a necessidade de adquirir *hardware* especializado.

O conjunto de algoritmos e protocolos utilizados pela criptomoeda Monero contribui para o fornecimento de segurança e privacidade para os usuários. Entretanto, assim como em qualquer sistema que dependa da sua capacidade de proteger os usuários, é necessário que as técnicas empregadas no sistema Monero sejam recorrentemente analisadas em profundidade, a fim de identificar possíveis falhas e contribuir com o desenvolvimento de criptomoedas privadas e descentralizadas.

2.2.5 Mineração via navegadores

A idéia de inserir códigos de mineração de criptomoedas em páginas da *web* surgiu logo após o lançamento da Bitcoin (ESKANDARI et al., 2018). Pouco tempo após a proliferação da idéia, várias aplicações de mineração desenvolvidas com a linguagem de programação JavaScript tornaram-se populares como: JSMiner, MineCrunch, Tidbit e BitcoinPlus. O processo de mineração utilizando navegadores foi divulgado como uma alternativa à exibição de propagandas para a monetização de conteúdos em páginas da *web*. Ao visitar uma página que contém uma aplicação de mineração, os recursos computacionais do usuário são utilizados para computar *digests* de blocos de transações, uma etapa essencial na criação de blocos e obtenção de recompensas em sistemas de criptomoedas. Esse tipo de serviço possui a vantagem de funcionar em qualquer plataforma, bastando que os usuários estejam utilizando um navegador que permita a execução de códigos JavaScript.

Algumas páginas *web* defendem o uso benigno de mineradores baseados em código JavaScript para a geração de lucro e manutenção de sua infraestrutura (SAAD; KHORMALI; MOHAISEN, 2018), solicitando que os usuários emprestem seu poder computacional em troca de acesso ao conteúdo da página. Entretanto, o uso dessas aplicações por usuários maliciosos difundiu-se rapidamente devido à facilidade de execução de ataques, necessitando apenas que as vítimas possuam uma conexão com a Internet e visitem uma página da *web* infectada com o código do minerador desenvolvido em JavaScript.

Após a difusão inicial da idéia e o surgimento de diversas aplicações de mineração via navegadores, desenvolvedores e páginas da *web* que compactuavam com o uso desse tipo de aplicações passaram a enfrentar problemas judiciais devido ao uso não autorizado dos recursos computacionais dos usuários (BLATTBERG, 2014). Além disso, com

a popularização da Bitcoin e o crescimento da base de usuários do sistema a dificuldade de mineração de novos blocos passou a aumentar rapidamente. O aumento constante da dificuldade, somado ao fato de que as aplicações para mineração desenvolvidas em JavaScript possuiam um desempenho inferior às aplicações nativas para *desktops*, desencorajou a utilização de aplicações de mineração via navegadores. Páginas *web* que faziam uso dessas aplicações obtiam pouco retorno monetário e estavam sujeitas a envolverem-se em problemas judiciais.

No ano de 2017, anos após a primeira onda de aplicações de mineração codificadas em JavaScript, as atividades de mineração em navegadores aumentaram expressivamente devido ao lançamento de um novo serviço de mineração de criptomoedas em plataformas *web*, chamado de Coinhive (RAUCHBERGER et al., 2018). De volta com a premissa do uso do poder computacional dos usuários para gerar lucros e fornecer uma alternativa à exibição de propagandas, o código de mineração do serviço Coinhive chegou a estar presente em cerca de 32.000 *websites* distintos em 2017 (KREBS, 2018). Desta vez, o foco da mineração deixou de ser a Bitcoin e voltou-se para a criptomoeda Monero. Um dos principais fatores que contribuiram para a escolha da Monero foi o seu algoritmo de PoW, CryptoNight, projetado para permitir que computadores com processadores de propósito geral sejam adequados para participar do processo de criação de blocos. Ao permitir que os processadores presentes em computadores comuns sejam capazes de calcular *hashes* de blocos de maneira mais eficiente do que *hardwares* especializados, o algoritmo CryptoNight torna viável a mineração de Monero através de códigos embutidos em páginas *web*.

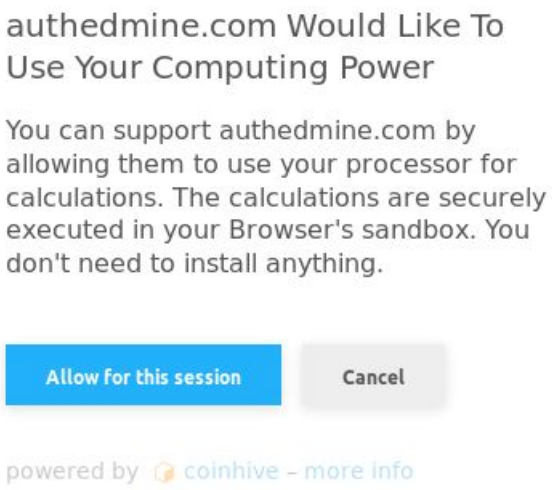
Além de servir como uma alternativa para a monetização de conteúdo, o serviço Coinhive despertou novamente o interesse de atacantes. A mineração de criptomoedas através de navegadores sem o consentimento dos usuários (*in-browser cryptojacking*) tornou-se novamente uma forma de ataque proeminente. Ataques de *cryptojacking* consistem na utilização dos recursos computacionais de uma ou mais vítimas, sem o seu consentimento, para a realização do processamento necessário à criação de blocos válidos em criptomoedas. No ano de 2017, o aumento na detecção de ataques de *in-browser cryptojacking* foi de 8.500% (MATHUR, 2018). Em junho do ano de 2018, a plataforma Coinhive foi responsável por 1,18% dos blocos minerados no sistema Monero (RÜTH et al., 2018). Um dos fatores que contribuiram para o crescimento do número de ataques foi o conjunto de mecanismos de privacidade oferecidos pelo sistema Monero. Ao infectar páginas da *web* com scripts de mineração maliciosos, um atacante é capaz de obter lucro ao receber pagamentos em criptomoedas pela validação de transações. Ao usar o sistema Monero os atacantes permanecem anônimos graças ao sistema de chaves de uso único.

Estima-se que 10 milhões de usuários sejam afetados por ataques de *in-browser cryptojacking* a cada mês (HONG et al., 2018). Páginas da *web* nas quais os usuários permanecem por longos períodos de tempo como plataformas de *streaming* de vídeo, são alvos ideais para a injeção de scripts de *cryptojacking*, visto que o lucro obtido através de

mineração é proporcional ao tempo que os usuários permanecem em uma página infectada.

Após reconhecer a importância de solicitar o consentimento dos usuários para a execução de scripts de mineração em *browsers*, a plataforma Coinhive lançou um serviço denominado AuthedMine. Esse serviço pergunta aos usuários se estes aceitam emprestar o seu poder computacional para a realização de cálculos antes de começar a computar *hashes*. No entanto, esse novo serviço gerou novas discussões à respeito da ética envolvida no uso do poder computacional dos usuários. A prática é controversa mesmo quando há consentimento, pois não fica claro se o usuários entendem o que estão consentindo ou o motivo pelo qual isso é necessário (ESKANDARI et al., 2018). A Figura 9 mostra a mensagem que solicita aos usuários o uso dos seus recursos computacionais. Embora, de um ponto de vista ético, o serviço AuthedMine seja uma alternativa mais apropriada do que os scripts de mineração padrão (sem consentimento), serviços que não requerem permissão continuam populares por causa do interesse de administradores maliciosos de páginas *web* e atacantes.

Figura 9 – Janela de confirmação para execução do serviço AuthedMine.

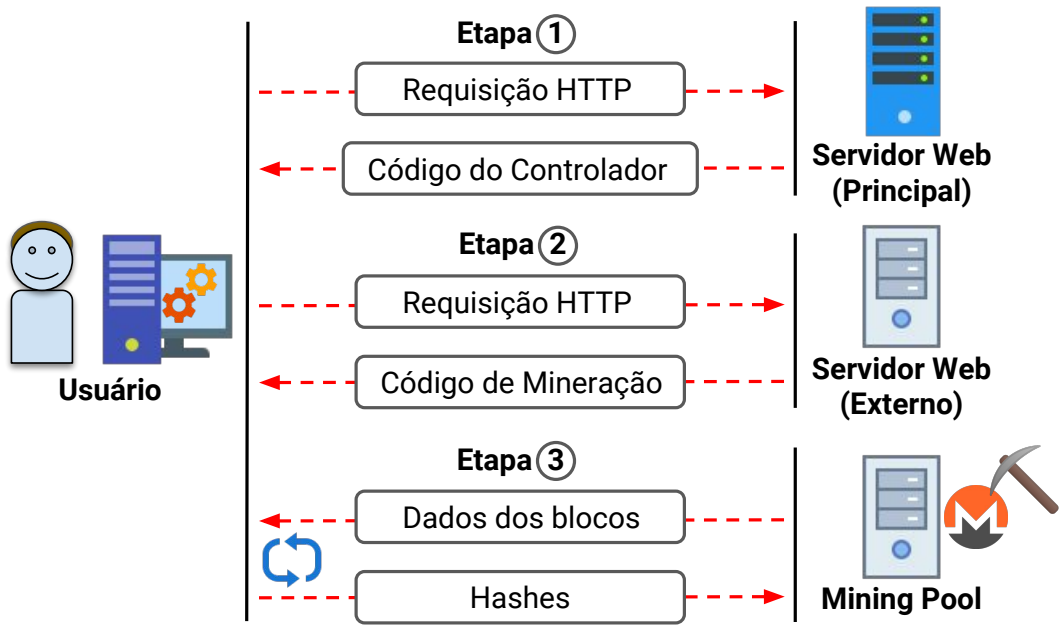


A execução de um ataque de *in-browser cryptojacking* geralmente depende de dois componentes distintos, um código controlador e um código de mineração. A Figura 10 ilustra o processo de comunicação necessário entre a máquina do usuário e os servidores do atacante para iniciar ao processo de mineração. A primeira etapa da comunicação ocorre quando um usuário visita uma página que executa um código de mineração. Ao comunicar-se pela primeira vez com a página, o navegador do usuário recebe e executa um código Javascript que inspeciona os recursos computacionais disponíveis em sua máquina. O script também verifica se o navegador da vítima suporta a execução de código WebAssembly (Wasm).

O formato de instruções Wasm é otimizado para permitir a execução de código em navegadores com um desempenho próximo ao de uma aplicação sendo executada nativamente na máquina (WEBASSEMBLY, 2018). Na próxima etapa o script comunica-se com um servidor externo definido pelo atacante, isto é, um servidor diferente do qual o usuário acessou. Se o navegador suporta a execução de código Wasm, o script do controlador faz o download de um código Wasm que é compilado na máquina da vítima, caso contrário, é realizado o download de instruções `asm.js`, um tipo de código JavaScript focado em desempenho.

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

Figura 10 – Diagrama ilustrando as etapas um ataque de *in-browser cryptojacking*.



Fonte: Adaptado de (KONOTH et al., 2018).

Ataques de *in-browser cryptojacking*, quando configurados para utilizar apenas uma fração do poder de processamento das vítimas, são pouco intrusivos e dificilmente perceptíveis. Na intenção de preveni-los, foram criadas extensões de navegadores como NoCoin (NOCOIN, 2018) e MinerBlock (MINERBLOCK, 2018) que mantêm listas de sites infectados e bloqueiam o acesso à estes sites. Porém, listas de bloqueio são ineficientes pois requerem constante manutenção para a adição de novas páginas, podem gerar falsos positivos e podem ser contornadas através de técnicas de evasão de detecção (SEGURA,

2018). Após a popularização das listas de bloqueio os atacantes passaram a utilizar diversas técnicas de evasão:

- **Servidores de retransmissão:** Ao configurar seu próprio servidor externo, um atacante é capaz de usá-lo como intermediário para a transmissão de dados até *mining pools*. O código malicioso comunica-se com o servidor do atacante, que é desconhecido e dificilmente será detectado através da análise de assinaturas maliciosas, e este comunica-se com *mining pools*, requisitando trabalho e enviando os *digests* computados.
- **Ofuscação de código:** Alguns códigos de *cryptojacking* tentam ofuscar os comandos exectuados pelo script utilizando técnicas como a utilização de funções de codificação e decodificação de *strings*.
- **Artifícios anti-depuração:** Códigos que utilizam artifícios anti-depuração verificam se estão sendo analisados através das ferramentas de depuração para desenvolvedores do navegador. Se estiver sendo analisado o código interrompe a sua execução.
- **Ofuscação do protocolo de comunicação:** Para dificultar a sua detecção alguns códigos codificam ou cifram as mensagens de comunicação com *mining pools*, enviadas através de um protocolo chamado Stratum.
- **Ocultação de *payload*:** Alguns atacantes não inserem seus códigos maliciosos diretamente nas páginas sendo atacadas, optando por incluí-los dentro de suas próprias versões de arquivos contendo código de bibliotecas já conhecidas. No momento do carregamento das bibliotecas nas páginas *web*, o código malicioso também é inicializado.
- **Controle do uso do processador:** Se um código malicioso utiliza todos os recursos computacionais da máquina da vítima, o dispositivo pode sofrer travamentos e lentidão, permitindo a identificação da aplicação de mineração. Alguns scripts limitam a sua utilização dos recursos da máquina da vítima, muitas vezes passando despercebidos em meio à outras aplicações que estão sendo executadas no momento, e.g. navegador e gerenciador de janelas.

Estima-se que a utilização de plataformas de mineração como Coinhive não gere um retorno monetário tão expressivo quanto a exibição de propagandas em páginas da *web* (SAAD; KHORMALI; MOHAISEN, 2018). A menos que as páginas utilizadoras de mineração em navegadores atraiam muitos usuários e ofereçam conteúdos que os mantenham no *site* por longos períodos de tempo, a utilização desse tipo de serviço não é recomendado. Outro fator que influencia no retorno das páginas que contêm mineradores é a quantidade de recursos utilizados durante o processo. Utilizar todo o poder

computacional dos usuários pode prejudicar as suas experiências ao navegar na própria página, impactando negativamente na sua popularidade. No entanto, atacantes podem beneficiar-se economicamente de ataques de *in-browser cryptojacking* ao infectar páginas que oferecem entretenimento e conteúdo envolvendo pirataria, pois usuários passam mais tempo procurando por recursos nessa categoria de páginas (KONOTH et al., 2018; HONG et al., 2018).

3 TRABALHOS RELACIONADOS

Trabalhos presentes na literatura propõem técnicas de ataques à criptomoedas com o intuito de prejudicar a rede ou revelar informações sobre as entidades envolvidas nas transações. Alguns trabalhos baseiam-se na interceptação de tráfego de rede (APOSTOLAKI; ZOHAR; VANBEVER, 2017; BIRYUKOV; KHOVRATOVICH; PUSTOGAROV, 2014), outros realizam análises baseando-se somente nos dados contidos no histórico de transações do *blockchain* (KUMAR et al., 2017; MILLER et al., 2017; RON; SHAMIR, 2013). A seguir são apresentadas algumas estratégias utilizadas para a deanonimização de dados, apresentadas em trabalhos com temas relacionados ao tema deste trabalho.

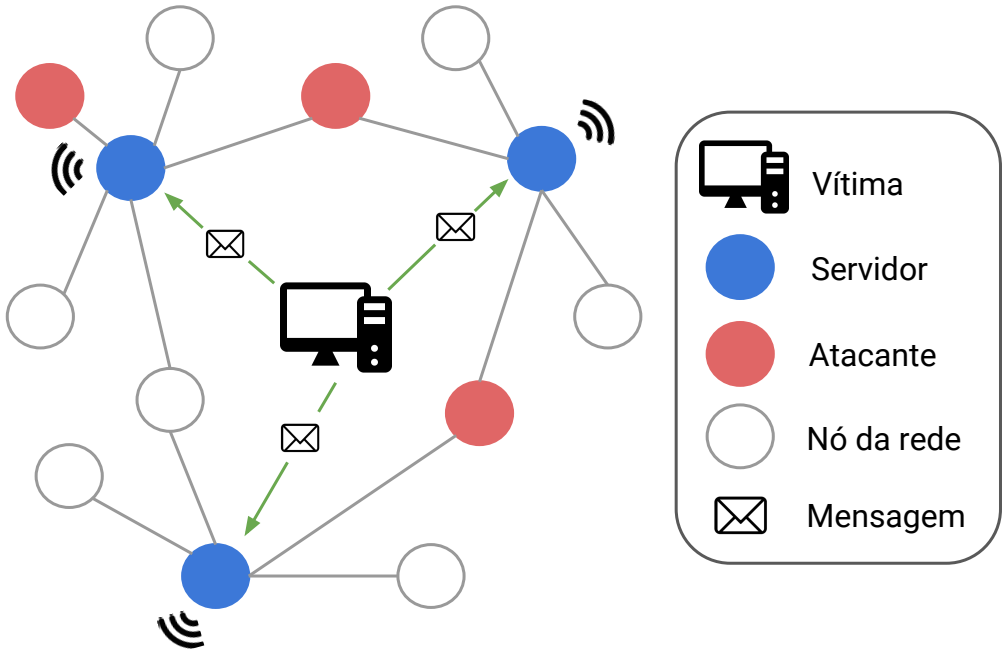
Utilizando heurísticas que exploram vulnerabilidades do sistema Monero, é possível analisar dados de transações presentes no *blockchain* da criptomoeda e rastrear as saídas de transações utilizadas como entradas em pagamentos (KUMAR et al., 2017; MILLER et al., 2017). A análise de dados de transações que datam desde o primeiro bloco da criptomoeda, revela uma falha no controle do número de *mixins* utilizados nas transações, permitindo que as entradas de transações que não utilizam nenhum *mixin* sejam rastreadas de maneira trivial. Entradas rastreadas também causam uma reação em cadeia que permite rastrear transações posteriores. A extração de informações sobre os blocos que deram origem a chaves presentes no *blockchain* possibilita a aplicação de heurísticas que realizam uma análise temporal dos blocos sobre os dados. Ao explorar uma característica da distribuição de escolha de *mixins*, que gera resultados tendenciosos, a heurística é capaz de identificar a chave real da transação com mais de 90% de certeza.

A coleta de dados sobre comunidades de mineradores que trabalham em conjunto fornece informações que ajudam a quantificar as práticas de mineração de criptomoedas (MILLER et al., 2017). Informações sobre as transações que remuneram participantes das comunidades de mineração são divulgadas com o objetivo de fornecer transparência à essa categoria de serviço. A análise destas informações permite que pesquisadores estimem a quantidade de blocos criados por algumas das maiores comunidades de mineração da criptomoeda Monero.

Chaves públicas usadas em transações do sistema Bitcoin podem ser correlacionadas com endereços IP públicos de usuários (BIRYUKOV; KHOVRATOVICH; PUSTOGAROV, 2014). Este ataque baseia-se na identificação dos clientes através dos dados dos nós de entrada aos quais os mesmos conectam-se. Para potencializar o efeito do ataque, primeiro deve ser realizado um procedimento para bloquear conexões de nós da rede Tor com a rede Bitcoin. Isso pode ser feito por intermédio da criação de um grande número de conexões com a rede Tor, selecionando para cada conexão um nó de saída distinto. A partir das conexões Tor, são disseminadas mensagens malformadas, fazendo com que a rede Bitcoin penalize os nós de saída até que eles sejam banidos. Este passo prejudica a comunicação de nós anônimos com a rede, forçando os usuários a conectarem-se utilizando seu endereço IP real. Os atacantes obtêm o endereço IP público dos usuários

a partir da inspeção das mensagens que são enviadas por eles na rede. A identificação dos usuários ocorre através da exploração do modelo de disseminação de mensagens do protocolo Bitcoin. A Figura 11 ilustra a organização da rede durante o ataque.

Figura 11 – Organização da rede durante o ataque de interceptação de tráfego.



Os atacantes executam instâncias do cliente do sistema Bitcoin e estabelecem conexões com o maior número de servidores possíveis. Quando um novo participante entra na rede, os servidores disseminam suas informações para que os outros participantes possam conectar-se com o novo nó. Os atacantes armazenam as informações do novo participante juntamente com as informações dos servidores que disseminaram a mensagem. Quando uma nova transação é compartilhada na rede, o atacante cria uma lista dos servidores que disseminaram a informação. Usando os dados obtidos anteriormente sobre os participantes da rede, os atacantes procuram por um usuário que esteja conectado exatamente aos servidores que disseminaram a mensagem. Ao encontrar o participante que está conectado aos mesmos servidores, há uma grande probabilidade de que seja ele quem gerou a transação. O endereço IP do usuário é identificado como a origem da transação recém criada.

Um grafo do histórico de transações da Bitcoin pode ser construído com os dados disponíveis no *blockchain* da criptomoeda. O grafo revela as atividades dos usuários na rede, incluindo a quantidade de transações, distribuição de entidades e estatísticas de uso da criptomoeda (RON; SHAMIR, 2013). Variações do algoritmo *Union-Find* podem ser utilizadas para encontrar diferentes endereços pertencentes à um mesmo usuário ou serviço. O grafo de transações permite visualizar o fluxo de moedas na rede e as maiores transações já efetuadas desde o lançamento da Bitcoin. Adicionalmente, ao investigar

o grafo de transações é possível identificar padrões de comportamento de usuários que transacionam grandes quantidades de criptomoedas e realizam procedimentos para dificultar o rastreamento do histórico das transações. Grafos de transações também podem ser utilizados na classificação de nós da Bitcoin de acordo com o seu nível de importância (FLEDER; KESTER; PILLAI, 2015; MEIKLEJOHN et al., 2013). As entidades que possuem o maior número de conexões no grafo são consideradas como mais importantes, pois participaram de uma quantidade maior de transações. O algoritmo *PageRank* (PAGE et al., 1999), utilizado comumente em mecanismos de busca para classificar páginas da *web*, pode ser usado para auxiliar na classificação dos nós do grafo.

Endereços de nós presentes no grafo de transações da Bitcoin podem ser correlacionados com informações de usuários coletadas em páginas da *web* (FLEDER; KESTER; PILLAI, 2015; MEIKLEJOHN et al., 2013). A utilização de uma técnica chamada de *web scraping*, que usa algoritmos para vasculhar páginas da *web* automaticamente, possibilita a coleta de informações em fóruns, redes sociais e plataformas de doação. Buscas são realizadas com o objetivo de encontrar informações de usuários que divulgaram suas chaves públicas da rede Bitcoin. Com os dados coletados é possível identificar os nós no grafo de transações da Bitcoin, associando as chaves presentes no grafo aos usuários através das informações encontradas na *web*.

A alteração do código do *software* de carteira do sistema Monero possibilita a exploração de uma vulnerabilidade que permite a criação de transações rastreáveis (WIJAYA et al., 2018). Após a escolha das chaves de uma transação, o sistema Monero não verifica se chaves incluídas como *mixins* em uma entrada de transação já foram incluídas como chaves reais em outra entrada na mesma transação. Isso permite a criação de várias entradas na mesma transação com um mesmo conjunto de chaves, onde, em algumas entradas, as chaves são incluídas como *mixins* e em uma delas a chave é utilizada para o pagamento. Desta forma, todas as chaves usadas como *mixins* poderão ser identificadas como gastas por outros usuários, diminuindo a privacidade das transações nas quais elas forem utilizadas posteriormente.

A Tabela 2 apresenta uma comparação dos trabalhos relacionados. As colunas da tabela representam as estratégias utilizadas nos trabalhos para fins de rastreio de dados e identificação dos usuários. Cada uma das estratégias utilizadas nos trabalhos relacionados serve propósitos diferentes. A combinação de técnicas como por exemplo, *web scraping* e análise do grafo de transações, pode contribuir para a geração resultados melhores, causando a identificação de um número maior de usuários.

- Análise dos dados de transações: Consiste na extração das informações sobre as transações armazenadas no *blockchain*. Após a extração dos dados, podem ser aplicadas as heurísticas de ataque. A análise dos dados de transações é fundamental para a criação de grafos de transações.

Tabela 2 – Comparação dos trabalhos relacionados.

	Análise dos dados de transações	**Análise do tráfego de rede**	**Análise do grafo de transações**	***Web Scraping***	**Agrupamento de endereços**	**Recriação do efeito de zero *mixins***
(KUMAR et al., 2017)	X					
(MILLER et al., 2017)	X					
(BIRYUKOV; KHOVRATOVICH; PUSTOGAROV, 2014)	X	X				
(RON; SHAMIR, 2013)	X		X		X	
(FLEDER; KESTER; PILLAI, 2015)	X		X	X	X	
(MEIKLEJOHN et al., 2013)	X		X	X	X	
(WIJAYA et al., 2018)	X					X

- Análise do trafego de rede: Realizada através da interceptação do tráfego de rede gerado pelos usuários conectados aos servidores do sistema. Quando um nó entra na rede, ele solicita aos seus nós de entrada, com os quais estabeleceu conexão, que disseminem seu endereço IP para que outros nós possam adicioná-lo à suas listas de conexões. Ao escolher um usuário para atacar, monitora-se quais nós disseminam o endereço desse usuário. Depois de conhecer os nós de entrada do usuário, as transações disseminadas por estes nós são atribuídas aos usuários que estejam conectados a eles. Com os endereços e as transações, um endereço IP pode ser associado à um pseudônimo do sistema. Esta estratégia é útil para descobrir os endereços públicos dos usuários do sistema. Entranto, vale ressaltar que o volume de usuários identificados com o uso desta técnica é pequeno, pois é necessário aguardar até que as vítimas enviem dados de transações pela rede para que o ataque possa ser realizado.

- Análise do grafo de transações: O grafo de transações é criado utilizando os endereços públicos dos usuários do sistema. Quando um usuário efetua uma transação, é criada uma conexão entre o endereço do remetente e o endereço do recipiente. O grafo representa as transações realizadas e a interação entre os usuários do sistema.

- *Web Scraping:* Esta estratégia é utilizada para a obtenção de informações de usuários disponíveis na *web*. Dados de usuários que compartilham suas chaves públicas são buscados em fóruns e sites de venda de moedas. Esta técnica pode ser utilizada para a identificação de nós após a construção do grafo de transações do sistema.

- Agrupamento de endereços: Facilita a visualização e a organização dos dados do grafo de transações. Ao construir o grafo de transações, endereços diferentes pertencentes à um mesmo usuário são representados por nós distintos. O agrupamento permite representar todos os endereços do mesmo usuário em um único nó do grafo. Isto contribui para a identificação das relações entre usuários distintos e permite quantificar as atividades dos usuários.

- Recriação do efeito de zero *mixins*: A alteração do código de carteiras de criptomoedas permite que atacantes efetuem transações com dados modificados. Ao burlar restrições no momento de criação de entradas de transações, pagamentos cujas chaves reais são visíveis no *blockchain* podem ser criados. Estas chaves podem ser identificadas como gastas por qualquer um que analise os dados das transações.

4 DESENVOLVIMENTO

Para a realização deste trabalho, foram investigados trabalhos relacionados e conduzidos experimentos e análises com base em técnicas de ataque existentes. Em um primeiro momento, foram realizadas pesquisas na literatura buscando por trabalhos que abordam segurança e privacidade em sistemas de criptomoedas, sejam elas focadas em privacidade dos usuários ou não. Os trabalhos encontrados exploram vulnerabilidades de sistemas de criptomoedas para identificar usuários. Duas estratégias de ataque contra a criptomoeda Monero, abordadas na literatura, foram selecionadas para serem replicadas. Os resultados obtidos com a replicação contribuíram para a demonstração da existência de vulnerabilidades no sistema.

No desenvolvimento deste trabalho foram utilizados o gerenciador de repositórios GitLab (GITLAB TEAM, 2018) e o sistema de controle de versões Git (SOFTWARE FREEDOM CONSERVANCY, 2018) para realizar o versionamento do código e auxiliar na organização dos arquivos. Para a replicação dos ataques, foi necessária a extração dos dados do *blockchain* da criptomoeda Monero. O *software* de cliente para a linha de comando do sistema Monero, versão 0.11.1.0, foi utilizado na sincronização dos dados locais com os dados das transações disponíveis na rede. O acesso aos dados do *blockchain* foi realizado através da ferramenta *Onion Monero Blockchain Explorer*, disponível no repositório de exemplos do Monero no GitHub. Esta ferramenta serve como uma interface de acesso aos dados das transações, retornando os dados solicitados atráves de consultas utilizando a API *Javascript Object Notation* (JSON). A extração dos dados dos blocos foi realizada com scripts na linguagem de programação Python versão 3.6 e as heurísticas de ataque foram codificadas na linguagem de programação C++. Detalhes acerca dos algoritmos utilizados e dos resultados obtidos são apresentados no Capítulo 5.

Após a realização das análises de rastreabilidade, foi conduzida uma avaliação das estratégias de ataque apresentadas nos trabalhos relacionadas com o intuito de identificar novos possíveis vetores de ataque ao sistema Monero. Para este fim, foram levados em consideração os mecanismos de defesa existentes. A Tabela 3 sumariza as estratégias de ataque apresentadas em trabalhos relacionados e os mecanismos de defesa que inviabilizam suas execuções.

Dos sete ataques apresentados, quatro são ineficazes contra o sistema Monero, devido aos mecanismos de privacidade oferecidos pelo protocolo CryptoNote. São eles: análise do grafo de transações, análise do tráfego de rede, *web scraping* e agrupamento de endereços. Entretanto, estes quatro ataques são eficazes contra a Bitcoin. Isto deve-se ao fato de que os ataques contra a Bitcoin exploram principalmente o conhecimento das chaves públicas dos usuários, que são imutáveis.

Os ataques de análise de *mixins*, análise temporal e recriação do efeito de zero *mixins* são direcionados à criptomoeda Monero. A análise temporal permite rastrear entradas de transações antigas, porém, tornou-se ineficaz contra transações recentes após

Tabela 3 – Estratégias de ataque e respectivos mecanismos de defesa.

Estratégia de ataque	**Mecanismos de defesa**
Análise de *mixins*	Aumento do número mínimo de *mixins*
Análise temporal	Novo algoritmo de seleção de *mixins*
Análise do grafo de transações	Endereços de uso único; Protocolo RingCT
Análise do tráfego de rede	Tecnologia Kovri
Web Scraping	Endereços de uso único
Agrupamento de endereços	Endereços de uso único; Protocolo RingCT
Recriação do efeito de zero *mixins*	Nenhum

uma atualização no algoritmo de seleção de *mixins* do sistema. A heurística de análise de *mixins* também é eficaz contra transações antigas, mas teve seu impacto mitigado após a elevação do número mínimo de *mixins* exigidas em cada entrada de transação. O ataque de recriação do efeito de zero *mixins* explora a falta de verificação da corretude das chaves presentes em uma transação, permitindo que uma chave sendo gasta em uma entrada seja utilizada como *mixin* em outra entrada dentro da mesma transação. Dessa forma é possível tornar chaves rastreáveis, diminuindo a privacidade das transações.

A partir da análise realizada, foi possível concluir que todos os ataques contra a criptomoeda Monero baseiam-se na exploração da premissa da irrastreabilidade de transações. Este tipo de ataque possui como objetivo reduzir o número de *mixins* presentes nos conjuntos de chaves de entrada de transações. A identificação de chaves já utilizadas permite que estas sejam removidas das entradas de novas transações, onde são usadas para ofuscar a chave real, reduzindo a privacidade das operações e no melhor dos casos permitindo identificar a chave real.

Levando em consideração os dados obtidos nas etapas anteriores do desenvolvimento, foi elaborado um novo ataque que explora vulnerabilidades nos mecanismos que tornam as transações irrastreáveis. O ataque baseia-se em efetuar um grande número de transações de modo que o atacante controle grandes parcelas das chaves de saída de transações presentes no *blockchain*. Foram realizadas simulações utilizando a linguagem de programação Python e as mesmas ferramentas de exploração do *blockchain* utilizadas nas análises preliminares. Uma discussão detalhada do ataque proposto e os resultados obtidos através das simulações são apresentados no Capítulo 6.

5 ANÁLISE DE RASTREABILIDADE

Neste capítulo são apresentados resultados obtidos com a implementação e verificação de ataques ao sistema Monero, utilizando estratégias existentes na literatura. O objetivo da realização destes ataques foi a extração dos dados do *blockchain*, exploração de falhas, e familiarização com os mecanismos de privacidade utilizados pelo sistema Monero. A realização dos ataques contribuiu para a identificação e simulação do ataque detalhado no Capítulo 6.

5.1 Extração de Dados

Foram extraídos do *blockchain* os dados de transações a partir do primeiro bloco, datado de 18 de Abril de 2014, até o bloco de número 1.514.000, criado no dia 20 de Fevereiro de 2018. No total, os 1.514.000 blocos extraídos contêm 3.976.181 transações, e destas, 1.514.000 são transações *coinbase*, que não possuem nenhuma entrada e remuneram mineradores pela criação do bloco. A primeira transação de um bloco é sempre uma transação *coinbase*. As informações extraídas foram armazenadas em um arquivo no formato CSV. Cada um dos blocos é composto pelos seguintes campos:

- Número do bloco
- *Timestamp* da criação
- Número de transações contidas no bloco
- Para cada transação:
  - *Hash* da transação
  - Número de entradas da transação
  - Número de *mixins* em cada entrada
  - Endereços de entrada da transação
  - Endereços de saída da transação
  - Valores de saída da transação em XMR

Estas informações constituem o conjunto de dados utilizados na realização da análise descrita na seção 5.2.

5.2 Análise dos Dados

Uma análise das transações do sistema Monero foi realizada utilizando heurísticas apresentadas em trabalhos existentes na literatura. Os ataques exploram o princípio de irrastreabilidade, apresentado na subseção 2.2.4, o princípio de não-vinculação de endereços permanece inafetado. Os resultados obtidos demonstram a existência de problemas

de privacidade nas transações efetuadas antes do lançamento do protocolo RingCT. Após o ínicio do uso das RingCTs, a porcentagem de transações afetadas pelas heurísticas começou a ser reduzida, pois o novo protocolo torna as transações resistentes aos ataques apresentados.

Duas heurísticas de rastreio foram selecionadas para serem aplicadas aos dados extraídos. A primeira heurística, apresentada na subseção 5.2.1, baseia-se na exploração de uma falha de implementação do sistema para descobrir as entradas reais sendo utilizadas nas transações. A segunda heurística, apresentada na subseção 5.2.2, explora uma deficiência no algoritmo de escolha de *mixins* para descobrir as entradas reais das transações.

Algumas ocorrências interessantes foram observadas nos dados do *blockchain* durante a análise dos dados:

1. No bloco de número 8.330, a primeira transação não *coinbase* possui 10 entradas, e nove destas possuem um *mixin* além da chave real. Curiosamente, a entrada remanescente, que não utiliza *mixins*, possui um valor dezenas de vezes maior do que as outras. Isto é provavelmente um erro do sistema, pois todas as entradas de uma transação devem possuir o mesmo número de *mixins*.

2. O bloco de número 202.612 possui 514 transações no total. Destas, 511 têm como valor de entrada 1 piconero(10^{-12}), a menor unidade da moeda. Nenhuma das 511 transações gera alguma saída. As duas últimas transações do bloco são transações normais e contêm valores de saída e a primeira é uma transação de remuneração pela criação do bloco. Este bloco foi criado através da exploração de uma vulnerabilidade no código do sistema Monero. A vulnerabilidade residia em uma função de arredondamento de potências de dois, permitindo que o *digest* de blocos com mais de 512 transações fosse computado utilizando áreas de memória não inicializadas, gerando dados arbitrários. Através desta vulnerabilidade um usuário foi capaz de criar dois blocos com 514 transações que possuiam o mesmo *digest*, porém as duas últimas transações eram diferentes em cada um deles. Isso causou uma divisão na rede que teve de ser resolvida manualmente ao incluir no código-fonte um trecho de código para sobrescrever o *hash* do bloco.

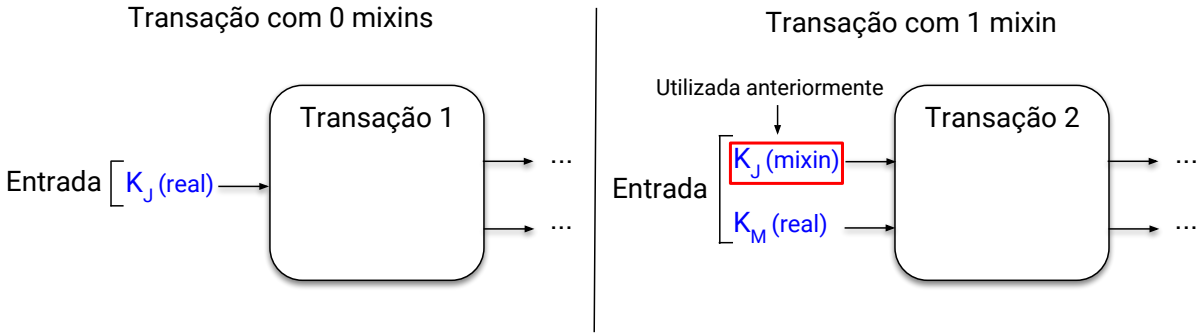
3. No bloco de número 981.002, a mesma chave aparece como saída nas duas transações do bloco. Isto não deve ocorrer, uma vez que todas as chaves de saída são geradas de forma que sejam únicas. Esta transação foi provavelmente criada através da exploração de um problema no código que permitia que a mesma saída de transação pudesse ser enviada duas vezes. O dinheiro enviado com a primeira chave poderia ser utilizado mas o dinheiro contido na segunda chave era perdido. Essa falha foi denominada de *The Burning Bug* fazendo alusão ao dinheiro perdido nas transações.

5.2.1 Análise de *Mixins*

A primeira estratégia de ataque aplicada foi a análise de transações com zero *mixins* (MILLER et al., 2017; KUMAR et al., 2017). Este ataque explora uma falha que permite efetuar transações sem a utilização de *mixins*. Isto ocorre por escolha do usuário ou pela falta de transações anteriores com valores iguais aos valores sendo transacionados. Mais tarde, o uso de denominações foi introduzido como meio de mitigar a falta de entradas com o mesmo valor, conforme discutido na subseção 2.2.4.

A falta de *mixins* em uma transação torna suas entradas reais visíveis a qualquer atacante que possua acesso aos dados do *blockchain*. A não utilização de *mixins* gera uma reação em cadeia que permite descobrir a entrada real de transações futuras. Isto afeta outras transações além da própria transação sem *mixins*. A Figura 12 ilustra o efeito que transações sem *mixins* desencadeam em outras transações, onde a primeira transação é realizada sem a utilização de qualquer *mixin* e posteriormente, a mesma chave é usada como *mixin* em outra transação. Suponha que João efetue uma transação com uma única entrada K_J e nenhum *mixin*. Esta entrada é sabidamente a chave sendo utilizada na transação. Agora suponha que Maria efetue uma transação com apenas um *mixin* além da chave verdadeira, K_M, e que o sistema escolha a chave K_J para ser utilizada como *mixin*. Neste caso, como a chave K_J já foi utilizada, é possível deduzir que a entrada real da transação é a chave K_M.

Figura 12 – Efeito em cadeia de transações com 0 *mixins*.



Além das transações que são afetadas pela quebra do princípio de irrastreabilidade, outras transações são afetadas, tendo o tamanho efetivo do seu conjunto de *mixins* diminuído. Considerando o conjunto de *mixins* para uma entrada em uma transação, para cada *mixin* sabidamente já utilizado presente na entrada, o tamanho do conjunto de possíveis chaves utilizadas na transação é diminuído, tornando mais fácil o processo de adivinhar a chave real.

O Algoritmo 1, adaptado do pseudocódigo da heurística de análise de zero *mixins* (KUMAR et al., 2017), descreve o processo de descoberta de chaves utilizadas em transa-

Algoritmo 1 Procedimento de análise de mixins

```
1:  procedure ANALISA-MIXINS(blocos)
2:      for each bloco ∈ blocos do                          ▷ Para cada bloco extraído
3:          transacoes ← extraiTransacoes(bloco)
4:          for each transacao ∈ transacoes do
5:              entradas ← extraiEntradas(transacao)
6:              for each entrada ∈ entradas do
7:                  chaves ← extraiChaves(entrada)
8:                  chavesParaAnalisar.add(chaves)
9:              end for
10:         end for
11:     end for
12:
13:     chavesUtilizadas ← {}
14:     while chavesUtilizadas for alterado do          ▷ Enquanto deduzir novas chaves
15:         for each entrada ∈ chavesParaAnalisar do
16:             chavesNaoRastreadas ← {}
17:             for each chave ∈ entrada do
18:                 if chave ∉ chavesUtilizadas then
19:                     chavesNaoRastreadas ← chavesNaoRastreadas ∪ chave
20:                 end if
21:             end for
22:             if |chavesNaoRastreadas| == 1 then
23:                 chavesUtilizadas ← chavesUtilizadas ∪ naoRastreadas
24:             end if
25:         end for
26:     end while
27:     return chavesUtilizadas
28: end procedure
```

ções anteriores. A primeira tarefa é extrair as chaves de entrada de cada uma das entradas presentes em cada transação. As transações contidas em cada bloco são extraídas (Linha 3) e cada uma tem as chaves de suas entradas armazenadas em uma lista de chaves que será utilizada na análise (Linhas 4-10). A variável *chavesParaAnalisar* é uma lista composta por listas menores que representam as chaves de cada uma das entradas. As chaves extraídas das entradas são compostas por todas as chaves incluídas nas transações, ou seja, as chaves reais das transações e aquelas que foram utilizadas como *mixins*. O próximo passo é realizar a análise das chaves e repetir o processo enquanto o algoritmo for capaz de marcar novas chaves como utilizadas (Linha 14). Se após uma iteração nenhuma nova chave for identificada, o algoritmo retorna as chaves deduzidas. Para cada lista de entradas presente em *chavesParaAnalisar* é criado um conjunto vazio (Linha 16). Para cada chave da entrada é realizada uma verificação. Se ela ainda não foi marcada como utilizada, é adicionada ao conjunto de chaves não rastreadas (Linhas 17-21). Se ao final da análise das chaves da entrada, apenas uma não foi rastreada, isto significa que a chave

é a única presente na entrada ou que todas as outras chaves já foram marcadas como utilizadas. Em ambos os casos, a chave restante é chave real da transação e é adicionada ao conjunto de chaves utilizadas (Linhas 22-23). Na primeira iteração do algoritmo, as chaves presentes em transações com zero *mixins* são identificadas como utilizadas. Nas iterações seguintes, o número de chaves marcadas como utilizadas cresce. Algumas transações com múltiplas entradas têm algumas de suas chaves identificadas a cada iteração até que em algum momento a chave real seja deduzida. Por fim, o conjunto de chaves identificadas como utilizadas é retornado (Linha 27).

A Tabela 4 mostra os 13 primeiros tamanhos de *mixins* utilizados em entradas de transações. São mostradas quantas entradas de transações utilizam cada uma das quantias de *mixins* e quantas dessas entradas foram deduzidas através da análise.

Tabela 4 – Frequência de mixins e quantia deduzida.

Quantidade de mixins	**Nº de entradas**	**Entradas deduzidas**	**Taxa de dedução**
0	12.207.748	12.207.748	100%
1	707.788	609.383	86,09%
2	4.496.449	1.774.903	39,47%
3	1.486.633	951.153	63,98%
4	2.966.970	446.946	15,06%
5	301.068	73.845	24,52%
6	425.726	201.659	47,36%
7	20.416	4.264	20,88%
8	27.115	3.483	12,84%
9	15.956	2.144	13,43%
10	111.064	22.863	20,58%
11	2.576	372	14,44%
12	3.793	716	18,87%

É importante observar que a taxa de dedução apresentada na Tabela 4 é afetada pela quantia de *mixins*. A probabilidade de rastreio aumenta com a redução do número de *mixins*. Outro fator que impacta na probabilidade de dedução das transações são as *mixins* escolhidas pelo sistema. Se as *mixins* escolhidas foram utilizadas em transações deduzíveis ou com nenhuma outra *mixin*, o efeito em cadeia afeta a privacidade da transação atual.

A incorporação das RingCTs ao sistema, a partir de 19 de Setembro de 2016, teve um impacto na taxa de dedução de transações com duas e quatro *mixins*. Isto ocorreu porque após o lançamento do protocolo, o número mínimo de *mixins* em uma transação foi elevado para duas e meses depois, para quatro. Como os usuários tendem a utilizar a quantidade mínima de *mixins* permitida, pois este é o valor padrão em muitas aplicações

que gerenciam endereços do sistema Monero, foi criado um grande número de transações não-rastreáveis utilizando duas e quatro *mixins*. As transações com zero, uma e três *mixins* possuem uma taxa de dedução alta porque a maioria das transações desses grupos foram efetuadas antes do lançamento do protocolo RingCT, quando as transações eram mais vulneráveis à dedução.

Para mostrar o impacto que este ataque pode causar até mesmo em transações com um número elevado de *mixins*, a Tabela 5 mostra algumas entradas com quantidades elevadas de *mixins* que tiverem suas chaves reais rastreadas pela heurística.

Tabela 5 – Entradas com número elevado de *mixins* afetadas pelo ataque.

Quantidade de mixins	**Entradas deduzidas**
50	149
70	22
90	10
100	39
153	1

Com o passar do tempo, a porcentagem de transações rastreáveis irá ser reduzida porque o uso do protocolo RingCT tornou-se obrigatório em todas as transações do sistema, a partir de setembro de 2017[1]. As transações RingCT só podem utilizar como *mixins* entradas de transações do mesmo tipo. Como o protocolo foi tornado obrigatório juntamente com uma política que requer o uso de pelo menos quatro *mixins* por entrada, não é mais possível criar transações transparentes que possam causar reações em cadeia ao utilizar o sistema normalmente. Entretanto, um atacante pode ser capaz de criar transações rastreáveis explorando vulnerabilidades do sistema, como no ataque de recriação do efeito de zero *mixins* (WIJAYA et al., 2018).

A Tabela 6 sumariza os resultados da aplicação do ataque de análise de *mixins*. O total de entradas deanonimizadas é composto pela soma do número de entradas que contêm 0 mixins mais o número de entradas que foram rastreadas através da análise de *mixins*. As entradas que contêm 0 *mixins* são sempre rastreáveis por consequência da utilização de apenas uma chave na assinatura da transação.

5.2.2 Análise Temporal

A segunda estratégia de ataque utilizada foi baseada na heurística de análise temporal (MILLER et al., 2017; KUMAR et al., 2017). A análise temporal considera que uma chave gerada por uma transação não permanece sem ser utilizada por muito tempo. Chaves geradas em transações antigas tem uma probabilidade muito maior de terem sido

[1] <https://getmonero.org/resources/moneropedia/ringCT.html>

Tabela 6 – Resumo dos resultados da análise de *mixins*.

Observável	**Quantidade**
Número total de entradas	22.845.510 (100%)
Entradas que contêm 0 mixins	12.196.416 (53,39%)
Entradas vulneráveis à dedução	4.097.846 (17,94%)
Total de entradas rastreadas	16.294.262 (71,32%)

utilizadas do que chaves geradas recentemente, portanto, é possível assumir que a chave real de uma transação é aquela cujo bloco de origem no *blockchain* é o mais recente dentre todas as entradas.

Apesar de ser uma heurística fácil de ser aplicada aos dados, a análise temporal é capaz de adivinhar corretamente a chave de mais de 90% das entradas das transações de versões antigas do sistema. Isto deve-se à uma característica da antiga distribuição matemática utilizada para a seleção de *mixins*, que favorecia a escolha de chaves antigas. A distribuição matemática utilizada na escolha de *mixins* foi atualizada em 13 de Dezembro de 2016, na versão 0.10.1 do sistema. Mudanças foram realizadas para garantir que pelo menos 50% das entradas de uma transação sejam escolhidas dentre as chaves geradas nos últimos 1,8 dias. Isto ajuda a evitar que a chave verdadeira de uma entrada seja sempre a chave mais recente.

O Algoritmo 2, baseado em (KUMAR et al., 2017; MILLER et al., 2017), mostra o pseudocódigo do procedimento de análise temporal. O mesmo processo é realizado para cada uma das entradas. Primeiro, o conjunto de chaves pertencentes a entrada é extraído e armazenado em uma lista de chaves (Linhas 3-6). Então, é identificada qual das chaves dentre as presentes na entrada é originária do bloco mais recente (Linha 7). As chaves restantes recebem uma marcação para identificá-las como *mixins*. Por fim, as entradas são retornadas com suas *mixins* marcadas (Linha 14).

Para tornar mais fácil o processo de identificação dos blocos de origem, as chaves foram armazenadas utilizando a sua posição relativa no *blockchain*. A posição relativa de uma chave é dada no formato (1)-(2)-(3), onde: (1) é o número do bloco de origem da chave, (2) é o índice da transação onde a chave aparece no bloco e (3) é o índice que representa a posição da chave nas saídas da transação. Os índices são considerados a partir de zero. Por exemplo, uma chave originária da segunda transação do bloco 645020, sendo a 3ª saída da transação é representada como 645020-1-2.

Esta heurística pode gerar falsos positivos porque baseia-se na probabilidade das chaves mais recentes serem as chaves reais, como acontece em grande parte das entradas, porém não ocorre para todas. Para medir a taxa de acertos da heurística, a análise temporal foi aplicada às mesmas transações que foram rastreadas com sucesso pela heurística de análise de *mixins*, para as quais é possível garantir que os resultados estão corretos.

Algoritmo 2 Procedimento de análise temporal

```
1: procedure ANALISE-TEMPORAL(entradas)
2:     for each entrada ∈ entradas do                ▷ Para cada uma das entradas
3:         chaves = []
4:         for each chave ∈ entrada do
5:             chaves.add(chave)
6:         end for
7:         chaveReal = chaveMaisRecente(chaves)
8:         for each chave ∈ entrada do
9:             if chave != chaveReal then
10:                chave += '*'          ▷ Marca as chaves identificadas como mixins
11:            end if
12:        end for
13:    end for
14:    return entradas                           ▷ Retorna a lista de chaves marcadas
15: end procedure
```

Desta forma é possível estimar a eficácia da análise temporal sobre as transações para as quais não se sabe a chave real.

Do conjunto de 4.097.846 entradas deduzidas utilizando a estratégia da análise de *mixins*, 3.727.410, ou seja, aproximadamente 90,96% foram corretamente deduzidas com a utilização da análise temporal das chaves. Isto significa que apenas em 9,04% dos casos a chave real não é a mais recente dentre as chaves as chaves de entrada.

Não é possível, porém, assumir que 90,96% das transações que resistiram à análise de *mixins* podem ser deduzidas pela análise temporal, pois novas políticas de escolha de *mixins* foram implementadas em 16 de dezembro de 2016, garantindo a inclusão de *mixins* recentes nas entradas, conforme discutido na subseção 2.2.4. Com a finalidade de estimar o máximo de transações que já foram passíveis de serem rastreadas, os ataques foram aplicados aos dados do *blockchain* que precedem a atualização na política de seleção de *mixins*, contidos nos blocos 0 ao 1200078.

Tabela 7 – Resultados dos ataques e estimativa de entradas rastreáveis.

Observável	**Quantidade**
Número total de entradas	17.374.129 (100%)
Entradas que contêm 0 mixins	12.130.656 (69,82%)
Entradas vulneráveis à dedução	3.481.943 (20,04%)
Entradas rastreadas	15.612.599 (89,86%)
Entradas não-rastreáveis pelo Algoritmo 1	1.761.530 (10,13%)
Taxa de acertos da análise temporal	92,48%
Total de entradas rastreáveis (estimado)	17.195.509 (98,97%)

A Tabela 7 apresenta os resultados da execução dos ataques nos dados do *blockchain* anteriores a atualização da política de seleção de *mixins*. Como pode ser observado, a análise temporal obteve uma taxa de acertos de 92,48% sobre as entradas para as quais as chaves reais são conhecidas. Isto permite assumir que das 1.761.530 entradas restantes, 92,48%, ou seja, 1.629.053 entradas, podem ser corretamente deduzidas pela análise temporal. Isto leva a uma taxa de rastreio estimada em 98,97% de todas as entradas presentes no *blockchain*, um resultado preocupante dado o foco da criptomoeda Monero em proteger as informações dos usuários.

Em resumo, os resultados apresentados evidenciam a necessidade de investigar falhas nas estratégias de privacidade empregadas no sistema Monero. A resolução dos problemas de segurança contribui para a criação de criptomoedas mais seguras, que atendam às necessidades de privacidade dos seus usuários.

6 O ATAQUE DE MANIPULAÇÃO DE *MIXINS*

Considere a existência de uma entrada de transação tx do sistema Monero que contém quatro chaves de entrada (in), uma chave contendo fundos utilizados para a realização do pagamento e três chaves *mixins* que ajudam a ocultar a chave real. A entrada da transação é representada pela seguinte expressão:

$$tx.in = \{pk_1, pk_2, pk_3, pk_4\}$$

Suponha que três das chaves públicas pk_x da entrada (e.g., pk_1, pk_2 e pk_3) pertencem ao conjunto de chaves que um atacante obteve como saídas de transações passadas. O conjunto de chaves de entrada é composto por quatro chaves, das quais apenas uma chave, neste exemplo a pk_4, não pertence ao atacante. Neste caso, ao remover as suas chaves do conjunto de entrada, o atacante consegue descobrir qual é a chave real utilizada na transação para efetuar um pagamento.

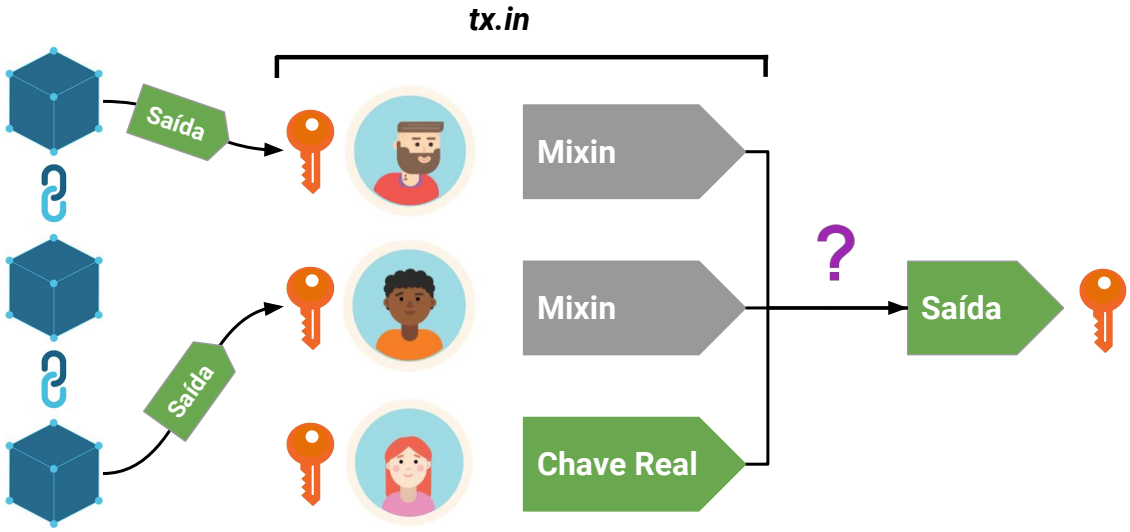
O atacante pode remover uma chave pk_x da entrada de uma transação realizada por outro usuário em duas situações. Na primeira situação, a chave faz parte do conjunto de saídas de transações recebidas pelo próprio atacante. Neste caso, se o atacante não utilizou a chave para efetuar um pagamento em uma transação, ela está sendo utilizada como *mixin*. No segundo caso, a chave já foi utilizada para efetuar um pagamento em uma transação anterior, como é o caso da chave pk_4 do exemplo anterior. É importante ressaltar que cada chave pública só pode ser utilizada para pagamento uma única vez.

Com base no exemplo apresentado, este capítulo introduz e discute um ataque de manipulação de *mixins*. Este ataque explora o mecanismo de privacidade das assinaturas em anel, que esconde as chaves reais de usuários que efetuam transações em um conjunto de chaves de entrada retiradas do *blockchain*. A Figura 13 ilustra o processo de criação de uma assinatura em anel.

No momento de criação de uma transação, o usuário deve escolher um número de chaves *mixins* para ocultar a chave real do pagamento. O sistema seleciona o número de chaves escolhido pelo usuário a partir das chaves públicas de saídas de transações presentes no *blockchain* e adiciona-as na entrada da transação, junto com a chave real, como ilustrado na Figura 13. Se mais de uma chave real for utilizada, cada chave possui o seu próprio conjunto de *mixins*. Após a escolha das chaves, o sistema cria uma assinatura criptográfica que permite que o receptor receba os fundos da transação sem saber a partir de qual chave o pagamento foi originado.

O objetivo do ataque é permitir que um usuário malicioso seja capaz de rastrear as chaves sendo utilizadas por usuários nas entradas de transações. Para isto, o atacante precisa conhecer todas as *mixins* presentes nas entradas que deseja rastrear. Ao remover todas as chaves utilizadas como *mixins*, o atacante irá descobrir a chave real do pagamento realizado naquela transação.

Figura 13 – Processo de criação de uma assinatura em anel.



A dificuldade do ataque está relacionada à necessidade de conhecer as chaves extras, ou *mixins*, utilizadas por usuários durante a criação das transações. O sistema Monero seleciona as chaves extras a partir das chaves públicas geradas como saídas de transações presentes no *blockchain*. O número de chaves de saída geradas como resultado de uma transação é variável e depende do número de endereços que receberão um pagamento. Cada um dos endereços receptores de fundos de uma transação receberá uma chave de saída contendo um valor em criptomoedas e esta chave poderá ser posteriormente selecionada pelo sistema para ser incluída como *mixin* em uma entrada de transação. Para fazer com que suas próprias chaves sejam selecionadas como *mixins*, sem comprometer serviços ou *softwares*, o atacante deve possuir chaves de saída de transações e aguardar até que uma nova transação selecione um conjunto de *mixins* composto somente pelas suas chaves. Infelizmente, para o atacante, existem milhões de chaves disponíveis no *blockchain* e as chances de que somente suas chaves sejam escolhidas para formar o conjunto de *mixins* de uma entrada de transação são pequenas.

De acordo com estudos sobre rastreabilidade de transações e reações em cadeia, conduzidos por desenvolvedores da criptomoeda Monero em 2014, uma das melhores estratégias sob a ótica do atacante é inundar a rede da criptomoeda o mais rápido possível com as suas próprias transações (NOETHER; MACKENZIE, 2014). Porém, a criação de transações possui um custo associado, a chamada taxa de transação, necessária para incentivar os mineradores a escolherem transações para validar. Quanto maior a taxa paga por uma transação, mais rápido ela será selecionada por mineradores. A taxa paga pela confirmação de uma transação é proporcional ao seu tamanho em *kilobytes*, que depende do número de entradas, *mixins* e saídas. A taxa por transação também depende da remuneração atual do sistema por bloco minerado e de alguns valores de referência,

definidos pelos desenvolvedores. O valor cobrado em XMR por cada *kilobyte* da transação é definido pela Equação 6.1.

$$Taxa\ por\ kB = (R/R_0) \times (M_0/M) \times F_0 \times (60/300) \times 4 \tag{6.1}$$

onde:

R = Valor base da recompensa em XMR por cada bloco minerado

R_0 = Valor base de referência (10 XMR)

M = Limite do tamanho de um bloco para que um minerador não tenha sua recompensa diminuída por tamanho excessivo do bloco.

M_0 = Tamanho mínimo de um bloco (300 kB)

F_0 = 0,002 XMR

4 = Fator de ajuste para multiplicação da taxa e aceleração da confirmação de transações (taxas padrão usam o valor 4).

Devido à existência desta taxa, que chegou a custar em média 20 dólares por transação em dezembro de 2017[1], adicionar ao *blockchain* do sistema uma grande quantidade de chaves originadas pelas suas próprias transações seria extremamente custoso para o atacante. Levando isto em consideração, os desenvolvedores da criptomoeda Monero negligenciaram a possibilidade da ocorrência deste tipo de ataque.

No dia 18 de outubro de 2018, foi incorporado ao sistema Monero um novo protocolo chamado *Bulletproof*. Este protocolo substituiu o antigo protocolo RingCT na tarefa de fornecer provas verificáveis da efetuação de transações sem revelar quaisquer dados sobre elas. O protocolo *Bulletproof* oferece a vantagem de gerar provas criptográficas com um tamanho reduzido, até 97% menores do que as do antigo protocolo. Um tamanho de transação reduzido implica diretamente na redução da taxa necessária à criação de transações. Após esta atualização, a execução de um ataque de manipulação de *mixins* tornou-se viável, como é demonstrado a seguir.

O ataque proposto explora o protocolo *Bulletproof* para permitir que o atacante gere uma quantidade significativa de chaves públicas de saída de transações e ocupe uma grande parte do *blockchain* com suas próprias chaves. Ao utilizar esta estratégia, as chances de que as chaves do atacante sejam selecionadas como *mixins* são maiores.

6.1 Modelo de atacante

O ataque proposto considera um atacante que possui acesso aos dados do *blockchain* da criptomoeda Monero e possui fundos suficientes para executar a quantidade de transações que forem necessárias para que ele seja capaz de rastrear transações de outros usuários. Uma análise dos custos de execução do ataque é apresentada na seção 6.4. O

[1] <https://bitinfocharts.com/comparison/monero-transactionfees.html>

atacante também possui quinze endereços distintos pertencentes a carteiras do sistema Monero e um destes contém o valor em XMR necessário para cobrir os gastos de criação das transações para a execução do ataque.

Foram escolhidos quinze endereços para a execução do ataque, pois esta é uma quantidade de chaves de saída frequente em transações. Transações com um número de chaves de saída menor do que oito são mais frequentes, no entanto, conforme é discutido na seção 6.2, criar transações com um número pequeno de saídas aumenta os custos do ataque. Quantias de chaves de saída maiores são menos frequentes, podendo fazer com que as transações do atacante se destaquem dentre as outras presentes no *blockchain*.

Por fim, o atacante possui a liberdade de criar tantas transações quantas desejar a qualquer momento desde que pague as taxas correspondentes. A confirmação das transações depende do trabalho dos mineradores do sistema e nem sempre elas serão selecionadas para serem validadas tão logo sejam criadas.

6.2 Estratégia de ataque

Para que seja capaz de rastrear entradas de transações, o atacante deve criar um grande número de chaves públicas de saída durante o intervalo de tempo no qual deseja executar o ataque. Somente as transações criadas após o início do processo de geração de chaves do atacante estarão sujeitas a serem rastreadas. Quanto maior o período durante o qual o atacante permanecer criando novas saídas de transação, maior será a eficácia do ataque, visto que o sistema seleciona 50% das chaves usadas como *mixins* a partir das saídas de transações geradas nos últimos 1,8 dias. Os outros 50% são selecionados a partir de saídas de transações presentes no *blockchain* que utilizam o protocolo RingCT de transações confidenciais.

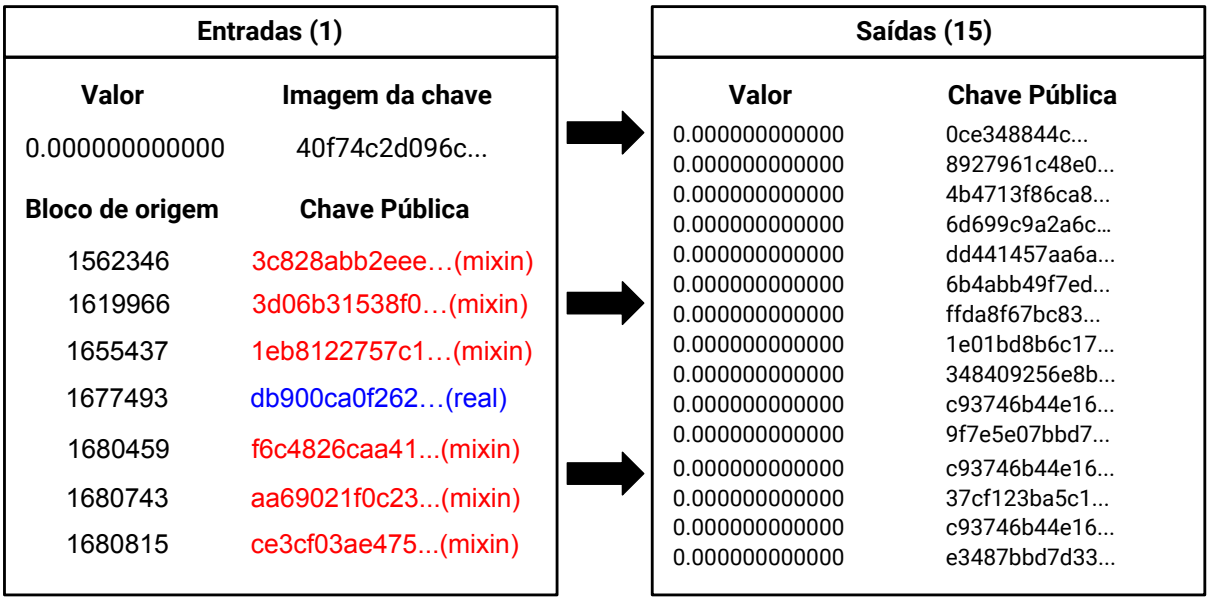
De modo a criar chaves de saída, o atacante deverá enviar pagamentos para seus outros endereços a partir do endereço de carteira que contém a maioria dos seus fundos em XMR. Estas transações resultam em novas chaves públicas que estarão em sua posse. O objetivo é gastar o menor valor possível com cada transação, portanto, a melhor alternativa é enviar o menor valor permitido em cada transação, 1 piconero, que equivale à 10^{-12} XMR, e pagar o valor da taxa associada. Ao criar chaves com valores pequenos, o atacante não precisa preocupar-se em utilizá-las posteriormente.

Uma parte significativa do custo de uma transação envolve os custos das garantias criptográficas de privacidade, fazendo com que o envio de fundos para vários endereços em uma transação com várias saídas seja menos custoso do que criar uma transação com apenas uma saída para cada pagamento. Por este motivo, o atacante beneficia-se da utilização de transações com um grande número de saídas para a execução do ataque. Vale ressaltar que é importante escolher um número de saídas que apareça com frequência em transações no *blockchain*. Isto ajuda a evitar que as transações do atacante se destaquem e levantam suspeitas. Para fins de análise, este trabalho considera quinze

saídas de transação como o número padrão utilizado pelo atacante, quatorze endereços que receberão chaves de 1 piconero e o próprio endereço que originou a transação, que receberá o troco da transação.

Outro fator que influencia no preço da transação é o número de *mixins*. Para diminuir ainda mais o custo das transações, deve ser utilizado com a chave do pagamento o menor número de chaves extras permitido pelo sistema, que é atualmente seis. A Figura 14 ilustra a estrutura de uma transação que pode ser utilizada para a execução do ataque. A única entrada da transação vem do endereço do atacante com o maior número de fundos e utiliza a menor quantidade de *mixins* permitida, seis. Os quinze endereços de saída são endereços das outras carteiras do atacante e um endereço da própria carteira de onde o pagamento foi originado, que receberá o troco da transação. Os valores das chaves são ocultados devido aos mecanismos que fornecem transações confidenciais.

Figura 14 – Exemplo da estrutura de uma transação para a execução do ataque.



Após efetuar os pagamentos, o atacante deve reunir uma lista com todas as saídas de transações que ele agora possui em todos os seus endereços. Para ser capaz de rastrear as chaves de outras transações, é necessária a obtenção de uma cópia dos dados do *blockchain* da criptomoeda Monero. O processo de rastreio consiste na checagem de todas as chaves presentes em uma entrada e a identificação daquelas que estão contidas no conjunto de chaves pertencentes ao atacante. O pseudocódigo apresentado no Algoritmo 3 descreve o procedimento de rastreio de chaves.

O procedimento é iniciado com o recebimento dos dados dos blocos extraídos do *blockchain* e o conjunto de chaves pertencentes ao atacante (Linha 1). Um laço de repetição determina que o código do rastreamento deve analisar as entradas novamente quando

Algoritmo 3 Procedimento de rastreio de chaves

```
1: procedure RASTREAR_ENTRADAS(blocos, chavesAtacante)
2:     chavesRastreadas ← {}
3:     while true do
4:         numChavesConhecidas ← |chavesAtacante|
5:         for each bloco ∈ blocos do                    ▷ Para cada bloco extraído
6:             transacoes ← extraiTransacoes(bloco)
7:             for each transacao ∈ transacoes do
8:                 entradas ← extraiEntradas(transacao)
9:                 for each entrada ∈ entradas do
10:                    chaves ← extraiChaves(entrada)
11:                    mixinsUsadas ← |chaves| − 1
12:                    mixinsRemovidas ← 0
13:                    for each chave ∈ chaves do
14:                        if chave ∈ chavesAtacante then
15:                            mixinsRemovidas ← mixinsRemovidas + 1
16:                        end if
17:                    end for
18:                    if mixinsRemovidas == mixinsUsadas then
19:                        chaveReal ← chaves − (chavesAtacante ∩ chaves)
20:                        chavesAtacante ← chavesAtacante ∪ chaveReal
21:                        chavesRastreadas ← chavesRastreadas ∪ chaveReal
22:                    end if
23:                end for
24:            end for
25:        end for
26:        if numChavesConhecidas == |chavesAtacante| then
27:            break
28:        end if
29:    end while
30:    return chavesRastreadas
31: end procedure
```

novas chaves forem rastreadas durante a iteração. Para isso, o tamanho do conjunto de chaves conhecidas é armazenado no início da iteração e verificado ao final da mesma (Linhas 4 e 26). Quando uma nova chave é rastreada, o algoritmo deve iterar sobre as transações novamente, pois a sua eliminação dos conjuntos de entrada pode permitir o rastreio de novas chaves. Para analisar as chaves, é realizada a extração de todas entradas de transações presentes no conjunto de dados (Linhas 5-25). O algoritmo verifica se as chaves estão contidas no conjunto do atacante. Se estiverem, a variável *mixinsRemovidas* é incrementada (Linha 15). Ao final da iteração sobre as chaves da entrada, se restar apenas uma chave não conhecida, esta é a chave real da transação, visto que as outras pertencem ao atacante e nenhuma delas foi utilizada. A nova chave rastreada é adicionada ao conjunto de chaves conhecidas pelo atacante e armazenada no conjunto *chavesRastreadas* (Linhas 20 e 21). As novas chaves podem ser eliminadas das entradas de transações,

permitindo o rastreio de outras entradas. Esse efeito se propaga, aumentando a eficácia do ataque. Ao final da iteração, se novas chaves foram rastreadas e adicionadas ao conjunto do atacante, o algoritmo continua, senão, o laço é interrompido (Linhas 26 e 27). Quando o laço é interrompido, as chaves rastreadas são retornadas (Linha 30).

6.3 Simulações e resultados

Com o objetivo de avaliar a capacidade de rastreio de chaves através da execução do ataque proposto, foram realizadas simulações de ataques. Para a execução das simulações foram extraídos conjuntos de dados do *blockchain* da criptomoeda Monero. Para simular a posse de uma porcentagem das chaves por parte do atacante, chaves de saída presentes nos dados do *blockchain* foram selecionadas de maneira pseudo aleatória. Para simular a execução do ataque, foi executado o Algoritmo 3 sobre as entradas de transações presentes nos conjuntos de dados. Os seguintes cenários foram utilizados nas simulações:

- Período de 3 meses, blocos 1.433.039 à 1.499.600.
- Período de 6 meses, blocos 1.366.664 à 1.499.600.
- Período de 9 meses, blocos 1.300.239 à 1.499.600.
- Período de 1 ano, blocos 1.236.197 à 1.499.600.

Os intervalos de blocos foram selecionados de maneira a abranger os dados mais recentes possíveis. Por esta razão, os cenários com períodos de tempo menores iniciam em blocos de número maior, que são mais recentes. A Tabela 8 resume os resultados das simulações nos quatro cenários avaliados. As quantidades de chaves do atacante correspondem apenas às chaves públicas de saídas de transação geradas durante o período avaliado.

Tabela 8 – Número de entradas de transações rastreadas de acordo com o número de chaves do atacante.

Chaves do atacante	**Entradas rastreáveis**			
	3 meses	**6 meses**	**9 meses**	**1 ano**
1%	21.741(1,66%)	42.325(1,67%)	47.338(1,25%)	57.958(1,36%)
25%	24.811(1,89%)	90.919(3,59%)	323.290(8,48%)	3.658.855(85,59%)
50%	67.977(5,16%)	511.148(20,35%)	3.208.179(84,83%)	3.814.831(89,22%)
75%	249.968(19,01%)	1.968.541(77,84%)	3.250.988(85,96%)	3.836.312(89,72%)
100%	534.778(40,73%)	2.143.723(84,84%)	3.263.091(86,28%)	3.839.963(89,81%)

Conforme o número de saídas de transações em posse do atacante aumenta, maiores são as chances de que uma entrada escolha somente as suas chaves como *mixins*. O

período de tempo durante o qual o atacante permanece criando novas saídas de transação é importante porque uma entrada de transação seleciona *mixins* a partir de chaves já presentes no *blockchain*, incluindo aquelas geradas há meses, desde que tenham sido criadas após o lançamento do protocolo RingCT de transações confidenciais. Quanto maior a cobertura das chaves do atacante no *blockchain*, melhor será a sua capacidade de rastrear transações. Quando o atacante possui 1% de todas as saídas de transação geradas, sua capacidade de rastreamento é de 1,66%, 1,67%, 1,25% e 1,36% das entradas geradas nos cenários de 3, 6, 9 e 12 meses, respectivamente. A oscilação observada nos valores deve-se a seleção de chaves durante a simulação, que pode favorecer ligeiramente os resultados do ataque caso sejam selecionadas chaves que aparecem com mais frequência nas entradas. Com o aumento do número de chaves em posse do atacante, seu poder de rastreio é elevado.

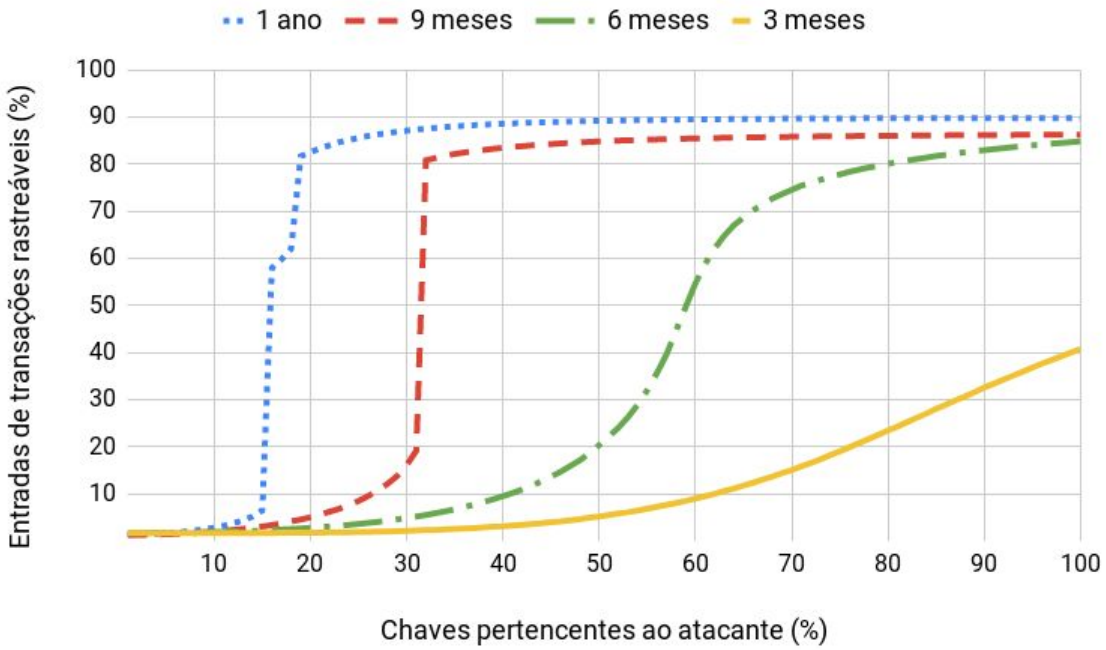
O cenário composto por três meses de dados apresenta as menores taxas de rastreio devido ao seu tamanho relativamente curto, implicando na escolha de *mixins* originadas antes do início do ataque, sobre as quais o atacante não tem controle. Conforme a Tabela 8, quando o atacante possui 50% das chaves, é capaz de rastrear 5,16% das entradas de transações no intervalo de 3 meses. Com 75% e 100% das chaves, 19,01% e 40,73% das entradas podem ser rastreadas. No cenário de seis meses, com de 50% das chaves o atacante adquire uma capacidade de rastreio de 20,35% e com 75% das chaves é capaz de rastrear as chaves reais de 77,84% das entradas utilizadas nesse período. Na simulação onde o atacante possui 100% das chaves de saída de transação geradas no período, sua capacidade de rastreio é de 84,84% das entradas de transações. Em um cenário real, não é possível que o atacante controle 100% das chaves geradas em um período de vários meses, a menos que fosse o único participante da rede. Os cenários onde o atacante possui 100% das chaves tem como objetivo mostrar que mesmo tendo controle sobre todas as chaves criadas em um período, o atacante não é capaz de rastrear todas as entradas de transações.

No cenário de 9 meses, com posse de 50% das chaves o atacante é capaz de rastrear 84,83% chaves reais das entradas. No cenário de 1 ano, ao controlar 25% das chaves o atacante consegue identificar 85,59% das chaves de pagamentos. A partir destes pontos nos dois cenários, o poder de rastreamento do atacante aproxima-se do máximo possível e deixa de crescer exponencialmente. O poder do atacante se aproxima do máximo observado antes do seu controle sobre as chaves estar próximo de 100%. A causa deste crescimento exponencial do número de entradas rastreadas é a ocorrência de reações em cadeia. A Figura 15 ilustra o aumento na capacidade de rastreio conforme o número de chaves em posse do atacante aumenta. No período de um ano, a partir do momento no qual o atacante possui 16% das chaves geradas a capacidade de rastreio aumenta exponencialmente. No período de nove meses, o mesmo ocorre quando o atacante possui mais do que 31% das chaves. Este efeito é causado por uma reação de massa crítica. Pesqui-

sadores do laboratório de pesquisas do sistema Monero descreveram este efeito em um trabalho sobre a rastreabilidade de transações e reações em cadeia no protocolo CryptoNote (NOETHER; MACKENZIE, 2014). Um cenário de massa crítica ocorre quando o atacante possui chaves suficientes para que seja capaz de rastrear transações, utilizar as chaves recém rastreadas para rastrear novas transações e assim por diante, gerando uma reação em cadeia que afeta um número expressivo de entradas.

Para fins de comparação, a Figura 16 mostra o aumento na capacidade de rastreio de chaves quando são levadas em consideração apenas as chaves que estão em posse do atacante no início da execução do ataque. Como pode ser observado, o número de entradas de transações rastreáveis cresce de maneira proporcional em relação ao número de chaves do atacante, diferentemente do que ocorre quando acontecem reações em cadeia.

Figura 15 – Gráfico ilustrando o aumento na capacidade de rastreio em cada cenário avaliado.



Algumas transações não puderam ser rastreadas, mas tiveram o tamanho efetivo do seu conjunto de *mixins* reduzido. Isto torna as entradas potencialmente mais vulneráveis à ataques de rastreamento. A Tabela 9 apresenta as quantidades de *mixins* removidas das entradas no cenário de um ano, mostrando o impacto do ataque incluindo as entradas de transações que não puderam ser rastreadas.

Os números de *mixins* removidas das entradas que aparecem com mais frequência na tabela são 2 e 4. Isto ocorre porque durante um longo período de tempo o tamanho mínimo do conjunto de *mixins* permitido pelo sistema foi 2. Posteriormente, o tamanho mínimo foi dobrado, levando ao aumento de transações que contêm 4 *mixins*. Das entradas

Figura 16 – Gráfico ilustrando o aumento na capacidade de rastreio em cada cenário avaliado, utilizando somente as chaves iniciais do atacante.

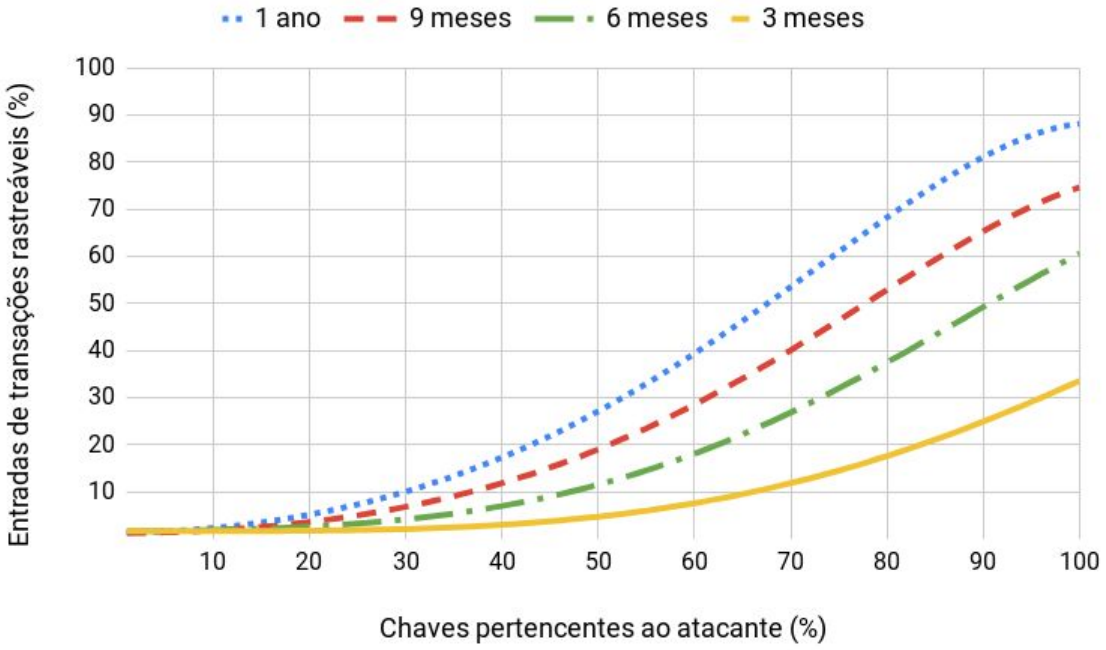


Tabela 9 – Número de *mixins* eliminadas de entradas de transações de acordo com o número de chaves pertencentes ao atacante em um período de 1 ano.

Mixins eliminadas	**Chaves pertencentes ao atacante**				
	1%	**25%**	**50%**	**75%**	**100%**
0	4.110.887	473.478	450.108	427.797	408.634
1	160.613	26.219	41.435	60.154	75.098
2	3.891	1.539.258	1.513.384	1.515.641	1.519.655
3	131	248.170	153.538	139.022	136.791
4	11	1.715.222	1.825.180	1.838.589	1.840.643
5	8	128.514	142.672	144.556	144.835
6	9	51.167	51.828	52.132	52.192
7	7	10.158	8.255	7.592	7.481
8	3	12.050	16.215	16.862	16.967
9	6	7.799	8.373	8.443	8.461
10	4	29.554	30.247	30.381	30.405
Outros	24	34.005	34.359	34.425	34.432

de transações apresentadas na Tabela 9, aquelas que tiveram 0 *mixins* eliminadas não foram afetadas pelo ataque de manipulação de *mixins* ou são compostas apenas pela chave do pagamento. Transações sem chaves extras ocorreram porque antes do lançamento do protocolo RingCT de transações confidenciais, chaves usadas como *mixins* deveriam possuir o mesmo valor da chave do pagamento. Se no momento da criação da transação não houvessem chaves com o mesmo valor para servirem de *mixins*, a transação era criada somente com a chave do pagamento. Na Tabela 9, as entradas que não tiveram *mixins* removidas representam 96,14%, 11,07%, 10,52%, 10%, e 9,55% das entradas quando o atacante possui 1%, 25%, 50%, 75% e 100% das chaves, respectivamente.

6.4 Análise de custos

A execução do ataque de manipulação de *mixins* requer que o atacante efetue uma grande quantidade de transações para criar chaves públicas de saída de transações. O atacante precisa pagar uma taxa a cada transação criada, para que os mineradores do sistema adicionem-as no *blockchain*. Quanto maior o poder de rastreio desejado pelo atacante, mais transações devem ser criadas e, consequentemente, o custo do ataque é maior. Esta seção apresenta uma análise do custo da execução do ataque de manipulação de *mixins* e apresenta uma alternativa para viabilização do ataque através de ataques de *in-browser crytojacking*.

Conforme foi observado no cenário de simulação contendo 1 ano de dados do *blockchain*, composto pelo intervalo de blocos 1.236.197 à 1.499.600, entre os meses de fevereiro de 2017 e fevereiro de 2018, o número médio de saídas de transações criadas por dia foi 11.713. Este número é utilizado como base para calcular a quantidade de chaves necessárias para que o atacante possua uma porcentagem de todas as chaves de saída de transações no *blockchain* durante o ataque. Se o atacante objetiva possuir 50% das chaves, deve criar por dia o mesmo número de saídas de transações do que todos os outros usuários juntos, ou seja, 11.713.

Para estimar o valor da taxa para cada transação criada, é considerada uma transação que contém quinze endereços de saída e um endereço de entrada com seis *mixins*, conforme discutido na seção 6.2. Através da extração da média do valor da taxa cobrada por algumas transações com o mesmo número de entradas e saídas, o custo de cada transação do atacante é definido como 0,00019 XMR ou 0,0118548 USD. Para o cálculo do custo do ataque foi utilizada a cotação do dia 23 de novembro de 2018, quando 1 XMR equivalia à 69,42 USD. A Tabela 10 mostra o número de chaves que o atacante precisa criar para controlar diferentes porcentagens de todas chaves.

Todos os números de chaves necessários foram estimados considerando a média diária de chaves criadas por outros usuários do sistema. A criação da quantidade de chaves necessárias deve ser distribuída ao longo do tempo, de forma que as chaves do atacante estejam presentes em blocos ao longo de todo o período de ataque. Isto é

Tabela 10 – Número de chaves necessárias para que o atacante tenha controle sobre uma porcentagem das chaves do *blockchain* durante o período do ataque.

Controle do atacante	**Chaves de saídas de transações necessárias**			
	3 meses	**6 meses**	**9 meses**	**1 ano**
1%	10.620	21.240	31.860	43.070
25%	351.360	702.720	1.054.080	1.424.960
50%	1.054.170	2.108.340	3.162.510	4.275.245
75%	3.162.510	6.325.020	9.487.530	12.825.735
99%	104.362.830	208.725.660	313.088.490	423.249.255

importante visto que 50% das *mixins* são selecionadas dos dados gerados nos últimos 1,8 dias.

Tabela 11 – Custos de criação das transações necessárias para controlar porcentagens das chaves do *blockchain*.

Controle do atacante	**Custos em taxas de transações (XMR/USD)**			
	3 meses	**6 meses**	**9 meses**	**1 ano**
1%	0,135 XMR 9,36 USD	0,269 XMR 18,67 USD	0,403 XMR 27,98 USD	0,545 XMR 37,84 USD
25%	4,450 XMR 308,65 USD	8,901 XMR 617,37 USD	13,351 XMR 926,02 USD	18,049 XMR 1.252,05 USD
50%	13,352 XMR 926,22 USD	26,705 XMR 1.852,52 USD	40,058 XMR 2.775,61 USD	54,153 XMR 3.752,26 USD
75%	40,058 XMR 2.775,61 USD	80,116 XMR 5.551,23 USD	120,175 XMR 8.326,92 USD	162,459 XMR 11.258,40 USD
99%	1.321,929 XMR 91.596,46 USD	2.643,858 XMR 183.192,92 USD	3.965,787 XMR 274.789,38 USD	5.361,157 XMR 371.474,56 USD

A Tabela 11 mostra uma estimativa dos custos de criação de transações para a execução do ataque. Um ataque que objetiva controlar 50% das chaves públicas de saída geradas custa ao atacante 3.752,26 dólares por ano. Com o controle de 50% das chaves o atacante é capaz de rastrear 84,83% das entradas de transações 9 meses após o início do ataque e 89,22% das entradas após 1 ano. Este é um valor baixo quando levado em consideração o poder aquisitivo de entidades que podem beneficiar-se da quebra de privacidade da criptomoeda Monero, como grandes empresas do setor financeiro, governos interessados em coleta de dados pessoais e investidores de outras criptomoedas. A pos-

sibilidade da execução de um ataque capaz de impactar na privacidade de até 89,81% das entradas de transações criadas em um período de 1 ano, com um custo tão baixo, representa um grande risco para a segurança da criptomoeda Monero. Os valores são baseados no número de chaves apresentados na Tabela 10. Entretanto, vale enfatizar que existe a possibilidade de reduzir os custos do ataque para zero. O atacante pode utilizar ataques de *in-browser cryptojacking* para a obtenção de fundos, como descrito na seção a seguir.

6.5 Obtenção de fundos para a execução do ataque

Como forma de obter os fundos em XMR necessários para o pagamento de taxas de transação, o atacante pode utilizar uma estratégia de obtenção de fundos passiva, baseada em ataques de *in-browser cryptojacking*. Ao realizar a injeção de códigos de mineração em sites de terceiros ou em suas próprias páginas da *web*, conforme discutido no Capítulo 2, subseção 2.2.5, o atacante lucra com o poder de processamento dos usuários e pode utilizar as criptomoedas mineradas para pagar as taxas de transações do ataque. Por meio desta estratégia é possível obter o valor necessário para executar o ataque.

A quantidade de fundos obtida pelo atacante depende do número de usuários que visitam a página que contém o código de mineração e do tempo durante o qual permanecem nela. Se o número de visitantes nas páginas infectadas exceder a quantidade necessária para financiar o ataque, além de obter fundos para pagar as taxas de transações do ataque, o atacante recebe lucro em criptmoedas. Para evitar a detecção do código de mineração o atacante deve utilizar técnicas de evasão de detecção como servidores de retransmissão, ofuscação de código, ocultação de *payload* e controle do uso do processador. Estas técnicas são discutidas em detalhes no Capítulo 2, subseção 2.2.5.

O poder de médio de processamento de *hashes* do algoritmo CryptoNight de cada usuário é considerado como 40,5 *hashes*/s (KONOTH et al., 2018). Para estimar o número de mineradores necessários, é considerado o número de usuários que precisam participar do processo de mineração 24 horas por dia durante o período do ataque. A Tabela 12 apresenta a quantidade de usuários que devem visitar as páginas infectadas para gerar renda suficiente para a execução do ataque.

A quantidade de usuários ativos necessários é a mesma para os diferentes cenários de ataque. A página com o código de mineração deve continuar ativa durante todo o período de execução do ataque. Para que o atacante seja capaz de controlar 1% das chaves são necessários 7 mineradores trabalhando 24 horas por dia durante o intervalo de ataque. Para controlar 25%, 50% e 75% das chaves são necessários 224, 671 e 2.012 mineradores respectivamente. Para controlar 99% das chaves, o número de visitantes nas páginas infectadas cresce exponencialmente, atingindo um número de usuários necessários de 66.375. Quanto maior o tempo durante o qual os usuários permanecerem nas páginas, maior será o lucro do atacante. Por esta razão, páginas que oferecem conteúdos como

Tabela 12 – Número de usuários ativos 24 horas por dia necessários para financiar o ataque.

Controle do atacante	**Mineradores necessários (24h/dia)**
1%	7
25%	224
50%	671
75%	2.012
99%	66.375

jogos e *streaming* de vídeo são mais apropriadas como alvos de códigos de mineração. Entretanto, dificilmente os usuários permanecerão por várias horas na mesma página da *web*. Uma página de *streaming* de vídeo popular, como `www.youtube.com`, recebe por mês uma média de 26,22 bilhões de visitas onde a duração média de cada uma é de apenas 20 minutos e 5 segundos (SAAD; KHORMALI; MOHAISEN, 2018). Portanto, apesar de serem necessários 2.012 usuários trabalhando 24 horas por dia para que o atacante controle 75% das transações conforme a Tabela 12, o número de usuários distintos necessários por dia é muito maior. Supondo que a duração média por sessão de usuário na página infectada seja de 5 minutos, para que o atacante controle 50% das transações são necessárias 193.248 visitas de usuários por dia.

O atacante pode incluir códigos de mineração em várias páginas da *web* distintas para aumentar seus lucros. Além disso, infectar páginas com públicos-alvos de diferentes fusos horários diferentes também contribui para que o ataque continue em execução em horários nos quais o tráfego de rede diminui em regiões específicas.

Através da execução do ataque de mineração via navegadores, o custo do ataque de manipulação de *mixins* ao atacante pode ser reduzido à zero. Desta forma, é possível controlar chaves públicas de saídas de transações do sistema Monero e rastrear entradas de transações mesmo que o atacante não possua dinheiro para a execução do ataque.

7 DESAFIOS DE PESQUISA

A tecnologia utilizada pelas criptomoedas é bastante recente. Mais recentes ainda são os sistemas cujo objetivo principal é oferecer privacidade. Como pode-se observar no Capítulo 3, existem apenas alguns estudos e ataques recentes à moedas digitais como a Monero.

Durante o desenvolvimento deste trabalho, foram identificados aspectos que podem ser explorados de maneira a consolidar sistemas que tem por objetivo manter a privacidade dos usuários. A seguir são apresentados alguns caminhos de pesquisa relacionadas à segurança e privacidade em criptomoedas baseadas em protocolos como o *CryptoNote*.

Correlação de usuários com chaves do sistema. Apesar da existência de ataques que permitam rastrear as entradas reais de transações do protocolo *CryptoNote*, até agora nenhum foi capaz de relacionar dados de identificação dos usuários, como nomes, *nicknames* e endereços IP com os endereços usados em transações. Alguns dos possíveis vetores de ataque são a interceptação de tráfego de rede contendo dados de transações e *web scraping* para coleta de informações de usuários em páginas da *web*.

Investigação de chaves privadas de visualização. Usuários de moedas baseadas no protocolo *CryptoNote* utilizam um par de chaves pública e privada de visualização para visualizar as saídas de transações recebidas e usá-las em pagamentos. Serviços que necessitam de transparência divulgam suas chaves privadas de visualização para que usuários inspecionem suas transações e verifiquem que não ocorreram fraudes. Entretanto, o conhecimento das chaves privadas de um serviço fornece acesso ao seu histórico de transações e pode permitir a correlação desses dados com usuários ou seus hábitos.

Investigação de ataques pós-*RingCTs*. O sistema Monero lançou o protocolo *RingCT*, que oculta o valor de transações, aumentando a privacidade dos usuários do sistema. A partir da introdução do protocolo, o número mínimo de *mixins* do sistema também foi elevado, impedindo a realização de ataques previamente conhecidos. Os ataques existentes possuem pouco impacto sobre as transações que utilizam o protocolo *RingCT*. São necessários estudos sobre o impacto deste protocolo na segurança do sistema Monero e sobre vulnerabilidades que possam estar associadas à este novo tipo de transação.

8 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma análise da rastreabilidade das transações da criptomoeda Monero utilizando estratégias de ataque presentes na literatura e um novo ataque denominado ataque de manipulação de *mixins*. O ataque proposto consiste na criação de um grande número de chaves de saídas de transações, com o objetivo de controlar as chaves que são utilizadas para garantir privacidade às transações.

Resultados de simulações mostram que ao executar o ataque proposto, um atacante que controla 25% das chaves de saídas de transação geradas em um período de um ano é capaz de rastrear 85,59% das entradas de transações efetuadas neste mesmo período. Os resultados obtidos demonstram a existência de vulnerabilidades nos mecanismos que fornecem privacidade às transações da criptomoeda Monero, com ênfase no recém lançado protocolo *Bulletproof*, que viabilizou a execução do ataque de manipulação de *mixins*.

Uma análise de custos do ataque de manipulação de *mixins* também foi apresentada. O custo da criação de transações necessárias para a execução do ataque foi avaliado e os resultados da análise mostram que controlar 50% das chaves do sistema Monero custa 3.752,26 dólares por ano. Através da observação dos resultados, conclui-se que o custo do ataque é baixo dado o grande impacto exercido sobre as transações de uma criptomoeda focada em privacidade.

Por fim, com a intenção de reduzir o custo do ataque, foi apresentada uma estratégia de obtenção de fundos em XMR, utilizando ataques de mineração via navegadores. Páginas da *web* que possuam um público de 671 usuários ativos ou mais durante o dia inteiro, podem ser infectadas com códigos de mineração para permitir que o atacante tenha controle de metade das chaves do sistema sem gastar nada. As análises e o ataque apresentados enfatizam a importância de detectar falhas em mecanismos de segurança e privacidade em criptomoedas.

REFERÊNCIAS

ADKISSON, J. **Why Bitcoin is So Volatile**. 2018. Disponível em: <https://www.forbes.com/sites/jayadkisson/2018/02/09/why-bitcoin-is-so-volatile/#680d382239fb>. Citado na página 23.

APOSTOLAKI, M.; ZOHAR, A.; VANBEVER, L. Hijacking bitcoin: Routing attacks on cryptocurrencies. In: IEEE. **Security and Privacy (SP), 2017 IEEE Symposium on**. [S.l.], 2017. p. 375–392. Citado na página 43.

BIRYUKOV, A.; KHOVRATOVICH, D.; PUSTOGAROV, I. Deanonymisation of clients in bitcoin p2p network. In: ACM. **Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security**. [S.l.], 2014. p. 15–29. Citado 4 vezes nas páginas 15, 35, 43 e 46.

BLATTBERG, E. **New Jersey slaps MIT Bitcoin hackers with subpoena — and they're fighting back**. 2014. Disponível em: <https://venturebeat.com/2014/02/12/new-jersey-slaps-mit-bitcoin-hackers-with-subpoena-and-theyre-fighting-back/2/>. Citado na página 36.

CHAUM, D. Blind signatures for untraceable payments. In: SPRINGER. **Advances in cryptology**. [S.l.], 1983. p. 199–203. Citado na página 15.

COINTOPPER. **Difference between ASIC, GPU and CPU mining**. 2018. Disponível em: <https://cointopper.com/guides/difference-between-asic-gpu-and-cpu-mining>. Citado na página 35.

CRYPTONOTE. **CryptoNote Technology**. 2015. Disponível em: <https://cryptonote.org/inside/>. Citado 2 vezes nas páginas 34 e 36.

DUFFIELD, E.; DIAZ, D. **Dash: A PrivacyCentric CryptoCurrency**. [S.l.]: September, 2014. Citado 2 vezes nas páginas 15 e 32.

ESKANDARI, S. et al. A first look at browser-based cryptojacking. **arXiv preprint arXiv:1803.02887**, 2018. Citado 2 vezes nas páginas 36 e 38.

FLEDER, M.; KESTER, M. S.; PILLAI, S. Bitcoin transaction graph analysis. **arXiv preprint arXiv:1502.01657**, 2015. Citado 3 vezes nas páginas 15, 45 e 46.

GITLAB TEAM. **GitLab**. 2018. Disponível em: <https://https://about.gitlab.com/>. Citado na página 49.

HONG, G. et al. How you get shot in the back: A systematical study about cryptojacking in the real world. In: ACM. **Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security**. [S.l.], 2018. p. 1701–1713. Citado 2 vezes nas páginas 37 e 41.

IEEE. Special Report on Blockchain World. **IEEE Spectrum**, v. 10, oct 2017. <https://spectrum.ieee.org/static/special-report-blockchain-world>. Citado na página 22.

JEFFERYS, K. **The Problem With ASICs**. 2018. Disponível em: <https://medium.com/@LokiNetwork/the-problem-with-asics-3c9cee44f383>. Citado na página 35.

KONOTH, R. K. et al. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In: ACM. **Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security**. [S.l.], 2018. p. 1714–1730. Citado 3 vezes nas páginas 39, 41 e 73.

KREBS, B. **Who and What Is Coinhive?** 2018. Disponível em: <https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/>. Citado na página 37.

KSHETRI, N. 1 blockchain's roles in meeting key supply chain management objectives. **International Journal of Information Management**, Elsevier, v. 39, p. 80–89, 2018. Citado na página 22.

KUMAR, A. et al. A traceability analysis of monero's blockchain. In: SPRINGER. **European Symposium on Research in Computer Security**. [S.l.], 2017. p. 153–173. Citado 7 vezes nas páginas 15, 31, 43, 46, 53, 56 e 57.

MATHUR, N. **Cybersecurity: Cryptojacking attacks exploded by 8,5002017, says report**. 2018. Disponível em: <https://www.livemint.com/Technology/pDOm3QIWiGJk0I96EpzrHP/Cybersecurity-Cryptojacking-attacks-exploded-by-8500-in-2.html>. Citado na página 37.

MEDFAR87. **Cryptocurrency Growth & Adoption Statistics**. 2018. Disponível em: <https://steemit.com/cryptocurrency/@medfar87/cryptocurrency-growth-and-adoption-statistics>. Citado na página 23.

MEIKLEJOHN, S. et al. A fistful of bitcoins: characterizing payments among men with no names. In: ACM. **Proceedings of the 2013 conference on Internet measurement conference**. [S.l.], 2013. p. 127–140. Citado 4 vezes nas páginas 15, 28, 45 e 46.

MILLER, A. et al. An empirical analysis of traceability in the monero blockchain. **arXiv preprint arXiv:1704.04299**, 2017. Citado 6 vezes nas páginas 15, 43, 46, 53, 56 e 57.

MINERBLOCK. 2018. Disponível em: <https://github.com/xd4rker/MinerBlock>. Citado na página 39.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Citado 4 vezes nas páginas 15, 25, 28 e 31.

NOCOIN. 2018. Disponível em: <https://github.com/keraf/NoCoin>. Citado na página 39.

NOETHER, S.; MACKENZIE, A. A note on chain reactions in traceability in cryptonote 2.0. **Research Bulletin MRL-0001. Monero Research Lab**, v. 1, p. 1–8, 2014. Citado 2 vezes nas páginas 62 e 69.

PAGE, L. et al. **The PageRank citation ranking: Bringing order to the web.** [S.l.], 1999. Citado na página 45.

POPOV, S. The tangle. 2014. Citado na página 23.

RAUCHBERGER, J. et al. The other side of the coin: A framework for detecting and analyzing web-based cryptocurrency mining campaigns. In: ACM. **Proceedings of the 13th International Conference on Availability, Reliability and Security**. [S.l.], 2018. p. 18. Citado na página 37.

RON, D.; SHAMIR, A. Quantitative analysis of the full bitcoin transaction graph. In: SPRINGER. **International Conference on Financial Cryptography and Data Security**. [S.l.], 2013. p. 6–24. Citado 3 vezes nas páginas 43, 44 e 46.

RÜTH, J. et al. Digging into browser-based crypto mining. In: ACM. **Proceedings of the Internet Measurement Conference 2018**. [S.l.], 2018. p. 70–76. Citado na página 37.

SAAD, M.; KHORMALI, A.; MOHAISEN, A. End-to-end analysis of in-browser cryptojacking. **arXiv preprint arXiv:1809.02152**, 2018. Citado 3 vezes nas páginas 36, 40 e 74.

SABERHAGEN, N. V. **Cryptonote v 2. 0**. 2013. Citado 2 vezes nas páginas 24 e 32.

SASSON, E. B. et al. Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE. **Security and Privacy (SP), 2014 IEEE Symposium on**. [S.l.], 2014. p. 459–474. Citado na página 15.

SCHWARTZ, D. et al. The ripple protocol consensus algorithm. 2014. Citado na página 23.

SEGURA, J. **Malicious cryptomining and the blacklist conundrum**. 2018. Disponível em: <https://blog.malwarebytes.com/threat-analysis/2018/03/malicious-cryptomining-and-the-blacklist-conundrum/>. Citado na página 40.

SOFTWARE FREEDOM CONSERVANCY. **Git**. 2018. Disponível em: <https://git-scm.com/>. Citado na página 49.

The Monero Project. **About Monero**. 2018. Disponível em: <https://src.getmonero.org/resources/about/>. Citado na página 15.

TORPEY, K. **Study Suggests 25 Percent of Bitcoin Users Are Associated With Illegal Activity**. 2018. Disponível em: <https://bitcoinmagazine.com/articles/study-suggests-25-percent-bitcoin-users-are-associated-illegal-activity1/>. Citado na página 31.

TSCHORSCH, F.; SCHEUERMANN, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. **IEEE Communications Surveys & Tutorials**, IEEE, v. 18, n. 3, p. 2084–2123, 2016. Citado 2 vezes nas páginas 22 e 26.

WEBASSEMBLY. 2018. Disponível em: <https://webassembly.org/>. Citado na página 39.

WIJAYA, D. A. et al. Monero ring attack: Recreating zero mixin transaction effect. In: IEEE. **2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)**. [S.l.], 2018. p. 1196–1201. Citado 3 vezes nas páginas 45, 46 e 56.

WILLIAMS, S. **Meet the Newest Cryptocurrency Trend: Privacy Coins**. 2017. Disponível em: <https://www.fool.com/investing/2017/12/27/meet-the-newest-cryptocurrency-trend-privacy-coins.aspx>. Citado na página 31.

WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. **Ethereum Project Yellow Paper**, v. 151, p. 1–32, 2014. Citado na página 15.