FloodXMR: Low-cost transaction flooding attack with Monero's bulletproof protocol*

João Otávio Massari Chervinski[1], Diego Kreutz[1,2], and Jiangshan Yu[3]

[1] Federal University of Pampa, Alegrete-RS, 97546-550, Brazil
[2] CritiX/SnT University of Luxembourg, Esch-sur-Alzette, L-4364, Luxembourg
joaootaviors@gmail.com, kreutz@acm.org
[3] Monash University, Melbourne, VIC 3800, Australia
jiangshan.yu@monash.edu

Abstract. Monero is one of the first and most popular cryptocurrencies to address privacy issues of other crypto coins such as Bitcoin. Monero has a market capitalization of over one billion US dollars, and is ranked the 12th most valuable cryptocurrency on CoinMarketCap (17 April 2019). This digital coin provides different mechanisms to protect its users, such as decoy keys or mixins to obfuscate transaction inputs. However, in spite of the efforts to protect Monero's users privacy, transaction tracing attacks are still feasible. Our contribution is twofold. First, we propose and evaluate a new traceability attack, called transaction flooding attack (FloodXMR). Second, we present an analysis of the costs required for an attacker to conduct FloodXMR. We show how an attacker can take advantage of Monero's Bulletproof protocol, which reduces transaction fees, to flood the network with his own transactions and, consequently, remove mixins from transaction inputs. Assuming an attack timeframe of 12 months, our findings show that an attacker can trace up to 47.63% of the transaction inputs at a cost of just 1,746.53 USD. Moreover, we show also that more than 90% of the inputs are affected by our tracing algorithm.

Keywords: Monero · Privacy · Traceability · Attack.

1 Introduction

The buzz around Bitcoin [7] and newer cryptocurrencies led to fast pace innovation in payment systems around the world. One of the main reasons behind the accelerated widespread of digital currencies is the new distributed data structure called blockchain. Blockchains allow the creation of decentralized trustless networks where different parties can engage in transactions without needing a trusted third-party as an intermediary. In the Bitcoin network, participants reach a consensus about the valid blocks through an algorithm called *Proof-of-Work*

* This work is partially supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128. Thanks to Marcelo Rezende Thielo for his insights and discussion.

(PoW), which requires solving a computational puzzle. Transactions executed between peers of the network are validated by cryptographic signatures.

However, it is worth emphasizing that crypto coins such as Bitcoin are not regulated by a central authority. Instead, such digital coins are maintained by world-wide contributors and both the source-code and history of transactions are open and can be accessed by anyone. In the Bitcoin network, each user has a pair of public and private keys, which are used for receiving and sending payments, respectively. A user wallet has an address (160 bits long hash) that can receive funds. As these addresses are pseudonyms and (supposedly) do not reveal any information about the owner, users believed that their Bitcoin transactions were anonymous. However, reports have shown that attackers are able to associate the Bitcoin addresses of the users with their IP addresses and user names found in forums and websites [1, 2, 5].

Monero was one of the first cryptocurrencies designed to address privacy issues (e.g. avoid linking user data with their pseudonym addresses) of previous digital coins. To provide transaction privacy, Monero is built upon the CryptoNote protocol [9]. This protocol offers two main features. First, transaction untraceability. Given a transaction input with multiple keys, it should not be possible to reveal which one of the keys was used to perform the payment, preventing the transaction from being traced. Second, address unlinkability. Given two different addresses, it should not be possible to link both addresses to the same user.

Unlike the immutable pseudonym keys used in cryptocurrencies such as Bitcoin, Monero's users rely on the use of two pairs of keys, one pair of viewing keys and one of spending keys. With the recipient's public viewing and spending keys, a sender is able to generate a unique address to receive payments. This unique key is called a stealth address. It is used to protect the identity of the recipient of the transaction. Finally, the sender's information is protected through decoy keys called mixins. Mixins are output keys of previous transactions. The decoy keys are used to obfuscate the payment key.

Despite providing different protocols and mechanisms to protect the identity of its users, Monero has been a recurrent target of attacks. For instance, in previous versions of the system, design and specification flaws allowed attackers to trace a large amount of transaction input keys [3,6]. One of the first flaws was to allow users to create transactions without any decoy keys. A second flaw was a bias in Monero's mixin selection algorithm, which made it easier to identify the payment keys.

Attacks against the privacy of transactions kept in Monero's blockchain try to exploit eventual weaknesses of the system to find out the payment keys. To prevent such attacks, the latest version of Monero (release 0.13.0) imposes a fixed number of ten decoy keys for each input of every transaction. As we discuss later on, the number of mixins has a significant impact on input key tracing attacks. For instance, between February of 2017 and February of 2018 the minimum number of decoy keys required was smaller, which led us to trace up to 89% of the payment keys of that period using our transaction flooding attack. Yet,

from February of 2018 to February of 2019, Monero's system both increased the minimum (required) number of decoy keys (set to 6 mixins on March 2018) and introduced a fixed number of decoy keys (10 mixins per transaction input since October 2018). As a result, for the latter timeframe we were able to trace roughly half (47%) of the payment keys, as discussed in Section 4.

We propose and evaluate a new attack, named transaction flooding attack (or FloodXMR), to trace the payment keys of Monero's blockchain. While the idea is rather simple, i.e., to flood Monero's network with transactions whose input and output keys are owned by the attacker, we need to address some challenges and pay attention to details that significantly impact the results, such as the chain reactions. First, the transaction fee is proportional to its size in bytes, which means that the attacker should carefully chose a cost-effective size. Second, as 50% of the decoy keys of the transaction's input are from the past 1.8 days, the attacker has to create transactions continuously. Third, we explore the critical mass reactions of Monero's blockchain, as it can nearly double the number of traced input keys, as we discuss later on. Fourth, we try to maximize the number of output keys per transaction. Ideally, the attacker should create cost-effective transactions with as many output keys as possible.

Our main contribution is fourfold: (a) a new transaction flooding attack to trace payment keys of other users of Monero's system; (b) an evaluation of the effectiveness of the attack on two timeframes of Monero's blockchain (from January 2017 to January 2018 and from February 2018 to February 2019); (c) a cost analysis showing how expensive can the attack be on each timeframe; and (d) an analysis of how Monero's Bulletproof protocol helps us to significantly reduce the cost of the attack.

This paper is organized as follows. Section 2 discussed the related work. In Section 3 we introduce the transaction flooding attack. We evaluate the attack using Monero's blockchain data in Section 4. Finally, we provide a few final remarks in Section 5.

2 Related Work

To our knowledge, no existing strategy has been capable of linking real-world information to user addresses on the Monero blockchain. Known attacks against Monero, such as tampering the wallet's code and personification of a remote node, are limited to revealing the real keys spent in transaction inputs [3,4,10,11].

In earlier versions of Monero, the biased results of the mixin sampling algorithm and the creation of transactions without any decoy keys could be explored by an attacker to trace transaction inputs [3,6]. The biased sampling algorithm favored older output keys when choosing mixins to be included in a transaction input. That led to the spent key being the most recent one, i.e, generated in the highest block among all the mixins, in more than 90% of inputs created before the algorithm was updated. The lack of decoy keys happened because there was no enforcement of rules to ensure a minimum amount of mixins for each input. Inputs without mixins are traceable as they contain only the spent key.

If the spent key was chosen by the system to be included as a mixin in a future transaction it could be ignored because it was already spent before, thus it could not be the key being spent in the input. Those strategies allowed attackers to compromise Monero's privacy just by analyzing transaction data contained in the blockchain.

The Monero wallet code was also a target of recent attacks [10]. Findings have shown that an attacker can tamper a wallet code and, consequently, choose the mixins for a given input. As the system will not verify wether or not those keys are being spent in other inputs of the same transaction, this makes it possible to mark all the keys in the transaction as spent, regardless of which input each key is spent in. By setting up a fake cryptocurrency wallet service, the attacker takes control of the selection of mixins and is able to include the spent keys in transactions created by other users, reducing the Monero's essencial privacy guarantees.

Attackers can also use personified remote nodes to trace inputs in the Monero network [4]. To avoid the heavy work of making and keeping up to date a full copy of the blockchain, clients of the network commonly rely on remote nodes to request transaction information and send payments. After checking the possible mixins of the request, the node operator may abort the client transaction and wait for a retry. Upon receiving the retry request, the attacker might be able to identify the real key by searching for the one appearing in both requests.

A recent work proposed a strategy to trace transaction inputs by finding closed-sets of public keys included in transaction inputs [11]. A closed-set is defined as a set of transaction inputs in which the number of distinct keys across all the inputs is the same as the number of inputs in the set. In a closed-set each public key must have been spent in one of the inputs in the set and used as mixin in the other inputs. This allows an attacker to identify output keys that have already been spent and remove them from other inputs. As the number of decoys is reduced the privacy of the transaction inputs is weakened, allowing the execution of further attacks.

We provide a summary of the different attacks against Monero's transaction privacy on Table 1. As expected, all attacks rely on analyzing the data in the blockchain to trace transaction inputs. Some attacks use passive approaches by exploiting specific vulnerabilities of the system, such as bias in the mixin selection algorithm, and then use it to analyze and filter the payment keys [3, 6]. Other attacks use active approaches, such as setting up additional services [4, 10] or tampering with wallet code to make transactions traceable.

3 The Transaction Flooding Attack

Assuming a Monero transaction tx with one input $tx.in$ containing four keys ($tx.in = \{pk_1, pk_2, pk_3, pk_4\}$), while one of the keys (e.g. pk_4) represent the real coin being spent, the remaining three keys are being used as decoys to obfuscate the real key being used in the transaction. However, if three of the public keys (e.g. pk_1, pk_2 and pk_3) are owned by the attacker, it becomes easy to find out

Table 1. Comparison of the attacks presented in related work.

Work	Blockchain data analysis	Wallet code tampering	Creation of transactions	Service personification	Affected versions
[3]	X				Prior to 0.9.0
[6]	X				Prior to 0.9.0
[10]	X	X	X	X	All versions
[4]	X			X	All versions
[11]	X				All versions
Our attack	X		X		All versions

which one is the payment key. This is one of the most basic principles of the transaction flooding attack, i.e., the attacker has to own a list of valid output keys and make it as big as possible.

The attacker can remove or mark as known a key pk_x from the input of a transaction created by another user when the key belongs to him/her. If the attacker has not yet spent the key on a payment, s/he knows that pk_x is being used as a decoy. Second, if the attacker knows that the key has already been spent in a previous transaction (e.g. pk_4), then s/he knows that the key is a mixin as well. In both cases, the key pk_x can be safely removed from the input keys $tx.in$ of a transaction tx.

The previous example can be exploited in practice through a transaction flooding attack. This attack explores Monero's ring signature scheme, which hides the real input keys by mixing them with different output keys (used as decoy keys) generated by previous transactions. The core idea of the transaction flooding attack is simple. The attacker has to create low-cost transactions to build up a big knowledge base (i.e. list of output keys) from which the system might select keys to be used as mixins in future transactions. As discussed before, if the attacker knows all keys except one of the transaction's input $tx.in$, s/he can easily find out which key is being spent in that input of the transaction.

In Monero, every time a new transaction is created, a minimum number of mixins (dependant on system version) must be included in each input. The system selects the mixins from the output keys of previous transactions and adds them to the transaction's input, as shown in Figure 1. Each input $tx.in$ of transaction tx will have its own set of mixins. Finally, a digital signature is created to allow the receiver to get the payment without knowing the payer's keys.

The main challenge for a successful transaction flooding attack is to have enough keys so that the system selects all mixins of an input $tx.in$ from the attacker's set of keys. To own output keys, the attacker has to flood the network with valid transactions, ideally with a very low-cost, making the attack feasible.

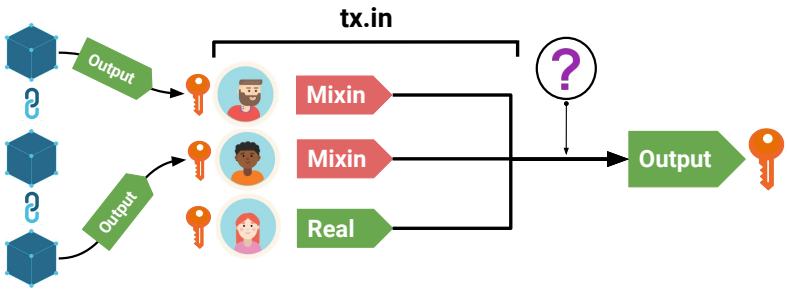

Fig. 1. Moneros' ring signature.

These transactions will have to use payment keys and receiver addresses owned by the attacker.

The number of output keys of a transaction (used as decoys in future transactions) is directly related to the number of addresses receiving coins. Each receiving address will get an output key containing a number of coins (XMR). Those output keys will be later selected by the system as mixins of future transactions.

It is worth emphasizing that each transaction has a fee. This fee is used for paying off the miners that perform computational work to validate the transaction. Moreover, the fee increases with the transaction's size in bytes. The number of inputs, mixins, and outputs have a direct impact on the transaction's size. Last, but not least, the fee varies also with the current block reward and other reference values. The amount charged in XMR for every kilobyte of data in a transaction is defined by the Equation 1[4].

$$Fee\ per\ kB = (R/R_0) \times (M_0/M) \times F_0 \times (60/300) \times 4 \tag{1}$$

where:

R = Base block reward

R_0 = Reference base reward (10 XMR)

M = Block size limit for the miner to avoid being penalized for excessive block size

M_0 = Fixed maximum block size (300 kB)

F_0 = 0.002 XMR

$60/300$ = Adjustment factor to account for the increase of maximum block size limit from 60kB to 300kB

4 = Adjustment factor for the fee multiplier. Default transactions have a value of 4 and the minimum fee has a multiplier of 1.

By the end of 2017, the transaction fee reached a peak average of 20 USD[5] mainly due to a boom on the cryptocurrency market. This example shows us

[4] https://ww.getmonero.org/2017/12/11/A-note-on-fees.html

[5] https://bitinfocharts.com/comparison/monero-transactionfees.html

that (eventually) a flooding attack can become pretty expensive. We believe that this was one of the reasons why this kind of attack was not explored by Monero's developers.

On October of 2018, the Monero community announced the launch of the Bulletproof protocol[6] to replace the *Ring Confidential Transaction* (RingCT) on the task of generating range proofs. This new protocol generates cryptographic proofs up to 97% smaller in size when compared to those of RingCT. Smaller proofs lead to smaller transaction size and, as a result, lower fees. In short, the introduction of the Bulletproof protocol made the transaction flooding attack even more interesting, as we show in Section 4.

3.1 The Attacker Model

As Monero's blockchain is public to anyone, an attacker is able to access to the Monero blockchain data. We assume that the attacker is willing to pay some (small) amount of fees in order to trace transactions. We discuss the cost of such an attack in Section 4.4.

We also assume that the attacker has created one hundred different Monero wallet addresses. One of the wallets contains the amount of XMR needed to cover the costs of the attack. Note that creating a new Monero wallet is easy and has no extra cost. One hundred addresses provide a good balance between transaction size (around 11 Kb for a transaction with 1 input and a ring size of 11) and transaction fee costs, as we discuss in Section 3.2.

Finally, we assume that the attacker is able to create as many transactions as he wants at any given time t as long as he pays the transaction fees. And it is up to the miners to select and validate the transactions, i.e., there is no timing guarantee.

3.2 Flooding Monero's Network

To trace input keys, the attacker must create a large number of output keys during the timeframe s/he wants to execute the attack. Only transactions created after the beginning of the attack are subject to tracing. As we show in Section 4, the attack gets more effective as the attacking timeframe stretches.

Monero's system selects 50% of the decoys from the output keys generated during the last 1.8 days. The remaining 50% of the mixins are selected from older outputs of transactions. To create his own output keys, the attacker has to send payments to his own addresses. Those transactions will generate new output keys which will be in the attacker's possession. Each transaction output can spend just 1 piconero (10^{-12} XMR). For the attacker, the cost of creating valid transactions is mainly the transaction fees.

To maximize the efficiency of the attack, new output keys must be created every day. Based on our experimental observations, one simple, yet effective, strategy for the attacker is to analyze the number of output keys generated

[6] `https://www.getmonero.org/2018/10/11/monero-0.13.0-released.html`

during the last 24 hours and then create the appropriate number of keys based on the percentage of the keys that s/he wishes to control. For example, if 5,000 keys were created during the past 24 hours and the attacker wishes to control 75% of the outputs, then 15,000 new outputs must be created. This strategy should be followed until the end of the attack.

The transaction fee is influenced by cryptographic mechanisms that provide privacy and security, such as range proofs used to secure the amount being sent and to prevent coin counterfeiting. With Monero's Bulletproof protocol, the increase on the fee costs, based on the size of the transaction, follows now a logarithmic function. This means that the Bulletproof significantly reduces the cost of transactions with multiple outputs. In the transaction flooding attack, we use one wallet address to pay 1 piconero to the remaining ninety-nine addresses. The payment address will receive the change.

One hundred outputs per transaction is a nearly optimal choice because it provides a good balance between transaction size, which is about 11 Kb when the transaction has 1 input and ring size of 11, and transaction fee costs. Even though the attacker can create transactions with more outputs (e.g. 1,000), the cost per output decreases very slowly for transactions with over 100 outputs, i.e., it is not worth having transactions with too many outputs.

It is worth emphasizing that bigger transactions are less likely to be chosen by miners, as the attacker is paying just the default transaction fee. A large transaction consumes a significant space in the block. From the miner's perspective, s/he can be better rewarded from lots of smaller transactions paying more per Kilobyte.

The number of mixins in each input affects the transaction fee. While this was an issue in previous versions of the system, since the 0.13.0 release, name "Beryllium Bullet", the ring size has a fixed size of 11 to achieve uniformity of transactions. Therefore, every transaction input has now exactly 10 decoy keys.

3.3 The Tracing Algorithm

The transaction flooding begins with the attacker creating new transactions. On each transaction, the attacker makes 99 payments, resulting in 99 output keys. The output keys of the attacker's transactions are stored in a single list, which will be used to trace input keys in transactions made by other users. Following, after obtaining a copy of Monero's blockchain, the tracing process consists in checking the input keys of all transactions created after the beginning of the attack. Keys owned by the attacker are removed from each input $tx.in$. The detailed tracking of input keys is described in Algorithm 1.

The tracking begins with two inputs, the list of blocks extracted from Monero's blockchain and the set of keys owned by the attacker (Line 1). If the set of keys known by the attacker increases (Line 20), then the analysis will be run again on all blocks as more input keys can be potentially traced (Lines 5 to 24). The tracing will stop only when no new real key (Line 19) can be found with the current list of keys owned by the attacker (Line 4 and 26).

Algorithm 1 Tracing input keys

```
1: procedure TRACE_INPUTS(blocks, attackerKeys)
2:     tracedKeys ← {}
3:     while true do
4:         knownKeys ← |attackerKeys|
5:         for each block ∈ blocks do                          ▷ For each block
6:             transactions ← getTransactions(block)
7:             for each transaction ∈ transactions do
8:                 inputs ← getInputs(transaction)
9:                 for each input ∈ inputs do
10:                    keys ← getKeys(input)
11:                    mixinSetSize ← |keys| − 1
12:                    mixinsRemoved ← 0
13:                    for each key ∈ keys do
14:                        if key ∈ attackerKeys then
15:                            mixinsRemoved ← mixinsRemoved + 1
16:                        end if
17:                    end for
18:                    if mixinsRemoved == mixinSetSize then
19:                        realKey ← keys − (attackerKeys ∩ keys)
20:                        attackerKeys ← attackerKeys ∪ realKey
21:                        tracedKeys ← tracedKeys ∪ realKey
22:                    end if
23:                end for
24:            end for
25:        end for
26:        if knownKeys == |attackerKeys| then
27:            break
28:        end if
29:    end while
30:    return tracedKeys
31: end procedure
```

On each iteration of Algorithm 1, all inputs of each transaction are extracted (Lines 8). For each input (Lines 9 to 23), we have to check which keys are in the set of keys known by the attacker. If the attacker knows all but one key of the transaction input (Lines 11 to 18), then the remaining key is the real one (traced key) and it is added to the list of keys owned by the attacker (Line 19 to 21).

All traced keys can be removed from transactions' inputs, allowing more inputs keys to be traced. This effect is propagated, increasing the efficacy of the attack. Once ended, the algorithm returns the list of traced keys (Line 30).

4 Evaluation

Executing our attack in the Monero network not only requires the time (e.g. 1 year) and transaction fees, but may also disrupt its services. So we simulate our

attack locally based on the real Monero data. As Monero increased the minimum number of decoy keys to 4 on February 2018, and introduced a fixed number of 10 decoy keys in October 2018, we choose to run our main experiment on Monero's one year data from February 2018, as this gives a more accurate result to reflect its new updates. We show our analysis on an old set of data (February 2017 to February 2018) of Appendix B.

To evaluate the transaction flooding attack, we extracted one year (from February 1st 2018 to February 1st 2019) data from Monero's blockchain, i.e., from block 1,499,601 to block 1,761,435. We took into account four different timeframes, namely the last 3 months of data (starting block: 1,695,128), the last 6 months of data (starting block: 1,606,715), the last 9 months of data (starting block: 1,562,861), and all 12 months of data (starting block: 1,499,601). As for the attacker's keys, we randomly selected output keys contained in the blockchain.

Additionally, we evaluated how much an attacker can benefit from a small scale attack, i.e, running the flooding attack for a few days only. We used blocks from 1,761,435 to 1,782,260 in 5 timeframes, namely the last 1.8 days (starting block: 1,780,976), the last 3.6 days (starting block: 1,779,654), the last 7.2 days (starting block: 1,777,069), the last 14.4 (starting block: 1,771,847), and all 28.8 days (starting block: 1,761,435). The timeframes are multiples of 1.8 days because that's the number of days of the recent zone for Monero's mixin selection algorithm. Half of the mixins are chosen from this zone.

4.1 Tracing Input Keys

We deploy the tracing algorithm described in Section 3.2 to evaluate the effectiveness of the attack. The results can be seen in Figure 2. Details are also available in Table 4 of Appendix A.

By owning a larger set (%) of output keys, the chances of the system choosing decoy keys belonging to the attacker are increased. Indeed, the number of traceable input keys grows with the percentage of keys owned by the attacker, as shown in Figure 2. For instance, if the attacker owns 80% of the output keys during a time span of three months, then s/he is capable of tracking around 35% of the payment keys. However, if the timeframe is augmented to one year, then the attacker will be able to trace more than 50% of the keys.

As can be observed, the attack's timeframe has an impact of the power of tracing input keys. This happens because output keys of older transactions are also selected as decoy keys by the system. The three months attack has the lowest tracing capability (roughly 43%), while in one year the attacker can trace nearly 64% of the input keys. However, to achieve these results s/he has to own 99% of the output keys. Again, the difference of more than 20% is due to the fact that transactions (in the three months timeframe) are more likely to choose decoy keys belonging to older transactions (e.g. transactions that happened between three and twelve months ago) over which the attacker has no control.

Another interesting behavior to observe in Figure 2 are the abrupt spikes between 50% and 70%. While by owning up to 50% of the keys the traceabil-

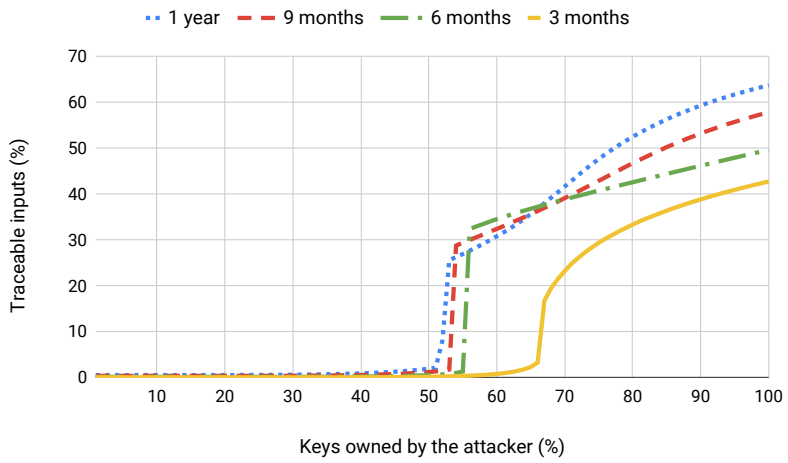

Fig. 2. Traceable input keys based on the power of the attacker.

ity of input keys is below 2%, it jumps to over 20%, for all timeframes, when the attacker owns more than 50% of the output keys. This sudden exponential growth of the tracing capability can be explained by the chain reactions on the CryptoNote protocol [8]. The critical mass reaction happens when the attacker owns enough keys to be able to trace transaction inputs and recursively uses the newly traced keys to trace more inputs, causing a chain reaction that affects a large number of inputs. It is worth mentioning that we took the chain reaction into account when designing Algorithm 1.

To have a better idea on the impact of critical mass reactions on tracing input keys, in Figure 3 we show the traceability results with the chain reaction disabled, i.e., newly traced keys are not used in the process of revealing more keys. It can be observed that the number of traceable inputs increases with the number of keys owned by the attacker, but a sudden exponential growth in the tracing power does not happen. Moreover, the number of traced input keys is significantly lower. For instance, when owning 99% of the output keys, the attacker will be able to trace just half as much input keys when compared to the previous results.

We evaluated also the traceability capability of our solution on an older dataset of Monero's blockchain, from block 1,236,197 to block 1,499,600 (from January 2017 to January 2018). Figure 5 of Appendix A shows the tracing results. As the minimum number of mixins was no more than four, the traceability power of the attacker was much higher. An attacker owning 99% of the output keys could trace nearly 90% of the input keys. However, the cost of the attack was three orders of magnitude higher, as we further discuss in Section 4.4.

Finally, it is worth emphasizing that some transactions could not be traced in the chosen timeframes, but had decoy keys removed from their inputs. That makes the inputs more susceptible to further attacks. In Table 2 of Section 4.2,

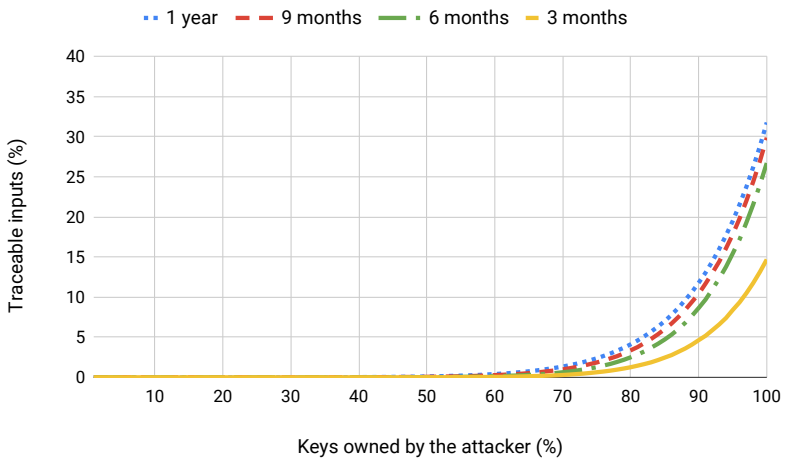

Fig. 3. Traceable input keys based on the power of the attacker (without chain reaction).

we summarize the number of mixins removed from the inputs in the one-year timeframe, showing the impact of the attack over all the inputs, including the ones that could not be traced.

4.2 Removed Decoy Keys

As expected, some transactions could not be traced in the chosen timeframes. However, we reduced the number of decoy keys of nearly all the 3,824,858 inputs belonging to the entire one-year timeframe. This makes those inputs more susceptible to further attacks. In Table 2, we summarize the number of mixins removed from inputs. Each cell contains the percentage of inputs affected by the tracking algorithm. For instance, when the attacker owns 99% of the keys, 5.15% of inputs had between 50 and 59% of their decoy keys removed.

When the attacker owns 75% or more keys, his/her tracing capability approaches its maximum. From this point on, a lot of inputs have either 6 or 10 mixins removed from their respective anonymity sets. That occurs because the ringsize is currently fixed as 11 and, consequently, there is a large number of transactions with 10 mixins. Before that, the minimum ringsize allowed was 7, leading to a large number of transactions with 6 mixins. Only the inputs that had a 0% reduction in their set of decoy keys were not affected by the transaction flooding attack. The inputs that have not been affected by the attack represent 93.85%, 30.12%, 15.08%, 10% and 9.38% of the inputs when the attacker owns 1%, 25%, 50%, 75% and 100% of the keys, respectively. As we could trace 63.36% of the input keys when the attacker owns 99% of the output keys, and 9.38% of the inputs where not affected by the attack, we have a remaining of 27.26%

Table 2. Impact of the transaction flooding attack on anonymity set sizes.

Anonymity set reduction (%)	Transaction inputs affected (%)				
	1% control	25% control	50% control	75% control	99% control
0	93.85	30.12	15.08	10	9.38
1 - 9	0.15	0.25	0.11	0.01	0
10 - 19	4.99	22.42	8.11	1.41	0.48
20 - 29	0.45	14.21	7.45	2.70	1.37
30 - 39	0.05	16.85	13.06	3.01	1.16
40 - 49	0	5.52	6.69	1.06	0.24
50 - 59	0	7.72	21.16	8.03	5.15
60 - 69	0	1.71	13.28	7.32	4.42
70 - 79	0	0.44	6.65	5.11	4.38
80 - 89	0	0.20	5.71	12.65	9.54
90 - 99	0	0	0.76	1.05	0.48
100	0.47	0.51	1.88	47.63	63.36

of transaction inputs with one or more decoy keys removed. This means that 90.62% of the entire set of inputs was affected by the tracing algorithm.

4.3 Small scale flooding attacks

We also analyzed the impact of small scale attacks using timeframes of 1.8 up to 28.8 days. Figure 4 shows the results for Monero's blockchain dataset from February 1st to March 1st of 2019. One of the first things that can be observed is the absence of critical mass reaction for such short timeframes.

For small timeframes, the tracing capability starts to increase significantly when the attacker owns more than 70% of the outputs keys. For a 1.8 days attack, it is only possible to trace inputs when the attacker has 70% of the output keys.

If the attacker owns 75% of the outputs, then s/he is capable of tracking 0.02%, 0.09%, 0.32%, 0.97%, and 3.09% of the inputs, respectively. While 0.32% seems to be a rather small percentage, it represents 320 traceable input keys in 7.2 days assuming an average of 10k transaction inputs added daily to the blockchain. With more than 75% of the output keys, the number of traceable inputs increases fast, reaching up to 0.20%, 1.20%, 4.17%, 11.31%, and 26.41% when the attacker owns 99% of the keys.

This analysis shows that an attacker can trace keys even if the attack timeframe is just 1.8 days. However, if the attacker wants to trace a significant number of input keys, s/he will have to use larger timeframes (e.g. more than 7 days).

4.4 Cost Analysis

Based on Monero's one-year blockchain data described in Section 4.1, the daily average number of outputs keys is around 11,512. We use this average as our

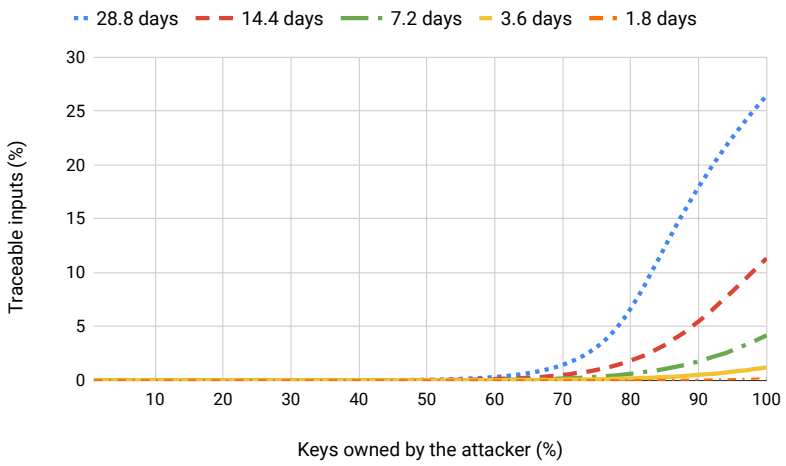

Fig. 4. Traceable input keys based on the power of the attacker

basis for analyzing the costs of a transaction flooding attack. We assume that 11,512 is the minimum number of output keys generated in a single day. Thus, if the attacker wants to control 50% of the output keys, then s/he has to generate the same amount of output keys per day, i.e., 11,512.

As discussed in Section 3.2, we assume the attacker is going to create transactions with one hundred output addresses and one input address with ten mixins. We use the default network fee per kilobyte of data in a transaction, which is currently 0.000020 XMR (0.00125 USD on April 4, 2019, when 1 XMR was worth approximately 62,92 USD). This same exchange rate is used on all the following examples. Table 5 of Appendix C summarizes the number of transaction outputs needed by the attacker to have different levels of control over the keys of Monero's blockchain during a specific timeframe.

In Table 3, we summarize the transaction fees of the flooding attack. If the attacker owns 75% of the output keys, then the yearly fee will be just 27.758 XMR (or 1,746.53 USD). This is arguably a very low-cost for an attack that is able to trace nearly 50% of the input keys. Not to mention that 1,746.53 USD is just a penny when it comes to big companies or governments that might be interested in tracing the inputs of Monero's transactions. The transaction flooding attack clearly poses a great threat to Monero's reputation and to the privacy of its users.

We did also the same cost analysis for the transaction flooding attack on the second dataset (Monero's blockchain data from February 2017 to February 2018). As the Bulletproof protocol was not active back then, the cost of the attack was nearly 1750x higher (e.g. 49,212.418 XMR, or 3,086,602.89 USD, for 75% of the keys for a one-year timeframe). Despite the high cost, it was back then a feasible attack.

Table 3. Fees of the transaction flooding attack.

Attacker control (%)	**Transaction fees (in XMR)**			
	3 months	**6 months**	**9 months**	**1 year**
1%	0.023	0.046	0.069	0.093
25%	0.760	1.521	2.281	3.084
50%	2.281	4.563	6.844	9.253
75%	6.844	13.689	20.533	27.758
99%	225.863	451.727	677.590	916.001

5 Conclusion

This work presented an analysis on the traceability of Monero's transactions using a novel attack named FloodXMR. The proposed attack consists in exploiting the Bulletproof protocol to create a large number of transactions, aiming to control a large portion of the keys that are used to provide privacy to Monero's transaction inputs.

Simulation results shown that by executing the proposed attack, a malicious actor which controls 75% of the transaction output keys generated in a one year timeframe is able to trace 47.63% of all transaction inputs created in the same time period. The results show the existence of vulnerabilites on Monero's privacy mechanisms, with emphasis on the recently launched Bulletproof protocol which was essential to making the proposed attack cost effective.

A cost analysis of the transaction flooding attack was also presented. The cost of creating the necessary transactions for the execution of the attack was evaluated and the results show that an attacker would need to spend 9.253 XMR or 582.19 USD in transaction fees in order to control 50% of the output keys in a one year period. Through the analysis of the results, we conclude that the attack's cost is low given the impact it has on the privacy of the transactions of a privacy-centered cryptocurrency.

The presented analyses emphasize the importance of detecting and patching vulnerabilities in privacy and security mechanisms provided by cryptocurrencies.

References

1. Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in bitcoin p2p network. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 15–29. ACM (2014)
2. Fleder, M., Kester, M.S., Pillai, S.: Bitcoin transaction graph analysis. arXiv preprint arXiv:1502.01657 (2015)

3. Kumar, A., Fischer, C., Tople, S., Saxena, P.: A traceability analysis of moneros blockchain. In: European Symposium on Research in Computer Security. pp. 153–173. Springer (2017)
4. Lee, K., Miller, A.: Authenticated data structures for privacy-preserving monero light clients. In: IEEE EuroS&PW. pp. 20–28. IEEE (2018)
5. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 127–140. ACM (2013)
6. Miller, A., Möser, M., Lee, K., Narayanan, A.: An empirical analysis of traceability in the monero blockchain. arXiv preprint arXiv:1704.04299 (2017)
7. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
8. Noether, S., Mackenzie, A.: A note on chain reactions in traceability in cryptonote 2.0. Research Bulletin MRL-0001. Monero Research Lab **1**, 1–8 (2014)
9. Van Saberhagen, N.: Cryptonote v 2.0 (2013), `https://static.coinpaprika.com/storage/cdn/whitepapers/1611.pdf`
10. Wijaya, D.A., Liu, J., Steinfeld, R., Liu, D.: Monero ring attack: Recreating zero mixin transaction effect. In: 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). pp. 1196–1201. IEEE (2018)
11. Yu, Z., Au, M.H., Yu, J., Yang, R., Xu, Q., Lau, W.F.: New empirical traceability analysis of cryptonote-style blockchains (2019)

A Tracing Input Keys

We used the tracing algorithm described in Section 3.2 to evaluate the effectiveness of the attack. The results can be seen in Table 4. The first column shows the percentage of keys owned by the attacker.

Table 4. Traceable input keys based on the attacker's power

Attacker's Keys	**Traceable inputs**			
	3 months	**6 months**	**9 months**	**1 year**
1%	0 (0%)	2,202 (0.09%)	9,339 (0.31%)	18,227 (0.47%)
25%	1 (0%)	2,274 (0.09%)	9,647 (0.32%)	19,573 (0.51%)
50%	1,241 (0.10%)	10,843 (0.47%)	33,967 (1.15%)	72,155 (1.88%)
75%	332,130 (29.39%)	930,143 (40.74%)	1,265,916 (42.90%)	1,821,946 (47.63%)
99%	482,813 (42.39%)	1,133,145 (49.29%)	1,706,870 (57.45%)	2,437,661 (63.36%)

In practice, it is not possible to own 100% of the output keys unless the attacker is the only participant receiving funds in the network. However, it is interesting to note that even owning 100% of the output keys, the attacker is not able to trace all input keys of the timeframe because the inputs of transactions use decoy keys from output keys generated before the attack began.

B Tracing Keys in Older Datasets

We used a second dataset to evaluate our transaction flooding attack. It begins on block 1,236,197 (requiring a minimum of 2 mixins at February 1st 2017) and it ends on block 1,499,600 (requiring a minimum of 4 mixins at February 1st 2018). On the other hand, the first dataset (see Section 4) begins at February 1st 2018 (minimum of 4 mixins, update to 6 mixins in March and 10 mixins in October) and ends at February 1st 2019.

As can be observed in Figure 5, the traceability power achieved by an attacker owning 75% of the keys in the one year timeframe is twice as much when compared to the first dataset (our recent dataset). As the number of decoy keys is very low, the critical mass reaction happens much earlier, e.g., when the attacker has just 16% of the outputs in the one year timeframe. This shows how important is the number of mixins on reducing the power of input key tracing attacks.

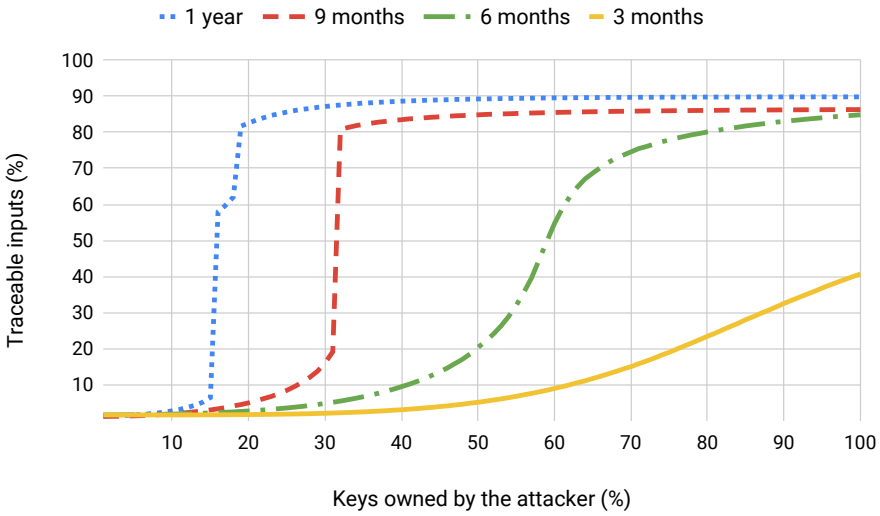

Fig. 5. Traceable input keys based on the power of the attacker

C Number of Output Keys

Table 5 summarizes the number of output keys an attacker needs to have to own the corresponding percentage of keys of the Monero network.

Table 5. Number of output keys an attacker needs to have

Attacker's control (%)	Number of output keys			
	3 months	**6 months**	**9 months**	**1 year**
1%	10,465	20,930	31,396	42,443
25%	345,360	690,720	1,036,080	1,400,626
50%	1,036,080	2,072,160	3,108,240	4,201,880
75%	3,108,240	6,216,480	9,324,720	12,605,640
99%	102,571,920	205,143,840	307,715,760	415,986,120

D Mathematical Analysis of the Attack

The attacker's traceability power increases with the number of keys s/he owns during the attack timeframe. As shown in Figure 6, the performance of the attack for the different timeframes (curves in the graph) behave similarly. At first, the attacker's tracing power increases very slowly. Then, it suddenly increases exponentially due to critical mass reactions. Following, the tracing power starts to increase slowly again, behaving like a $f(x) = x^{\frac{1}{3}}$ curve.

We chose an equation (see Equation 2) with the structure of a *Cumulative Distribution Function* (CDF) along with a power function to model the results of the transaction flooding attack. The CDF is useful for modeling phenomena where the variables exhibit different behaviors at different intervals, such as before and after reaching the critical mass point in a curve. The CDF's 'S'-like shape allows us to replicate the sudden growth upon reaching the critical mass point while the power function describes the steady increase in traceability power when the attacker's keys approach 100%.

$$w1 \times \frac{\log_e(\lambda_1 \times x)}{1 + e^{-((x-x_0)\times\lambda_2)}} + \frac{\lambda_3 \times x^j}{1 + e^{-((x-x_1)\times\lambda_4)}} \tag{2}$$

Figure 6 reproduced the transaction flooding attack results using the Equation 2. As can be observed, the outcomes of the equation are a pretty good approximation of the results shown in Section 4.1.

We used the following values for the variables of the Equation 2 to model the attack results. We have a set of values for each timeframe.

- **3 months:** $w1 = 6; x_0 = 67; x_1 = 70; j = \frac{2}{5}; \lambda_1 = 0.15; \lambda_2 = 1.2; \lambda_3 = 4; \lambda_4 = 0.2$;
- **6 months:** $w1 = 7; x_0 = 56; x_1 = 65; j = 1; \lambda_1 = 1.5; \lambda_2 = 4; \lambda_3 = 0.15; \lambda_4 = 0.1$;
- **9 months:** $w1 = 7; x_0 = 53; x_1 = 75; j = 1; \lambda_1 = 1.2; \lambda_2 = 4; \lambda_3 = 0.3; \lambda_4 = 0.08$;
- **1 year:** $w1 = 6; x_0 = 51; x_1 = 65; j = \frac{3}{4}; \lambda_1 = 1; \lambda_2 = 3; \lambda_3 = 1.1; \lambda_4 = 0.1$;

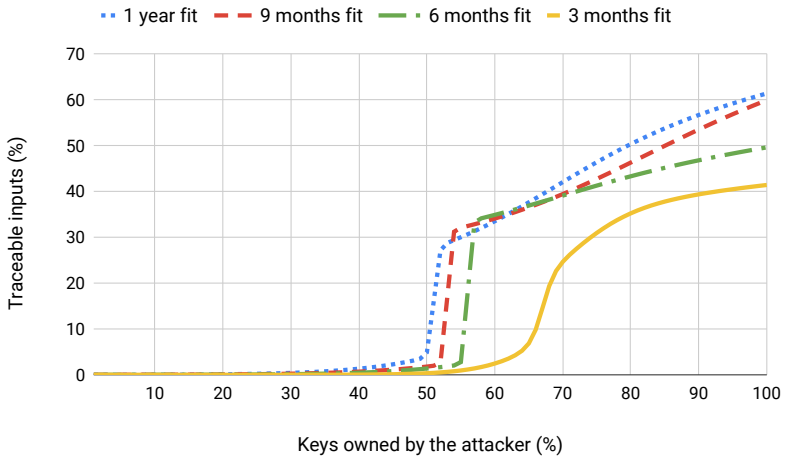

Fig. 6. Modeling the attack results with the Equation 2.