Privacy and Linkability of Mining in Zcash

Alex Biryukov
University of Luxembourg
alex.biryukov@uni.lu

Daniel Feher
University of Luxembourg
daniel.feher@uni.lu

***Abstract*—With the growth in popularity for cryptocurrencies the need for privacy preserving blockchains is growing as well. Zcash is such a blockchain, providing transaction privacy through zero-knowledge proofs. In this paper we analyze transaction linkability in Zcash based on the currency minting transactions (mining). Using predictable usage patterns and clustering heuristics on mining transactions an attacker can link to publicly visible addresses over 84% of the volume of the transactions that use a ZK-proof. Since majority of Zcash transactions are not yet using ZK-proofs, we show that overall 95.5% of the total number of Zcash transactions are potentially linkable to public addresses by just observing the mining activity.**

***Index Terms*—blockchain, cryptocurrency, privacy, linkability, Zcash, mining**

I. Introduction

Since the birth of Bitcoin [1], the popularity of blockchain research and technology has been growing at an almost unmatched pace. A blockchain consists of blocks of valid transactions, while the separate blocks are chained together with a cryptographic hash function. The idea of providing a currency without any central authority quickly grabbed the attention of many people, including researchers.

Even though originally thought to be anonymous, due to its use of a public key infrastructure as pseudonyms, Bitcoin was shown to be a lot less private in practice, as every transaction information becomes forever public, being stored in the immutable blockchain. This led to some new blockchains being designed, with their main focus being the privacy of the users. The two most popular (and relevant from research point of view) privacy-oriented blockchains are Monero and Zcash. Monero provides privacy by coupling several possible transactions with the actual spending transaction while also hiding the actual amounts using cryptographic ring signatures. Zcash's privacy is based on practical zero-knowledge proofs called zk-SNARKs. Zcash's proofs are theoretically secure, providing safety as long as the underlying Elliptic Curve Cryptography is not broken and the initialization procedure (generating a common reference string) has been performed in a secure and trusted way. Zcash's transactions that use a zero knowledge proof are called *shielded* transactions.

On the other hand Zcash also has a public part which mimics that of Bitcoin. There can be transactions which move coins from a public address to a shielded one (hiding) and vice versa (revealing). The main goal of this paper is to make an in-depth study of linkability of hiding and revealing shielded transactions, thus comparing the effect of the new anonymity features of Zcash with those of Bitcoin.

As was shown in [2], the majority of shielded transactions are connected to mining pools. Mining pools are collections of miners looking for a solution of the respective proof of work (a mathematical lottery based on finding partial preimages for a cryptographic hash function), where they mine together in order to reduce the variance of solving a puzzle while keeping the expected returns in terms of earned coins. These pools are usually managed by a single entity who handles the payouts to the miners proportional to the amount of work they provided for solving a puzzle. In Zcash a miner can only claim its reward for solving a puzzle after it moved it to a shielded address (hiding) in order to anonymize the newly minted coins, however in this paper we show that this approach does not work properly. We also point at the shortcomings of the previous analysis [2] and provide novel clustering heuristics that cover 88.2% of the mined coins tracked through the shielded pool. Overall we increase the coverage based on mining pools in terms of transaction volume from 65.6% in [2] to 84.1% on the same dataset.

This work shows that even if a blockchain is theoretically safe, bad use practices and an intermix of hidden and public transactions can lead to considerable information leakage defeating the very strong cryptographic privacy features of Zcash. Moreover, since hidden transactions form only 13.4% of the total number of transactions, 95.5% of all Zcash transactions are currently potentially linkable, which is very close to privacy level of the original Bitcoin. This situation should be gradually improving with the recent "sapling" update which speeds up the processing of hidden transactions. Nevertheless this study questions the whole concept of a blockchain with mixed public and private transactions.

A. Related Work

There have been multiple studies focusing on deanonymizing blockchain transactions [3]–[5], some with a special focus on the privacy preserving blockchains, mainly on Monero [6], [7] and Dash [8].

There has been a short paper concerning Zcash by Quesnelle [9] focusing on just one of the predictable usage patterns - the so-called round trip transactions. The paper describes the linkage of equal hiding and revealing values, where the time difference is short between them. Another recent article about Zcash is by Kappos et al. [2], which is in general more concerned about directly deanonymizing specific entities, while our work is purely focused on mining pools (which are the source of Zcash and produce majority of the shielded transactions). In the rest of the paper we will show the shortcomings and inconsistencies considering mining pools in that paper by reproducing their results while also providing novel clustering heuristics for mining pools that perform significantly better. Our paper also covers 11 months more data, which is roughly 61% more blocks and 78% more transactions, as Zcash is still only 2 years old.

II. Background

Zcash is a privacy preserving cryptocurrency launched on 28 October, 2016. Compared to Monero, Zcash's privacy is based on practical zero-knowledge proofs called zk-SNARKs [10]–[12]. These proofs are common reference string (CRS) based proofs, where the CRS was generated with a multiparty protocol.

The structure of Zcash is similar to that of Bitcoin, as the original version of Zcash was planned to be an extension of the bitcoin protocol [13]. The blockchain itself is unspent transaction output (UTXO) based, using 2.5 minute block generation time and Equihash [14] as its proof-of-work function. The current mining reward is 12.5 ZEC/block, 10 ZEC goes to the miner who found the block and 2.5 ZEC goes to the Zcash developers as the "Founder's Reward". Such a transaction is called a "Coinbase" transaction. After the first 4 years, the Mining reward will be reduced to 6.25 ZEC, but all of it will go to the miner.

The currency in the blockchain is called ZEC, while the smallest possible value is 1 Zatoshi, where 1 ZEC $= 10^8$ Zatoshi. The default transaction fee is 10^4 Zatoshi. The total supply of ZEC will be slightly less than 21 million, which is the same as in Bitcoin.

The mining is mostly done by mining pools, where the average threshold for payouts is 10^6 Zatoshi, but some pools also use 10^5 Zatoshi as the lower limit.

In general there are two types of transactions in Zcash. The first are transparent transactions. These transactions work the same way as Bitcoin transactions, with some previously unspent outputs as the inputs, and the new unspent outputs as the outputs of the transaction. The difference between the overall value of inputs and outputs is the transaction fee. They can only transfer coins between public or transparent addresses, which in the rest of the paper we will refer to as t-addresses, since in the blockchain they start with a "t". Such transactions are also called t-to-t transactions and are currently a default (this may change in the future).

The second type of transactions are the ones that send or receive coins to or from a hidden address. These addresses start with a "z" and thus in the rest of the paper we will refer to them as z-addresses. A transaction can use both t- and z-addresses, but the z-address is not revealed on the chain, only a proof that there is a valid z-address that sent or received the unknown amount of coins. In the rest of the paper we will refer to any transaction that involves a z-address as a shielded transaction.

Shielded transactions can consist of several underlying zero knowledge proofs called joinsplits, as a joinsplit only supports 2 hidden inputs and 2 hidden outputs due to technical limitations of the first version of the ZK-cryptography where the name joinsplit originates as well (the new version called "sapling" does not have this limitation). This means that if somebody wants to send or receive coins from more than 2 z-addresses, they will have to generate multiple proofs. A joinsplit has two public parameters, the amount of previously public coins, called "vpub_old" , and similarly the amount of revealed new public coins, called "vpub_new". There is a pair of these values for every joinsplit in the transaction. If we sum up every "vpub_old" value for every shielded transaction up to a block b (lets call this sum $\mathit{hidingsum}_b$), and then do the same for every "vpub_new" value as well ($\mathit{revealingsum}_b$), then the difference $\mathit{hidingsum}_b - \mathit{revealingsum}_b$ is exactly how many coins are in hidden addresses at the time of block b.

There can be 4 general different shielded transactions:

- z-to-z transactions: the simplest case is where there is no public input or output, which means the transfer is only between z-addresses. The only revealed new amount in the "vpub_new" field is the transaction fee.
- z-to-t transactions: in these transactions there is no public input, but there is at least one public output, where the sum of the outputs has to be less than or equal to the revealed new coins, while the remainder is the transaction fee.
- t-to-z transactions: in this case, there are no public outputs in a transaction, only public inputs. The sum of the inputs has to be larger than or equal to the amount of newly hidden coins, while the remainder is the transaction fee.
- tz-to-tz transactions: the last case, where zk-SNARKs are involved, but there are public inputs and outputs as well in the transaction. In this case the transaction fee is the difference between the newly revealed coins of the zk-SNARKs and the sum of public outputs.

	t-out	Yes	No	Yes
t-in	z-in/out	No	Yes	Yes
Yes	No	t-to-t	t-to-z	tz-to-tz
No	Yes	z-to-t	z-to-z	z-to-t
Yes	Yes	tz-to-tz	t-to-z	tz-to-tz

TABLE I: Every type of transaction based on the type of input and output addresses, and how they are identified. Note that we can distinguish only 5 types (only 4 that are shielded) out of the 9 possible, as we do not know whether there was a z-address as input or output, when there is a t-address as input or output.

A. Notation

Let us describe in detail the notation that we will use in the rest of the paper. The chain of blocks itself is noted with a C. The n-th block of the chain is noted with C_n. The notation $C_{[n]}$ means the first n blocks of the chain, $C_{[-n]}$ means the last n blocks of a chain, while $C_{[k,n]}$ is the range of blocks from block k to block n.

The set of transactions in the block n is noted as X_n, while $X_{[n]}$, $X_{[-n]}$ and $X_{[k,n]}$ are the same as before, but in this case containing all the transactions in these blocks in their order. Shielded transactions are noted as X^{sh}. This means, that to list every shielded transaction in a range of blocks from block k to n would be denoted as $X^{sh}_{[k,n]}$.

The inputs and outputs of a transaction $x \in X$ are simply noted as inputs(x) and outputs(x). To denote the input or output addresses of a transaction, we write inputs$_{adr}(x)$ and outputs$_{adr}(x)$. For the values of these addresses, we write inputs$_{val}(x)$ and outputs$_{val}(x)$. A single shielded transaction is noted as x^{sh}. The value vpub_old and vpub_new of the transaction is noted as x^{sh}_{vo} and x^{sh}_{vn}.

B. Tools used

An important aspect of blockchains is that, in order to verify any transaction, a full node has to keep a database of every transaction that has ever happened. This data can become quite large: the Bitcoin chain is currently more than 190GB, while the Zcash chain is 20 GB. Because of the size of the database, the tools used for the analysis become an important aspect on its own, since efficiency of the tool determines the number of different experiments we can run on the database.

We have created a tool specific for Zcash as a fork of the recently released tool BlockSci [8], specialized for Bitcoin [1] and its hard forks. It is reported to be multiple times faster than any previous tool. The tool is available at *https://github.com/cryptolu/BlockSci/*.

Zcash [15] is based on Zerocash [13] and was originally a hard fork of Bitcoin. The current commercial release is similar to Bitcoin in its main structure, and the RPC interface of the official wallet is an extension of the official Bitcoin interface. Using this, we could modify the BlockSci code to parse Zcash as well, as the tool supports parsing information from an RPC interface. We have extended the original database with the new attributes that only appear in a Zcash transaction. Let us explain below what these are.

We defined an attribute signifying that a transaction is a shielded transaction, which means it involves a zk-SNARK. If it is, then we add as extra attributes the number of joinsplits in the transaction, and their public values. Just to simplify, we can also request the sum of all of the public inputs and outputs of the joinsplits.

In our tests[1] we used the Python interface of the library, as Zcash is still fairly small in size compared to Bitcoin, and the efficiency of our functions were still manageable. As an example for the efficiency, we ran a quick test, where we cycled through every single transaction whether they involve a zk-SNARK, and if they do, we keep track of how much value was hidden and revealed overall, while also keeping track of the different kind of shielded transactions (Table I). This script finished in 6.5 seconds without any parallelization in the code (for 416,062 blocks and 3,993,633 transactions).

[1]Computer Specs: AMD Ryzen 5 2400g, 32 GB RAM, Running Ubuntu 18.04

III. Deanonymizing The Miners

In Zcash block rewards are always claimed by sending the coins to a z-address first. After that, the owner of the coin can use it freely. This led us to investigate how miners and mining pools use their rewards: do they convert it back to public addresses, and if yes, could a connection be found between these two transactions? Another aspect is that currently no mining pool (that we know of) supports payouts to a hidden address. This means that these miner payouts have to be visible somewhere on the blockchain (as they are paid to public addresses), which means the mining pool has to reveal the mined coins.

There are two general patterns for payouts. The first one is converting the mined coins back to a public address controlled by the pool, and then paying the miners in public transactions (we will call it pattern T). The second pattern is paying the miners directly from a hidden address to the public addresses (pattern Z). This means that the transaction on the blockchain appears as having no inputs (since the coins are sent from a hidden address) but having tens, hundreds, or sometimes even thousands of outputs. For both cases the single payment per address is usually in the range of 0.001-0.1 ZEC. Since this is a very specific transaction structure, it is easy to recognize.

To link transactions to mining pools, the simplest method is checking the website of the mining pool, whether they have a top miner section with the miner's public Zcash address. If this information is available, we can scan for these addresses and their latest received transactions, mapping them to the previously identified payment structures. This way we can identify which pattern the specific mining pool is using.

A. Pattern T Mining Pools

In the case of pattern T, after a constant public address is found, the rewards are transferred directly to a set of addresses controlled by the same entity a (signed as controlled(a)). By summing up the total amount of received coins, they become linkable to the specific input of the mining pool and the direct connection between the hiding and revealing transactions can be made (Heuristic 1).

Heuristic 1: Pattern T Heuristic, with a starting address a

```
procedure PatternTPool($a$)
    PoolAddrs $\leftarrow$ controlled($a$)
    PoolTxs $= \emptyset$
    for $x \in X^{sh}_{[k,n]}$ do
        if $\exists$ outputs$_{adr}(x) \in$ PoolAddrs then
            PoolTxs $\leftarrow x$
        end if
    end for
    return PoolTxs
end procedure
```

Even if the information on top miners is not available, the payout transactions of a mining pool can be still found. To find pools using pattern T, first scan through every shielded transaction disregarding the ones that are already identified. Calculate how much ZEC any public address received from a hidden address in the same range of blocks as before, and then compare and correlate these values to how many blocks different mining pools mined. If one extends this approach to multiple scanned block intervals, the link becomes even stronger. We used this approach to find the corresponding addresses and will describe the results in more detail in section III-C.

We have tested this approach with a varying set of block ranges and accuracy requirements. The block ranges we used were 500, 1000, 2000, 4000, 8000 and 10000. In terms of accuracy of a match we used 10%, 5% and 1%. In our tests there was no overall block range that worked for all of the pools. The reason for that is that mining pools transfer their received coins through a shielded address on a different schedule. Some pools shield their coins instantly, while others might wait to accumulate thousands of coins to hide and then later reveal them. The later would give reason to using larger block ranges (e.g. more than 4000). On the other hand if a pool changes its address often, the long range might not provide enough matches, while a shorter range would work better.

The biggest drawback of this method is that it is difficult to link small mining pools that have only mined a handful of blocks, as in those cases there are always multiple matches for amount of revealed values. In statistical terms this is not a huge problem, as these are only marginal pools and they provide less than 1% of the mining power (number of blocks mined). If higher precision is necessary one could mine in these smaller pools, to identify the relevant addresses.

B. Pattern Z Mining Pools

For pattern Z, the connection between the hiding and revealing transactions is not trivial, as there is no single constant address, instead hundreds - or sometimes thousands - of addresses. However the actual miner addresses will reappear regularly in the pool reward transactions with frequency depending on the power of their mining hardware and on the frequency of the pool payouts. Thus we scan every shielded transaction and find the ones with the pool-payout structure (i.e. lots of outputs). Once a pattern Z transaction is found we check its outputs, and look for overlapping addresses with the already existing set of miner addresses. If the number of overlaps exceeds a certain threshold (e.g. $\geq$40), we consider that transaction to be sent by the same mining pool, and also expand our set of miners with the new addresses. Scanning iteratively through a range of blocks until a new transactions and miners can be added to the existing sets, it is possible to find most of the transactions connected to a pool (2).

The drawback of this approach is, that it is only usable for shorter periods of time, e.g. 2,000 blocks ($\sim$4 days), as we have observed that miners sometimes change their mining pools. If the range of blocks is too large, because of the migrating miners one might consider a transaction from a different pool to be the same as the currently investigated one, since the number of overlapping addresses would become too high. If that happens even for one transaction, from that point on the heuristic might identify even more transactions from the different pool, creating a very large set of transactions and miners of multiple mining pools.

Heuristic 2: Pattern Z Heuristic, with an sx as a starting transaction

```
procedure PatternZPool($sx$,$X^{sh}_{[k,n]}$)
    Miners $\leftarrow$ outputs$_{adr}(sx)$
    PoolTxs $\leftarrow$ $sx$
    OldMiners $= \emptyset$
    while OldMiners $\neq$ Miners do        ▷ Execute, until the miner set cannot be updated anymore
        OldMiners $=$ Miners
        for $x \in X^{sh}_{[k,n]}$ do
            if $|$outputs$_{adr}(x) \cap$ Miners $| \geq 40$ then
                Miners $\leftarrow$ outputs$_{adr}(x)$
                PoolTxs $\leftarrow$ $x$
            end if
        end for
    end while
    return PoolTxs
end procedure
```

The accuracy of the identified transaction can be verified by adding up the overall value of the payouts and the number of blocks the mining pool actually mined, and then comparing these two values whether they are close to each other. The algorithm accepts if the difference is small ($\leq$5%).

Let us call the set of remaining not-yet-linked mining power from block k to n *UnknownPower(k,n)*. To find a pool using pattern Z that does not have a top miners section on their website we use the following approach. First we disregard every shielded transaction that has been already identified. Then we look for transactions that have tens of outputs. For every such transaction we check with the previous method (2) for overlapping addresses in them, add up the overall received value and compare it with the number of mined blocks per mining pool (3).

Choosing the correct threshold values is a non trivial task, and because of that we have tested the heuristic with a wide selection of them. We ran the algorithm with differing block ranges (the same ones as for pattern T), while we have also used different number of overlapping miners (10, 20, 40, 80). Our experiments show that using smaller overlap sizes (10) causes over-identification of transactions because of miners who either mine for multiple pools, or switch pools in-between, while a too large threshold value will not identify enough transactions. This problem becomes worse with increase in the block range, as it is more likely that more miners will switch a mining pool in a larger timespan.

C. Results of the Heuristics

If we consider the entire chain, then our heuristics linked 88.4% of the mining reward movement in the shielded addresses. In the following table (Table II) we show the exact

Heuristic 3: Heuristic for finding a pattern Z style mining pool without a base_tx

```
The set of uncovered miner transactions are used as MinerTxs
UnknownTxs = $X^{sh}_{[k,n]} \setminus$ MinerTxs
NewPools = $\emptyset$
for $x \in$ UnknownTxs do
    NewPools ← PatternZPool($x$,UncoveredTxs)
end for
for $p \in$ NewPools do
    if $\sum_{x \in p}$ outputs$_{val}(x) \in$ UnknownPower(k,n) then
        $p$ is the set of payout transactions for the pool with the matching mining power
    end if
end for
```

results per mining pool, where we also provide the amount of value mined by the pool to show that our linking was unique. Overall our heuristics linked 84% of the volume of z-to-t transactions, if we add the simple Founder Heuristic from [2]. Without those large valued transactions we cover 70.3% of the volume.

Name	Pattern T	Pattern Z	Mined Value	Linked Portion
Flypool	14,435	94,277	1.79M ZEC	0.995
F2pool	1,075	0	1.35M ZEC	0.994
Nanopool	0	40,083	338K ZEC	0.981
Poolin	126	0	138K ZEC	0.996
Suprnova	12,920	0	167K ZEC	0.961
Coinmine.pl	0	7,204	78K ZEC	0.925
MiningPoolHub	7,598[2]	0	156K ZEC	0.999
BitClub Pool	67[2]	0	1.9K ZEC	0.969
DwarfPool	2,953	0	27K ZEC	1.0
Slushpool	3,027	0	49K ZEC	0.999
Antpool	378	0	93.8K ZEC	0.999
Zpool.Guru	88	0	824 ZEC	1.0
Nicehash	203	0	429 ZEC	0.999
Luxor	185	0	6K ZEC	1.0
Solo Miners	3,698	0	43.8K ZEC	1.0

TABLE II: Results from our heuristic per mining pools, only for pools and miners where the linked transactions by our heuristics were verified by the overall mined values

Below we show how many transactions our heuristics linked this way and how many transaction were in the different categories before and after this process.

- Remaining shielded transactions: 179,057 (originally 534,944, 66.5% have been linked)
- t-to-z: 58,557 (originally 222,306, 73.6% linked)
- z-to-t and maybe to-z: 192,138 (originally 280,754, 68.4% linked)
- z-to-z: 14,431 (8% of the remaining transactions)
- tz-to-tz: 17,453 (9.7% of the remaining transactions)

In terms of thresholds, as we have mentioned earlier we have experimented with many of them, and for Pattern Z we achieved the best results for a range of blocks of 2,000 and an overlap of 80 for Flypool and Nanopool, while if we remove those transactions from our set, then for Coinmine.pl the best results were achieved by an overlap of 20 miners and a block range of 10,000. Choosing a too small overlap and/or too large block range results in marking too many transactions because of miner migration and mining for multiple pools, while too large overlap and too small block range will result in marking not enough transactions.

[2]We could only verify MiningPoolHub's payouts from block 193,000 and BitClub Pool's payouts from block 120,000

D. Accuracy of the Heuristics

In case of pattern T payouts, the transaction linkage is sound and verifiable by comparing the number of blocks mined by an entity and the amount of ZEC the suspected address received in the same period. Indeed, these two values only differ by a small amount (5%), and there is no other entity with a similar mining power in the inspected interval (more than 10% difference).

In case of pattern Z payouts, the verification is done similarly, but in this case it can not be decided whether every single transaction was found, as a statistically negligible number of payout transactions could be missed. On the other hand, if an attacker starts with a payout transaction from a different mining pool, the resulting set of transactions will be disjoint, if the parameters are set correctly and the overall payed out value will match the number of mined blocks as well. Both of our heuristics are verified with the results in Table II.

IV. Comparison of results to previous work

Below we attempted to reproduce the results (Table III) of the paper by Kappos et al. [2], where we followed the instructions step-by-step provided by their paper. We also compare our verified results to theirs on the same dataset.

Name	Kappos et. al. [2]	Reproduction	Our Work
Flypool	3	3	67,985
F2pool	720	717	717
Nanopool	4,107	3,568	19,984
Suprnova	0	0	11,185
Coinmine.pl	0	0	6,678
Waterhole	5	5	0
BitClub Pool	1,516	1,210	101
MiningPoolHub	0	0	1,335
DwarfPool	1	1	2,833
Slushpool	0	0	941
Coinotron	0	0	0
Nicehash	0	0	203
MinerGate	0	0	0
Zecmine.pro	0	0	0

TABLE III: Comparison of results with [2] in terms of number of linked z-to-t transactions, reproduction of their and our Pattern T (1) and Pattern Z (2) heuristics on the same dataset

In their paper they claim to uncover 120,629 of the z-to-t transaction that are connected to miners, but based on the paper and the data they report one can only count their Founder transactions (1,953) and the ones in Table III, which overall sums up to 8,305. On the other hand we have implemented their heuristics, and the heuristic did return 98,274 miner transactions accounting for 62.8% of the overall value (which is more than their reported 52.1%). If we remove the restriction of at least 100 outputs in the transaction, we uncover $\sim$122,000 transactions accounting for 69.4% of the overall value. We could not find out what causes these discrepancies.

We uncovered these results using every address that ever mined a block. When we reduced the addresses to only ones controlled by a mining pool, the results reduced to 6,853 transactions and 26.9% of the revealed value. This harsh reduce in results was caused by the following. There were 3 addresses that mined a single (or very few) block that are controlled by a large exchange. If we include all transactions where the receiver addresses are controlled by someone who mined a block, we include all the addresses that are controlled by the exchange, which results in thousands of addresses. As a lot of miners mine directly to an exchange address, most of these transaction are actually payouts but they are not found based on the logic presented in the paper. This is reinforced by the discrepancy between their reported number of uncovered transactions (120,629) and the number of z-to-t transactions linked in Table III.

Our opinion is that the authors did not consider the difference between how many blocks somebody mined and how much value was comparatively payed out. With our implementation of their heuristic there are cases where a miner mined only 1 block in the history of the chain, but are somehow responsible for the payouts of thousands of transactions. Otherwise we might have misunderstood something based in the paper, but we tried to implement the heuristics literally.

A. *Inflation of results*

The results of [2] in terms of % is inflated by the behavior of F2pool. F2pool is one of the largest mining pools that used a payout structure that was a mixture of pattern-T and pattern-Z payouts which we describe in detail.

First, let us refer to the address corresponding to F2Pool that received the minted coins in a coinbase transaction as T_{F2}. As we have described earlier, in order for a miner to use the newly minted coins, it has to spend it first to a shielded address. In case of F2Pool this happened on average 1.6 times a day. Then in a single revealing transaction the pool paid out all of its miners, where the value of the revealing transaction was the same as the value of the hiding transaction minus the transaction fee. Among the output addresses one address that also receives coins is T_{F2}. Then later the next time F2Pool is hiding its minted coins among the inputs of that transaction is the output of the previous revealing transaction.

This constant loop of coins inflates the revealed amount of coins from a shielded address. After calculating it exactly, we have found that this single loop is responsible for 492 thousand coins of the overall 3.788 million coins (13.1%) revealed during the timespan of [2]. If we remove the loop and only consider coins that were not part of it their original 65.6% result is reduced to 60.5%, while our approach decreases from 84.1% only to 82% (considering only mining and founder transactions in both cases).

References

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

[2] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An Empirical Analysis of Anonymity in Zcash," *ArXiv e-prints*, May 2018.

[3] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[4] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 127–140.

[5] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin p2p network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 15–29.

[6] A. Miller, M. Möser, K. Lee, and A. Narayanan, "An empirical analysis of linkability in the monero blockchain," *arXiv preprint arXiv:1704.04299*, 2017.

[7] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of moneros blockchain," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 153–173.

[8] H. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," *arXiv preprint arXiv:1709.02489*, 2017.

[9] J. Quesnelle, "On the linkability of zcash transactions," *arXiv preprint arXiv:1712.01210*, 2017.

[10] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 415–432.

[11] N. Bitansky, A. Chiesa, Y. Ishai, O. Paneth, and R. Ostrovsky, "Succinct non-interactive arguments via linear interactive proofs," in *Theory of Cryptography*. Springer, 2013, pp. 315–333.

[12] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 238–252.

[13] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 459–474.

[14] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric proof-of-work based on the generalized birthday problem," *Ledger*, vol. 2, pp. 1–30, 2017.

[15] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," Tech. rep. 2016-1.10. Zerocoin Electric Coin Company, Tech. Rep., 2016.