Deanonymization of Hidden Transactions in Zcash

Alex Biryukov, Daniel Feher
University of Luxembourg
firstname.lastname@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 describe the general attributes of this blockchain, the privacy of the typical usage scenarios while also investigating the information leakage provided by the interaction between hidden and public addresses. Using predictable usage patterns and clustering heuristics an attacker can link to publicly visible addresses over 85.8% of the transactions that use a zero-knowledge proof. Overall 97.9% of Zcash transactions are potentially linkable to public addresses.

1 Introduction

Since the birth of Bitcoin [11], the popularity of blockchain research and technology has been growing in an almost unmatched pace. The idea of providing a currency without any central authority quickly grabbed the attention of many people, including researchers. The number of blockchains alone is currently around 1,600 according to CoinMarketCap, one of the popular blockchain statistics websites. Bitcoin's market cap alone is currently 156 Billion $.

Even though originally thought to be anonymous, due to its use of pseudonyms, Bitcoin was shown to be a lot less private, as every transaction information is public and stored on the blockchain. This led to some new cryptocurrencies being designed, with their main focus being the privacy of the users. The two most popular blockchains are Monero and Zcash. While Monero provides privacy by built-in mixing capabilities using 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 initialization procedure was performed in a secure and trusted way.

In this paper we provide advanced empirical analysis of privacy on the Zcash blockchain, describing various transaction types and observable patterns as well as important entities and actors. Our techniques can reveal over 85.8% of the hidden theoretically secure transactions. Majority are connected to mining pools and their payouts, public exchanges and transactions with predictable values. After we remove this set of relatively easily linkable transactions we look deeper into the remaining hidden transactions (about 3K transactions per 10,000 blocks – about a 2.5 weeks period). We propose new linking techniques based on subset-sums and value fingerprinting to potentially deanonymize 30% of those harder cases. We also describe what we call a Danaan-Gift attack, which is a small donation which tags a larger value, with a tag potentially still visible after the value is shielded, transacted in secret and then de-shielded.

This work shows that even if something is theoretically safe, bad use practices can lead to considerable information leakage. Moreover, since hidden transactions form only 14.5% of the total number of transactions 97.9% of all Zcash transactions are currently linkable like in the Bitcoin blockchain. This study questions the whole concept of a blockchain with mixed public and private transactions.

1.1 Related Work

There have been multiple studies focusing on deanonymizing blockchain transactions, and general payment characterization papers [2, 9, 14], and a few of them focused on the privacy preserving ones as well, mainly on Monero [8, 10] and Dash [6]. There has been a short paper concerning Zcash as well [13] focusing on just one of the predictable usage patterns - so called round trip transactions. After this research was finished we learned that related article [7] has been uploaded to the arxiv.org on 8-May-2018. This work is capable of linking 69.1% of shielded transactions while we cover 85.8% of them. This is probably due to our better clustering algorithms and more careful study of the mining landscape while the other work is more focused on clustering addresses of exchanges and tracing the money flow of high profile actors like the Founders or Shadow Brokers. Moreover we can link 30% of the hard-core private transactions.

2 Background

Zcash is a privacy preserving cryptocurrency launched on 29 October, 2016. Compared to Monero, Zcash's privacy is based on practical zero-knowledge proofs called zk-SNARKs [3, 4, 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. The blockchain itself is unspent transaction output (UTXO) based, using 2.5 minute block generation time and Equihash [1] 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 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 transaction work the same way as a bitcoin transaction, with unspent outputs as the inputs, and the new unspent outputs as the outputs of the transaction, while 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, as in the blockchain the public key of these addresses always starts 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 any transactions that are sending or receiving coins from a hidden address, and the public key for these addresses always starts with a "z". In the rest of the paper we will refer to these addresses 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 JoinSplit or JS transaction, referring to the underlying name and technology presented in the Zerocash [15] paper.

JoinSplit transactions can consist of several underlying joinsplits, as a joinsplit only supports 2 hidden inputs and 2 hidden outputs (technical limitations of the currently used cryptography), which 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 JoinSplit transaction up to a block b (lets call this sum $hidingsum_b$), and then do the same for every "vpub_new" value as well ($revealingsum_b$), then the difference $hidingsum_b - revealingsum_b$ is exactly how many coins are in hidden addresses at the time of block b.

There can be 4 general different JoinSplit 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 coins 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 a joinsplit is 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 joinsplits and the sum of public outputs.

$output$ \ $input$	t	No	Yes	No	Yes
t	z \ z	No	No	Yes	Yes
No	No	–	–	–	–
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 1: Every type of transaction based on the type of input and output addresses, and how they are identified. Note that we can distinguish only 4 types 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.

2.1 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. A specific 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. JoinSplit transactions are noted as X^{js}. This means, that to list every JoinSplit transaction in a range of blocks from block k to n would be denoted as $X^{js}_{[k,n]}$.

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 JoinSplit transaction is noted as x^{js}. The value vpub_old and vpub_new of the transaction is noted as x^{js}_{vo} and x^{js}_{vn}.

3 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 grow quite large: the Bitcoin chain is currently 170GB, while the Zcash chain is 15 GB. Because of the size of the database, the tools used for the analysis become an important aspect on its own as well, as the 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 [6], specialized for Bitcoin [11] and its hard forks. It is reported to be multiple times faster than any previous tool.

Zcash [5] is based on Zerocash [15] 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 are these.

We defined an attribute signifying that a transaction is a JoinSplit 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 JoinSplit transactions (Table 1). This script finished in 37 seconds without any parallelization in the code (for 306,964 blocks and 2,849,771 transactions).

4 Main Statistics

At the time of writing, the theoretical supply (Figure 1) of Zcash is 3,712,056.25 ZEC. In reality it is less than that for 2 reasons. First, in some cases the miners did not claim any transaction fees in the mined block. Second, in the first 20,000 blocks of Zcash, the block reward was increased incrementally by 62,500 Zatoshi in every block, which already contained the founder's fee, until it reached the 12.5 ZEC block reward at the 20,000th block. The problem was that some miners did not use the correct reward for some blocks, but the blocks were still accepted, and are now part of the main chain. Because of this the actual value that is currently in circulation and available to spend is 3,711,910.94156848 ZEC, i.e. a discrepancy of about 145 ZEC.

	Block	Theor. Value	Real Value
Unclaimed fee	26718	12.50032323 ZEC	12.5 ZEC
Incorrect Reward	14708	9.19335171 ZEC	7.35472671 ZEC

Table 2: Examples of incorrectly claimed block rewards

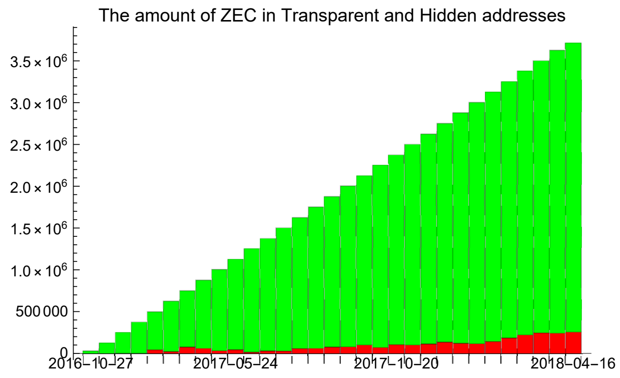

Figure 1: Supply of ZEC over time (Green), and fraction stored in hidden addresses (Red)

To draw a general picture about the blockchain, we have created several graphs describing different attributes. We investigated what are the most common output values and how are these values distributed (Figure 2,3,4). The peak at 0.01 ZEC value in Figure 2 is caused by the mining pools' threshold payout value. In general, these graphs are dominated by the miner payout outputs. Even though miner transactions are not a majority inside the total number of transactions, they dominate in terms of the number of output values since a single miner payout transaction can have thousands of outputs.

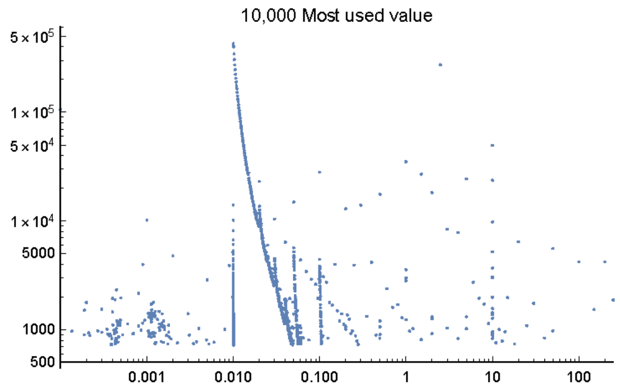

Figure 2: Value vs frequency for 10,000 most frequent output values (log-log scale)

We have also observed how is ZEC spent, meaning how long does it take for a received output to be spent. We have generated two graphs, where the first one (Figure 5) describes the fraction of outputs spent in the various time windows, while the second one (Figure 6) describes the total value spent in the same time windows. One may wonder what are the two spikes first around one month after the launch of Zcash, and the second around 24 May 2017.

[1] Computer Specs: Intel I7 3770K, 8 GB RAM, Running Ubuntu 16.04

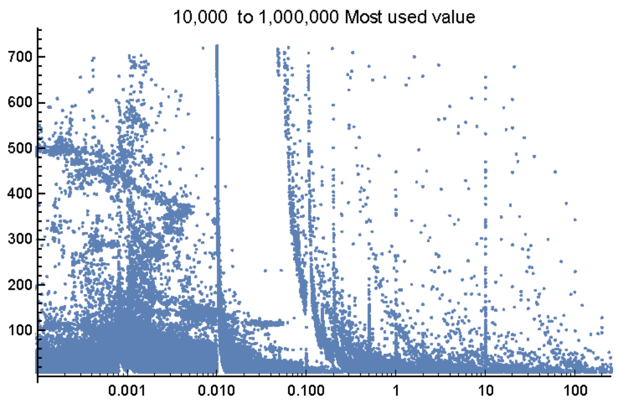

Figure 3: Value vs frequency for the 10,000 – 1,000,000th most common output values

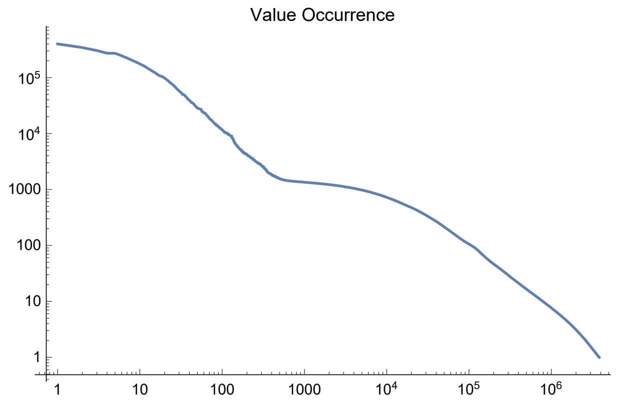

Figure 4: Rank versus frequency plot of the most common output values

The first spike is probably related to the USD/ZEC exchange rate, as on 25 November, the rate increased to 100 USD/ZEC. The second spike is connected to the news of the Zcash developers announcing a partnership with JPMorgan on 22 May 2017, which resulted in the exchange rate more than doubling from 110 USD/ZEC to a brief 290 USD/ZEC, and later normalizing around 250 USD/ZEC.

4.1 Detecting Fake Coins

There is a concern in the Zcash community that there is no way to check the total amount of coins in the system. If somebody breaks the underlying cryptography or (more likely) finds a software bug in the JoinSplit transactions, they could mint new coins out of thin air in a covert way, as was the case in CryptoNote-based currencies. The question is: could blockchain analysis help to detect it or provide some guarantees that it is not happening? The key to answering this question is to keep track of the correct supply of coins (as in the previous section).

The main problem is that we can only keep track of how much money is hidden or revealed in a JoinSplit transaction. If somebody would create fake coins with breaking a JoinSplit transaction, as long as it is not transferred to a public address, we would not be able to recognize this. On the other hand, even if the fake coins would be transferred to a public address, as long as there are still

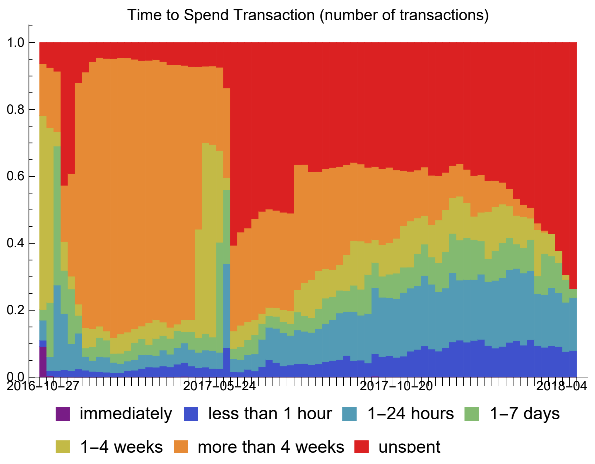

Figure 5: Percentage of time to spend a transaction output based on the number of outputs

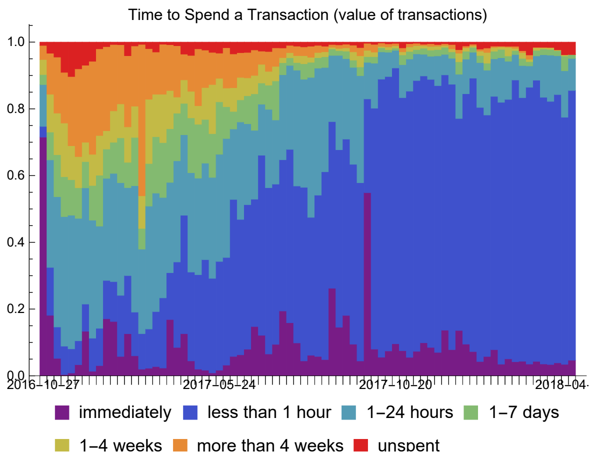

Figure 6: Percentage of time to spend a transaction output based on the value of outputs

available coins in hidden addresses (which we can keep track of) we can not detect the fake ones.

This means we are only able to realize the creation of fake coins in one case which is the following. The attacker has to create and move to t-addresses more coins, than the amount of coins in the hidden z-addresses. If this would happen, one would see that there is a negative amount of coins in the hidden addresses, which is impossible. To be undetectable, the speed at which the attacker can move his fake coins to the public t-addresses should be smaller than the speed of growth of the z-address pool.

4.2 Usage of JoinSplit transactions

An interesting question is how often users use the JoinSplit transactions, as these transactions are what differentiates Zcash from other usual blockchains, like Bitcoin and Litecoin. We will show these statistics first as a total number since the genesis block, and then as periodic data with a period of 10,000 blocks (about 2.5 weeks).

All time as of 16th of April 2018 (Figure 7,8):

- number of transactions: 2,849,771
- Joinsplit transactions: 413,116 (14.5%)
    - t-to-z: 172,865 (41.8%)

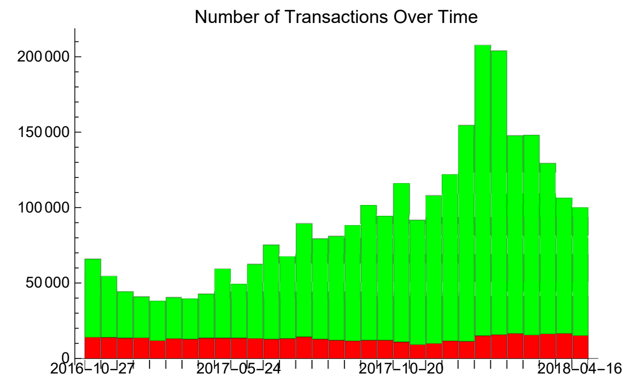

Figure 7: Number of public and JoinSplit Transactions over time

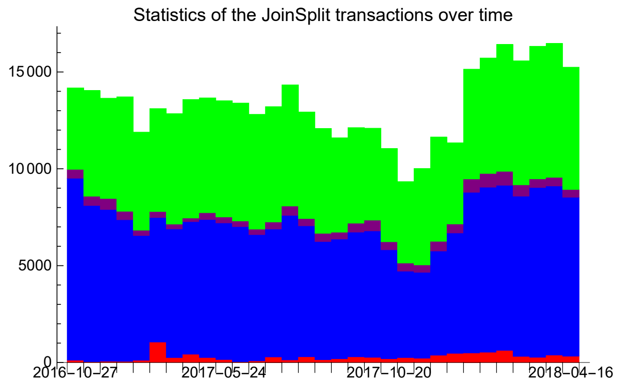

Figure 8: Breakdown of JoinSplit transactions. Green: t-to-z, Blue: z-to-t, Red: z-to-z, Purple: tz-to-tz

- z-to-t and maybe to-z: 218,134 (52.8%)
- z-to-z: 8,750 (2.1%)
- t-to-t with z involvement: 13,367 (3.2%)

Last 10,000 blocks as of 16th of April 2018:

- number of transactions: 99,853
- Joinsplit transactions: 15,248 (15.3%)
  - t-to-z: 6,317 (41.4%)
  - z-to-t and maybe to-z: 8,196 (53.7%)
  - z-to-z: 334 (2.2%)
  - t-to-t with z involvement: 401 (2.6%)

4.3 Address Control

Another interesting statistic is the number of address controllers on the blockchain. Address Control and ownership are not equivalent, as for example an exchange controls all its addresses, but users can transfer and withdraw money from the exchange. We have recorded 2,170,512 different public keys that have appeared on the blockchain (by 16 April, 2018). But not all of these keys are controlled by different entities.

We have tried to identify the groups of addresses controlled by separate entities. The first heuristic we used for this is from the observation, that in order to use multiple addresses as inputs for a transaction, the creator of the transaction has to know the private keys for all those addresses, which means he is controlling them. This attribute is transitive as well, which means that if x_1 has addresses a,b, and x_2 has b,c as its input addresses, then the same entity controls a,b and c. This heuristic leads to a Union-Find algorithm which identified 703,247 different address controllers.

Then we applied the heuristic called ”change address protocol,” described in detail in [9]. In short, the algorithm considers an address to be ”change address” if it had only one input and one output in its history, and has no coins left. According to the ”bitcoind” wallet using the default setting for transfers the remaining coins of a transaction are automatically transferred to a new address, instead of sending it back to the input address. By tracking these change addresses, one can associate them to the same controller. With this technique the number of address controllers has been reduced further to 211,883.

In the rest of the paper, for a single address a we will denote the set of addresses that is controlled by the same entity which controls a as controlled(a).

4.4 Statistics Without Mining Transactions

Our techniques have identified the majority of the mining transactions, since the reward claiming transactions can be found trivially, and the mining pool payout transactions were uncovered with the methods described in Section 6. We were interested in how the general statistics of the Zcash blockchain would look after removing all of these transactions from our analysis.

For example, one of the interesting observations is that the value usage frequency graph without mining rewards (Figure 9) is close to linear in a log-log representation. Here we sort all transaction values observed in the blockchain (excluding mining rewards) by their frequency. Then we plot the rank of a value versus its frequency in a log scale. The linear line matches the frequency distribution $f(n) = \frac{6 \cdot 10^4}{n^{0.75}}$, where n is the rank of the specific value. This means that the distribution of values follows the Zipf law which occurs in many areas of science (ex. frequencies of words in a natural language, financial transactions). Also interesting is that mining output values ”spoil” the picture if we add them to the plot as can be seen in Figure 4, since they follow a different distribution and are also affected by mining-pool payout thresholds.

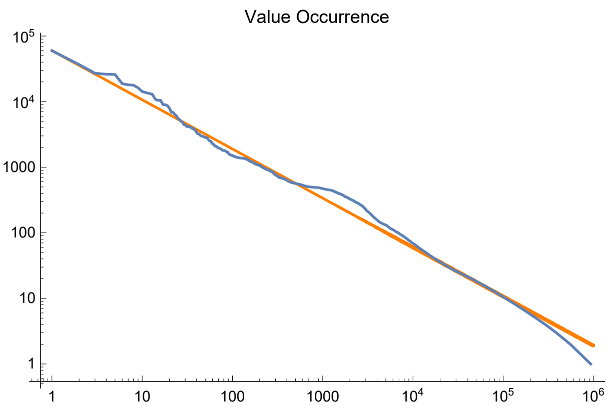

Figure 9: Rank versus frequency plot of the most common output values (without mining outputs)

We have re-generated some of the previously shown graphs, where we either removed the mining transactions and outputs completely (Figure 11,12,13,14), or marked them separately (Figure 10).

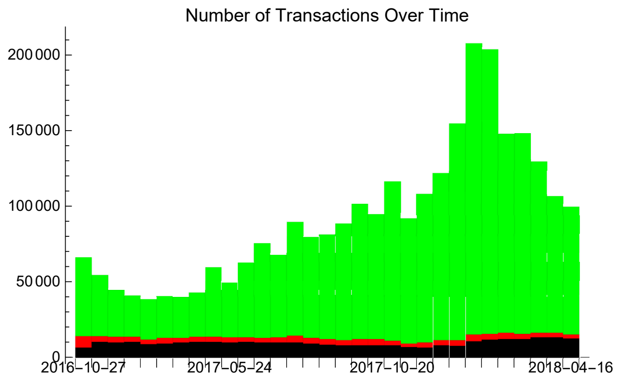

Figure 10: Number of transaction over time. Green: public transactions, Black: deanonymized miner transactions, Red: remaining JoinSplit transactions

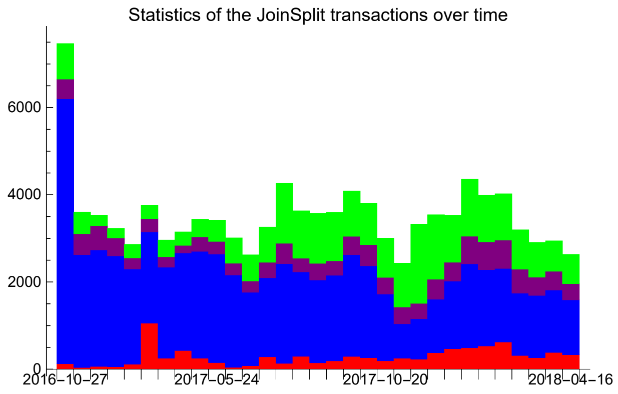

Figure 11: Breakdown of the remaining JoinSplit transactions after removing the mining transactions. Green: t-to-z, Blue: z-to-t, Red: z-to-z, Purple: t-to-t

If we compare Figures 2,3 with Figures 12,13, one can see that the difference is the most frequent output values are missing from Figure 12. We know that the difference between the graphs is the missing mining output values. The mining output values are grouped mostly around 0.01 ZEC and other smaller round values, as these are the usual mining payout thresholds.

Figure 14 is essentially a combined distribution of values vs frequencies from Figures 12,13 but not in log-scale and showing with different colors the popularity of the first decimal digit of the transaction value, from violet - 1, blue -2,..., red - 9. This shows that values starting with digit 1 are much more frequent.

5 Basic Economics

We have investigated the basic economics of Zcash, and found a few interesting figures (Figures 15,16,17,18). In the graphs the addresses are grouped based on the amount of Zcash held on each address, and the grouping is done in a logarithmic base (e.g. the first group are the empty (spent) addresses that do not have any money left, the second group are the addresses that have between 0 and 10

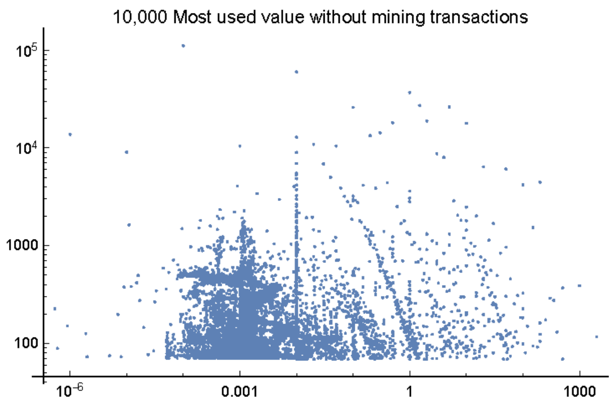

Figure 12: Value vs frequency for 10,000 most frequent values without mining outputs (log-log scale)

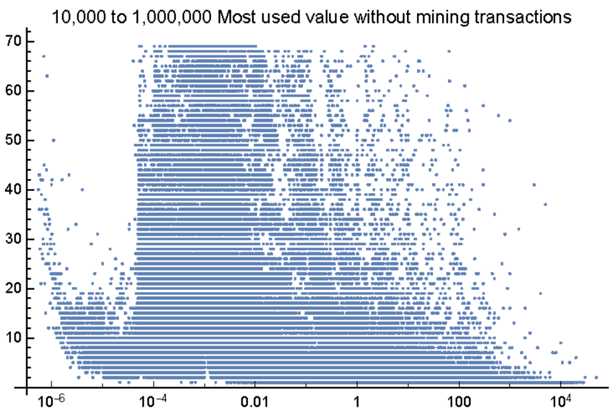

Figure 13: Value vs frequency for the 10,000–1,000,000th most common output values without mining outputs

Zatoshi, then $10 \leq x < 100$, etc. until the highest grossing address, which is a bit more then 100,000 ZEC, where 1 ZEC is 10^8 Zatoshi). In Figure 15 the bin $[10^{-2}, 10^{-1}]$ ZEC sticks out because 10^{-2} is a typical mining reward threshold.

We were also interested in the traffic between the largest and most used addresses (Figures 21,22). We used three separate metrics to distinguish an address from the rest, the notation is also described in Figure 20:

- Address Balance: the difference of received and sent ZEC over a time period (If we look at the whole blockchain, it is equivalent with the amount of ZEC held on an address)
- ZEC Flow: the amount of ZEC that went through the address over a time period (all time, last 10 thousand blocks, etc.)
- Number of Transactions: the exact number of transactions that address was part of over the same time period as before.

In the following sub-sections we will describe what is shown in these information flow graphs and what are their main characteristics. In these graphs we also clustered multiple addresses which are in the same set of controlled(a) and show them as a single node.

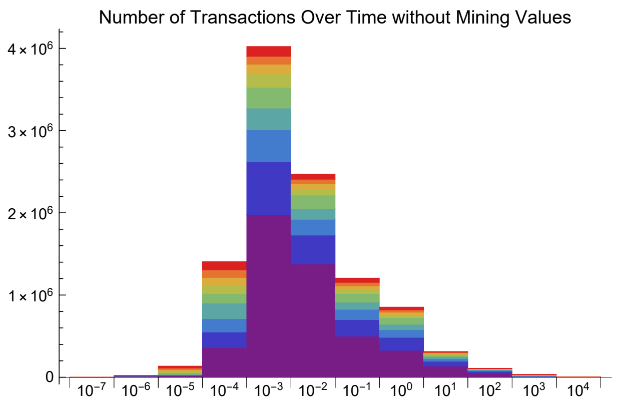

Figure 14: Frequency of output values without mining outputs

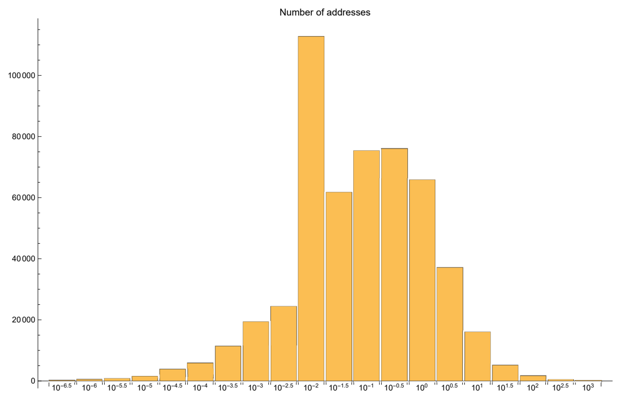

Figure 15: The number of addresses in each group, excluding the empty ones

5.1 Z-in and Z-out nodes

First, we introduce two new actors that do not appear in other Bitcoin-like blockhains. We create a node for values sent to the ”wormhole” of z-transactions. We call it a wormhole, as ZEC coins disappear in it and then appear in an absolutely new place. We present the entry to the wormhole with a black circle and the exit from it as a grey circle. Here (Figure 21,22) for example one can see all the mining pools (with colour green) claiming their block rewards by sending it to the wormhole. On the other hand one can see a lot of transactions towards miners (with colour red) from the wormhole exit - the grey node. We also see the activities from the Zcash corporations itself (marked with blue).

5.2 Mining pools

Mining pools are represented with the colour green. Most of the mining pools are just simply sending values to the wormhole to claim the block rewards, but here are a few that may seem out of place. We have looked at these addresses, and found that in some cases the mining pool first transfers all of the funds to a public t-address, and then pays out the miners from that address, in which case we still considered those as regular mining pool addresses. Also there could be questions about why some nodes are not connected to either the z-in or z-out nodes. This is because we did not draw edges where the total flow was

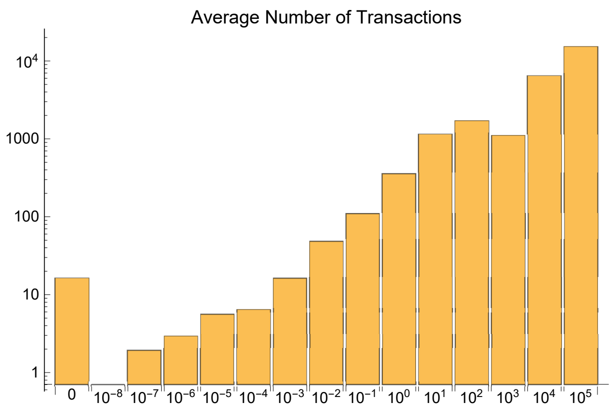

Figure 16: The average number of transactions per group.

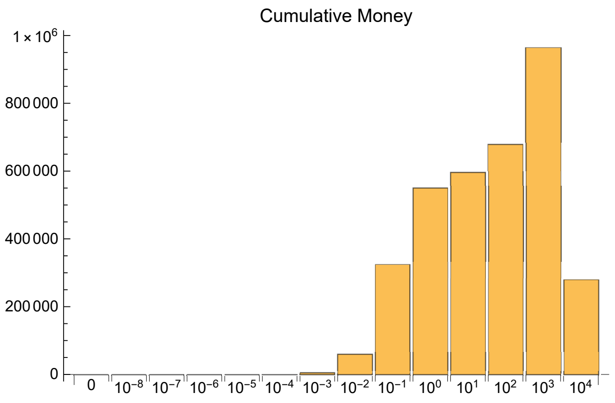

Figure 17: The cumulative ZEC held by the groups

lower then a 1,000 ZEC.

5.3 Miners

Miners are represented with the colour red. One would maybe expect more of them on this graph, but the fact is, we found that a lot of miners are mining directly to an address controlled by an exchange. That is the reason why we can see edges from the Z-out node to multiple potential exchanges.

Our techniques for finding the miners’ and mining pools’ exact addresses are described in Section 6. These methods have identified 477,099 different public addresses, that are presumed to belong to miners. Out of these 197,229 addresses are controlled by exchanges, which means that over 41% of the miners are mining directly to an exchange.

5.4 Zcash corporation

The Zcash corporation can be found by inspecting its usage patterns. One can easily see, that they always claim the founder’s rewards by batching them into 250 ZEC large inputs. Then with a closer look one can see a pattern of 250 ZEC coins reappearing from Z-out. It is easy to calculate, that they have received so far $\sim$ 300,000 founder’s rewards, summing up to around 740 thousand ZEC, while if we add up the deanonymizing transactions from Z-out for the value 250 ZEC we get 525 thousand ZEC revealed. It is with a high probability, that the ad-

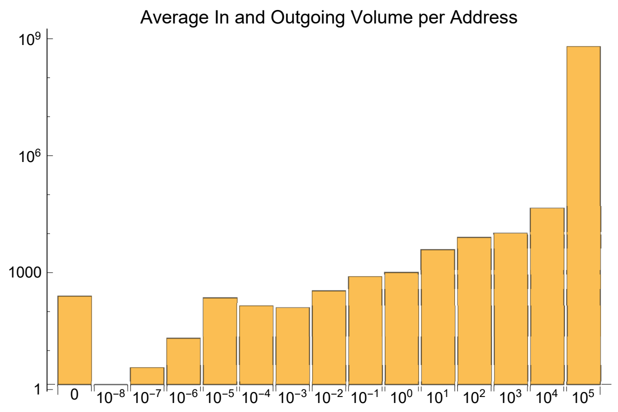

Figure 18: The average amount of coin traffic per group

dresses that receive these coins are either controlled by the Zcash company or foundation, or were sent to such an address by them.

5.5 Exchanges

Finally, we take a look at exchanges. They are presented with a yellow colour in the graph. One can easily see, that these addresses are the center of the flow of coins. We managed to find the exact addresses to some of the most popular ones, like Poloniex, Kraken, Bittrex, Bitfinex, etc.. Most of the big exchanges work with a central address, where they keep the liquidity of the exchange site. When somebody transfers coins to an exchange site, the coins are sooner or later transferred automatically to the main address of the exchange site. We found these exchanges' addresses by either simply searching forums or we have created accounts on these sites and transferred some small amounts onto them from our own funds.

It is also clearly visible, that these addresses are connected to each other with large traffic flowing between them, signaling direct trading and exchange between the sites as well. It is either traffic by exchange site owners or people withdrawing coins from one exchange to another. Another interesting attribute of these addresses is their "leaves". These are addresses not connecting to anything else. We have investigated them, and found that most of them are so called "sinks", with large amounts of coins on them, without even a single outgoing transaction. These are probably investors, who just bought Zcash as an investment and do not use it regularly, or addresses of the exchange site, to simply store their liquidity separately, either on a hardware wallet or something similar (so called "cold storage").

We also have addresses that we only suspect to be exchanges, those are marked with a darker shade of yellow. They are only suspected, because we have not found any tangible proof yet. A typical structure for an exchange, apart from the leaf addresses is that when it is an output in a transaction, those transactions generally have many inputs (for example, more than 40), and most of the inputs are of small values, generally less than 1 ZEC. On the other hand, when the address is spending an unspent output, those transactions generally do not have many outputs, as they are usually withdrawals of a single or a handful of users.

6 Deanonymizing The Miners

In Zcash block rewards are claimed by sending the coins to a z-address. 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 pay 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 pair 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 is the specific mining pool using.

6.1 Pattern T Mining Pools

In the case of pattern T, after a constant public address is found, the rewards are transferred directly to the set of addresses controlled(a). By summing up the total amount of received coins, they become connectable to the specific input 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 $x \in X^{js}_{[k,n]}$ do
        if $\exists$ outputs$_{adr}(x) \in$ PoolAddrs then
            PoolTxs $\leftarrow x$
        end if
    end for
    return PoolTxs
end procedure
```

Even if the information of 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 JoinSplit transaction disregarding the ones that are already identified. Calculate how much ZEC did any public address receive from a hidden address in the same range of blocks as before, and then compare these values to how many blocks did different mining pools mine. If there is a close match, and the address is an input to many transactions with the structure of pool-payouts, that address can be associated with the corresponding mining pool. Another method for pattern T pools is scanning for revealing transactions, where the output value is close to 10, or a multiple of 10, as the miner reward is 10 ZEC plus the transaction fees. If an address found this way is an input to transactions categorized as payouts, guess, which mining pool is it connected to, but in this case there are usually multiple candidates with the same amount of blocks mined.

6.2 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. On the other hand because there are a few candidate transactions based on the top miners sections, and from the outputs of these transactions a large set of miner addresses can be created that are supposedly mining for the same pool. Using this information, scan every JoinSplit transaction and find the ones with the pool-payout structure (i.e. lot of outputs). Once a pattern Z transaction is found, 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. ≥50), 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).

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

```
procedure PATTERNZPOOL($sx$,$X^{js}_{[k,n]}$)
    Miners ← outputs$_{adr}(sx)$
    PoolTxs ← $sx$
    OldMiners = $\emptyset$
    while OldMiners ≠ Miners do ▷ Execute, until the miner set can not be updated anymore
        OldMiners = Miners
        for $x \in X^{js}_{[k,n]}$ do
            if $|$outputs$_{adr}(x) \cap$ Miners $| \geq 50$ then
                Miners ← outputs$_{adr}(x)$
                PoolTxs ← $x$
            end if
        end for
    end while
    return PoolTxs
end procedure
```

The drawback of this approach is, that it is only usable for shorter periods of time, e.g. 10,000 blocks (2.5 weeks), 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, because 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.

The accuracy of the identified transaction can be verified by adding up the overall value of the payouts, how many blocks did the mining pool actually mine, and then comparing these two values whether they are close to each other. Accept if the difference is negligible (1-2%).

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

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 = $X^{js}_{[k,n]} \setminus$ 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* then
        $p$ is the set of payout transactions for the pool with the matching mining power
    end if
end for
```

6.3 Results of the Heuristics

Below is the current ranking of the mining pools based on the last 10,000 blocks (16th of April, 2018). The total number of transactions was 99,852, while the number of JoinSplit transactions was 15,249. Our methods have found all of the payout transactions for the significant mining pools, which corresponds to the 97.2% of the total mining power of the Zcash network (several smaller miners/pools have been identified as well, which add up to 1% mining power):

- Flypool: 5,753 Blocks (pattern Z)
- Nanopool: 1,171 Blocks (pattern Z)
- Miningpoolhub: 1,092 (pattern T)
- F2Pool: 680 Blocks (pattern Z)
- A single large miner not in a pool: 505 (pattern T)

- Coinotron: 234 (pattern Z)
- Suprnova: 158 (pattern T)
- CoinMine.pl: 79 (pattern T)

If we consider the entire chain, then our heuristics linked 92% of the mining reward payouts, and if focused on the last 3 months only, this number increases to 96.8%. We have also included a graph (Figure 19) describing the mining power distribution over time, where the last column is described in detail above.

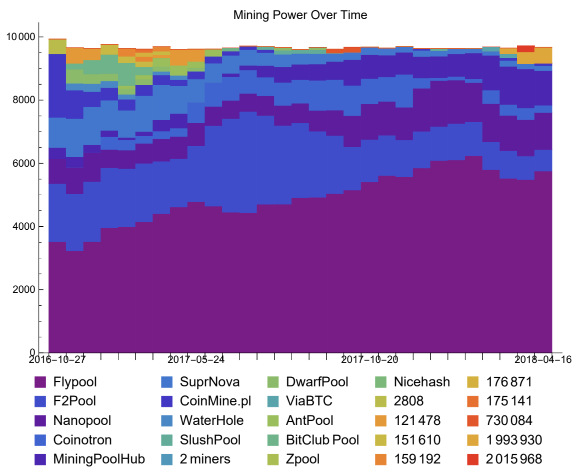

Figure 19: Mining power distribution over time. If there is only a number presented, then we could not find the mining pool for that miner, possibly it is a single miner

Below we show how many transactions did our heuristics deanonymize this way and how many are left after this process. As every t-to-z transaction is removed, where the input was a block reward, and every transaction that was identified and associated to either the Zcash corporation (described in section 5.4), or to a mining pool the techniques resulted in the following numbers (Last 10,000 blocks as of 16th of April 2018):

- Remaining JoinSplit transactions: 2,661 (originally 15,249, 82% have been deanonymized)
- t-to-z: 669 (originally 6,317, 88.5% deanonymized)
- z-to-t and maybe to-z: 1,258 (originally 8,197, 84.6% deanonymized)
- z-to-z: 334 (12.3% of the remaining transactions)
- t-to-t with z involvement: 400 (14.7% of the remaining transactions)

7 Other Deanonymization techniques

We have looked into other ways of deanonymizing transactions, where we connect different inputs and outputs of JoinSplit transactions. For this part, we removed all the identified transactions concerning miners.

The first approach is a direct match of input and output values, where there is no other occurrence of the specific value. We can still find that up to about 10% of the values return a perfect match on the input and output side. The remaining transactions are about 1,200 z-in and z-out transactions each, and 330 z-to-z transactions in a 10,000 block span. We did some further analysis on these remaining values. If we add up the amount of coins flowing into and out of hidden addresses in these remaining transactions, the inputs sum up to 29,424 ZEC, while the sum of outputs is 33,676 ZEC, where around 2,800 ZEC in the outputs is still from mining rewards, and there are plenty of founder's rewards not revealed either, while they are already subtracted from the sum of inputs.

7.1 Subset sum

We have investigated the usefulness of subset sums for deanonymizing JoinSplit transactions by connecting a single t-to-z transaction to multiple z-to-t transactions, or vice versa. The idea was to check if users hide their value in a single transaction, but reveal it over time in multiple payments, or the same way hiding their coins in multiple phases, while revealing it in a single transaction.

First, we had to consider how many numbers can be summed up overall. If we consider the average number of remaining JoinSplit transactions, we see that about a 1,000 coin revealing and hiding transaction remain for every 10,000 blocks after removing the mining transactions. If the largest possible value is approximately 10^{14} Zatoshi (100,000 ZEC, 1 Zatoshi, we allow this many different values), then having 10^7 number of combinations is the birthday bound of possible values. If we consider the number of possible combinations of 1,000 inputs or outputs, then it is easy to see that even $\binom{1000}{3} = 1.6 \cdot 10^8$. This means, that even the sum of 3 values could be just a random match.

We have still calculated the possible 3 matches as well, but found it improbable in the end, as for the $\approx$ 1000 values, while we only found subset sums of 3 values for 220 and 263 values respectively, these sums were possible overall in more than a 1000 ways, which means that for every value that had a solution with a sum of 3 values, it had on average 5 possibilities. This could still be useful for the attacker since this a small anonymity set.

However since we focused on unique identification, we only investigated sums of 3 values if there was no possible 2 sum, and only one possible 3 sum. Our method found matches for 129 outputs and 181 inputs respectively. Out of these matches there are a few, that are the same value appearing multiple times as an input or an output.

The technique did find a handful of interesting matches, especially in cases where one member of the sum was a value with high entropy (in detail in Appendix A), while the other is a round value (e.g. the input value is 3.54156325 ZEC, while there are 2 outputs with the values 0.40002 ZEC and 3.14154325 ZEC[2]). This led to further analysis idea, which is explored in the next subsection.

Also, even though in these values for inputs the fee is already counted, and for outputs it is not yet counted, which means they should match on both sides, we still see values, where the difference is exactly 10,000 Zatoshi,

[2]These are fictitious values to preserve the privacy of the actual transaction.

which is the default transaction fee. This leads us to an explanation, that the value was also moved once as a z-to-z transaction, and then the receiver is revealing it to the public. Either a user moved the coins to himself inside Z wrongly assuming that he gets more anonymity this way or more plausibly just a change of ownership happened inside Z (and not gaining anonymity compared to a change of ownership happening with t-addresses).

7.2 Fingerprinted Values

We have also found another promising technique for connecting different z-input and output values, that had no direct connections so far. We will use an approach that we call *value fingerprints*. In our definition, a fingerprint of a value is the last 7 digits of a transaction value in Zatoshis (to remind the reader 1 ZEC = 10^8 Zatoshis). The last 4 digits are especially stable as a fingerprint since this value is below the typical transaction fee of 10^4 Zatoshis (which is currently below 3 US cents). Thus they usually have little economical meaning and represent just a remnant of the previous transactions. The distinguishability of a fingerprint depends on its entropy (Appendix A), which in this case describes how frequent is the value itself. Intuitively round values are much more frequent than random values.

Using fingerprints, the attacker can link inputs and outputs in the following way. First fix the input value, and check for outputs that have the same last 4-7 digits. If the last 4 digits are round (e.g. '0000'), then we only record a match when both values are at least 0.1 ZEC large, the output is smaller and was recorded in a succeeding block compared to the input, and the two values are close to each other (i.e. $|\frac{InputValue}{OutputValue} - 1| < 0.01$). If the digits have a high entropy, then the input simply has to be bigger and had to be recorded in an earlier block than the output. This way in our analyzed 10,000 block range, the technique found 83 matches, and at least half of them seem to have a very good probability to be linked.

In the previous section 7.1 the heuristic linked some transactions, where one of the values in the sum is exactly 10,000 Zatoshi. These transaction are linked with fingerprint technique as well, as the last 4 digits are the same, and the values are very close to each other as well.

After inspecting the remaining values that were not linked with any of the two techniques, we have found that the majority, about 2/3 of them were round values with low entropy, while the other remaining values had no connections to each other.

7.3 Danaan-Gift Attack (Malicious Value Fingerprinting)

Fingerprints can be used as a tool for deanonynimization of an address, considering that 1,000 Zatoshis are worth 0.3 US Cents, if the exchange rate is 300 USD.

Suppose the attacker is trying to identify the spending of a public address, who converts all their ZECs to hidden addresses regularly. The attacker can transfer very small carefully chosen amount of Zatoshis to this specific address, and hope that it leaves the trail of a fingerprint when they are converting it in and out of a hidden address. As the attacker sees the current public value on the address, he sends a chosen value such that the resulting sum has a detectable fingerprint (different fingerprints for different addresses). For example making that the sum or its lower digits are a high entropy value. Afterwards the attacker just has to monitor the z-t transactions for his fingerprints.

Such attack can be performed against entities which accept public donations (e.g. WikiLeaks) since for them receiving money from unknown source would look less suspicious. Moreover the attacker may monitor the address and resend the fingerprint in case another donation erases his old tag.

This attack is not 100% successful, but the likelihood of success is not negligible, as it considerably reduces the transaction's anonymity set at a relatively low cost for the attacker. Moreover, in some cases a larger transaction with a specific fingerprint value masquerading as donations might be even less suspicious for the address owner.

8 Suggestions for Using the Zcash Anonymity Features

Even though Zcash is theoretically secure, there are malpractices which destroy privacy or increase linkability of the JoinSplit transactions. Below are some recommendations on using the Zcash privacy features.

Avoid converting the exact same amount of ZEC that was received to a z-address. This is the easiest to link, especially if the transaction value has a high entropy, that is easily tracked. Splitting the value into several rounded parts might avoid this (see Section 7.1). Also one needs to avoid transferring them to different addresses, when it is easily deducible that those addresses are controlled by the same entity (section 4.3).

Do not use hidden transactions as a mixer To some users z-addresses may seem as a provably secure cryptographic mixing service. It is not. As we have shown in the paper, converting coins to a hidden z-address, and then moving them to a public t-address in full, does not prevent linkability.

Reveal round values of ZEC Revealing coins in round values increases the anonymity set of a transactions, as these values can usually be associated with a lot of addresses, reducing the likelihood to find a perfect input-output match. This might also mean that some small values are left in a Z-address, but if we consider our fingerprinting technique, the last 4 digit are worth less then a US cent. Even then, if your transaction value has a high entropy then even converting the last 4 digits to 0s might not be enough. For example, 5.27835109 ZEC truncated to 5.2783 ZEC still produces a value which is unique to the whole blockchain and thus is trivially linkable.

Only reveal as many coins as needed. This is the most general, but probably the most useful practice as well. Unnecessary z-to-t transactions lead to information traces, that could be combined for deanonymization, as shown in the previous sections. Any trace can be used to reduce the number of potential unlinkable transactions.

9 Acknowledgement

This work was partially funded with the grant from the Zcash foundation.

References

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

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

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

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

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

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

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

[8] Kumar, A., Fischer, C., Tople, S., and Saxena, P. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security* (2017), Springer, pp. 153–173.

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

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

[11] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.

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

[13] Quesnelle, J. On the linkability of zcash transactions. *arXiv preprint arXiv:1712.01210* (2017).

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

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

A Entropy of transaction values

Transaction values appearing on the blockchain reflect some real-world economical meaning or information about external world (ex. values connected to mining rewards, exchange rates, cost of services, etc.). There could be two approaches to the detection of ”special” values on the blockchain. The first one is to study the frequencies of values on the blockchain. More frequent values represent some common economic activity, on the other hand the more frequent is the value the larger its anonymity set. The rare (high entropy) values are of interest to the attacker since they may serve as transaction fingerprints and may help with deanonymization.

The second approach is to look at the patterns of the specific value itself, just considering it as a string of bits and checking its ”compressibility” (sort of Kolmogorov complexity), regardless of its frequency on the blockchain.

We have defined our own entropy function, which is trying to combine the two approaches. For example, a value which is high entropy in the compression metric but low entropy in the first metric (frequent), might correspond to an exchange rate with another currency.

B Main Entities and Transaction Flow

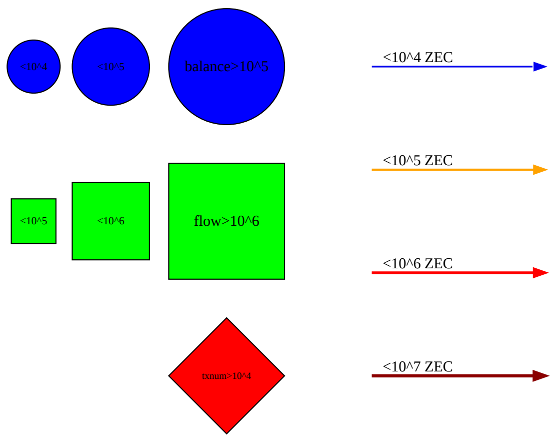

Figure 20: The legend for our flow graphs. The shape of the node determines in which category (circle is balance, square is flow, diamond is number of transactions) is that specific node in the top 300 of all nodes, and if a node is in multiple categories, then it represents in which category is it rated the highest. The node size represents the amount of balance, flow or transactions on a logarithmic scale. The edge colour and width represents the overall amount of flow between two addresses on a logarithmic scale. If there is traffic in both directions, then the edge has an arrow on both end. If the edge is bidirectional, but the colours would be different for the two directions, then we draw the corresponding halves with the correct colour.

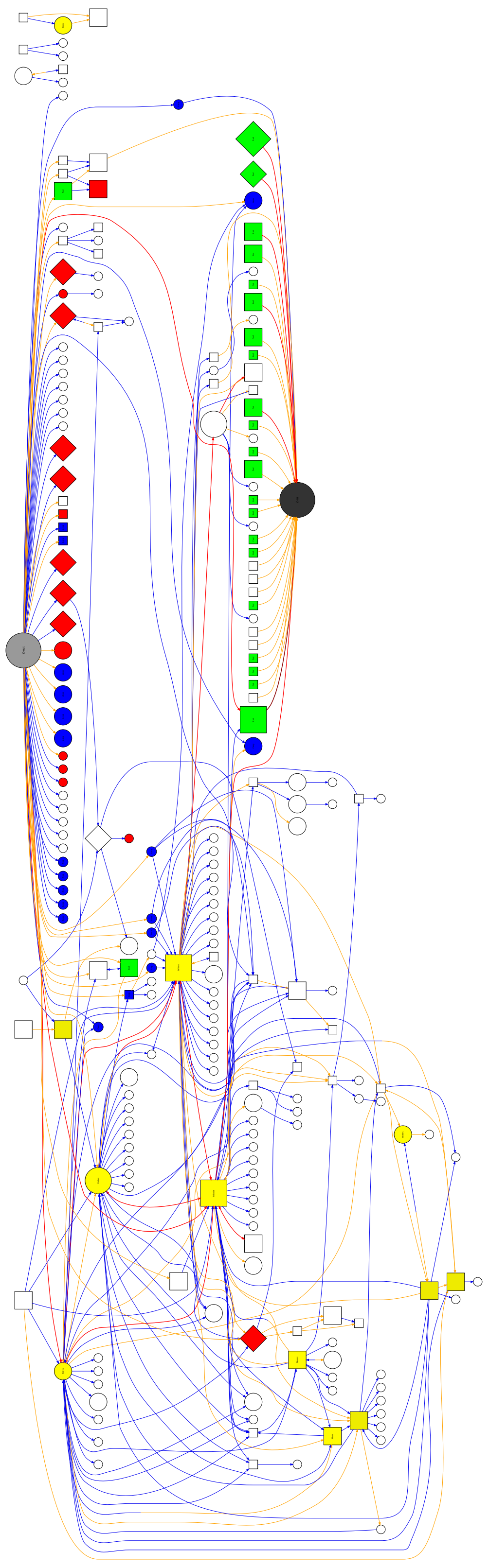

Figure 21: Network Graph for the top 300 address in every metric, where the minimum value of an edge is 1,000 ZEC

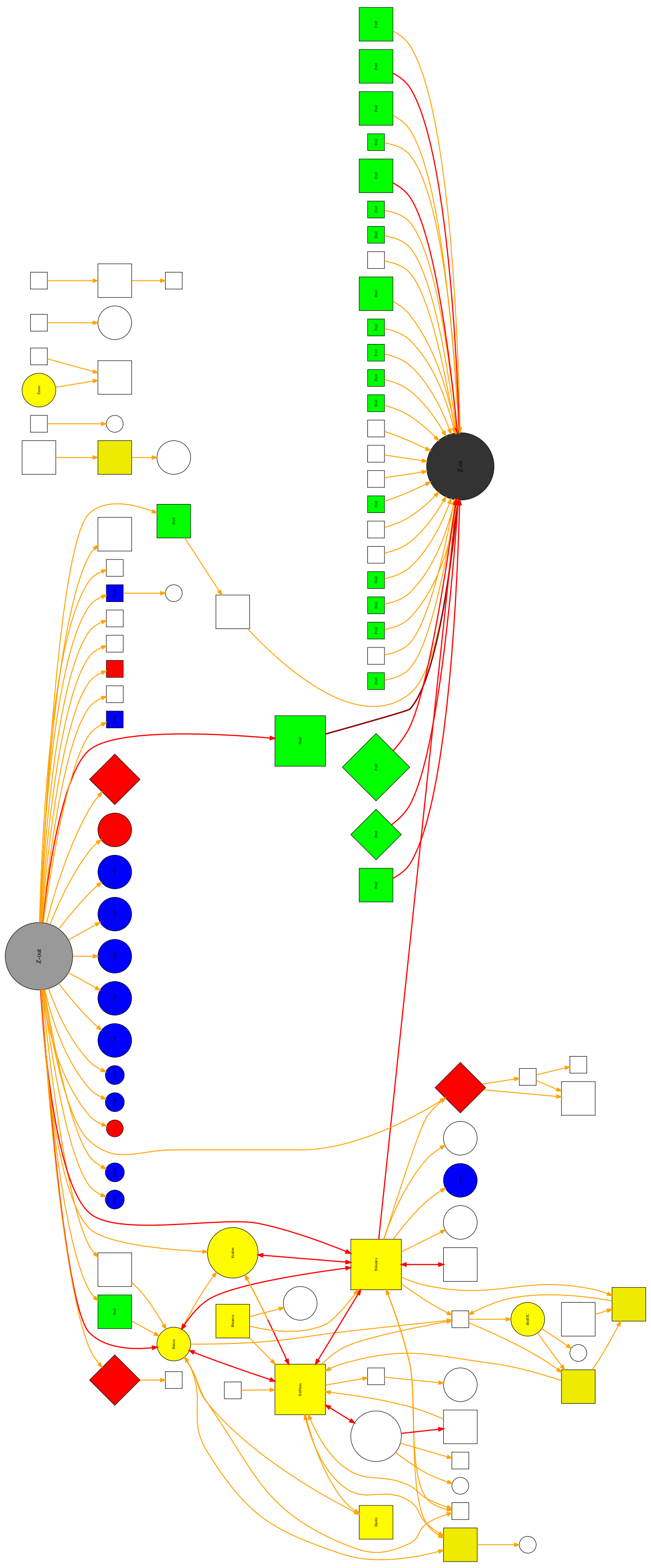

Figure 22: Network Graph for the top 300 address in every metric, where the minimum value of an edge is 10,000 ZEC