Towards an Anonymity Metric Analysis of Blockchains

Pablo Melana-Dayton and Shuai Wang

March 14, 2018

***Abstract*— Many users of cryptocurrencies seek the ability to create anonymous payments. Many cryptocurrencies claim to provide anonymous payments. It is hard to evaluate these claims. We use past work in anonymity to select an anonymity metric for evaluating Monero and Zcash. We evaluate this metric's performance using existing attacks on Monero and Zcash. We then evaluate this metric on a new attack and use it to suggest ways to circumvent this attack.**

I. INTRODUCTION

As people become more and more careful about their privacy, anonymity plays a more and more essential role in today's transactions. Among the various types of communications, anonymity among cryptocurrencies has been a hot topic of late. Various implementations have emerged trying to provide privacy for cryptocurrencies. This paper chooses two main cryptocurrencies to analyze –Monero and Zcash. Zcash is more like an upgrade of Bitcoin, providing zero-knowledge proofs to protect privacy, while Monero uses ring signatures and hidden addresses to separately guarantee transaction validity and privacy.

Given their different natures, this paper uses various attacks to evaluate their anonymity. In Monero, a low number of mixins leads to an effective zero mixin attack that can reveal a vast majority of spent outputs. In previous versions of Monero, a temporal analysis could determine the true input being spent in a transaction. A final Monero attack relied on the merging of outputs to determine the true inputs being spent in a transaction. In Zcash, one attack can link transactions that go through the hassle of anonymizing themselves.

But then an intuitive question comes, what is the metric to measure the performance of an implementation's anonymity? Reiter and Rubin proposed to use the probability the attacker sent to the potential sender to calculate anonymity to calculate, while Berrthold at al simple use log2(N) to compute anonymity. Since these methods ignore the relationship between users, later research advised to use entropy to analyze anonymity. This paper will analyze different anonymous metrics and choose one to calculate Zcashs and Moneros anonymity.

II. BACKGROUND

A. Anonymity Metrics

Many metrics have been proposed to measure the anonymity of systems. When it comes to cryptocurrencies, it is useful to measure the ability of an attacker to track payments through a blockchain. In the case of Monero, one useful attack to identify is the ability of an attacker to identify when an input has been spent. Ideally, all mixins for an input in a transaction are equally likely to be the real input. In the case of Z-Cash, the attack of interest is the ability of an attacker to associate shielding and deshielding JoinSplit transactions. In other words, an attacker can track payments in Zcash if she can identify the deshielding operation (and therefore the corresponding receiving transparent address) that transfers the value shielded by a certain shielding transaction.

In the case of cryptocurrencies, we would like our metric to consider both the number of potential association targets and their corresponding probability distributions. This will allow us to consider both external attacker knowledge (e.g. attacker controlled inputs) and knowledge that is internal to the specific blockchain. As such, we can consider three different anonymity metrics.

1) SD Metric: The first anonymity metric to be considered is the effective anonymity set size presented by Serjantov-Danezis[5]. Given an anonymity set $A = x_i$ with a probability distribution s.t. p_i is the probability that x_i is the *real* member of the set, the effective size S of A is

$$S = -\sum_{x_i \in A} p_i \log_2(p_i)$$

If $|A| = N$, then the maximum possible value for S is $-log_2(1/A) = log_2(A)$. The difficulty in using this metric related to using it as a method of comparison between anonymity sets of different size.

2) Degree of Anonymity: The second anonymity metric we consider is the degree of anonymity as presented by *Diaz et al.*[6]. Given an anonymity set $A = x_i$ with a probability distribution s.t. p_i is the probability that x_i is the *real* member of the set, the degree of anonymity of A is

$$d(A) = \frac{-\sum_{x_i \in A} p_i \log_2(p_i)}{|A|}$$

While this metric is easier to use when comparing anonymity sets of different sizes, it doesn't account for a change in anonymity set size. For the purpose of considering external circumstances to the network, we would like to consider the size of the anonymity set in our metric.

3) Scaled Anonymity Set Size: Our last anonymity metric is a sort of combination of the above two metrics. The Scaled Anonymity Set Size (SASS) is defined by Andersson[7]. As a metric, it scales linearly with anonymity set size, providing an easy and intuitive understanding of anonymity. Given an anonymity set $A = x_i$ with a probability distribution s.t. p_i is the probability that x_i is the *real* member of the set, the SASS of A is defined as:

$$H(A) = -\sum_{x_i \in A} p_i \log_2 (p_i)$$

$$SASS(A) = 2^{H(A)}$$

We will use this anonymity metric for our analyses.

B. Monero

A Monero transaction consists of inputs and outputs. The inputs are the source of value that gets sent in a transaction. This value is retrieved in outputs that are generated by the transaction. The input being spent in a transaction can be hidden by using mixins. The outputs that are receiving the values can also be obfuscated. Each output created in a transaction can serve as an input to a future transaction or a mixin for said input. Using a key ring, Monero can verify ownership of outputs to be used in a transaction without revealing the keys of the sender.

The anonymity of Monero should guarantee the satisfaction of these two properties[4]:
1. **Unlinkability**: for any two transactions, it should be impossible to deduce that they were sent to the same recipient.
2. **Untraceability**: Given a transaction input, the real output being redeemed in it should be anonymous among a set of outputs.

In order to guarantee these properties, Stealth address has been utilized on the path and ring signatures have been implemented on the sources.

1) ***Stealth Address****:* Since the goal of Monero is to obfuscate both receivers and senders of transactions, Monero users have two private-public key pairs, one for sending and one for receiving. Users are able to use these to generate single use receiving addresses that still allow received XMR to be spent.

2) ***Ring signatures****:* For the sake of providing senders privacy, instead of transactions being signed only by the sender, they're signed by a ring signature. The ring signature ambiguates the true sender of a transaction. In order to increase privacy, the sender uses a one-time key to build the ring. Therefore, for any nodes rather than the sender and receiver, all these sign looks similar, and there seems no way to tell which one is the actual sender. This one-time key is used to validate the ownership of an input being sent to prevent double spending.

3) ***RingCT****:* Since denominations exist in Monero, without any further efforts, the amount of coins exchangerd in each transaction is transparent. This might cause some problems for anonymity. As a result, RingCT was implemented to hide the amount of XMR being sent in a transaction.

C. ZCash

In order to add anonymity to the online transaction, ZCash uses zk-SNARKs to provide a safe transfer situation. Theoretically, zk-SNARKs can encrypt the address data and let all other users (except receiver) unable to decrypt the address information, protecting peoples privacy. However, it achieves at the expense of high computing cost to encrypt. It needs minutes even hours to privately transfer block data from source to destination, which, in many situations, is too long to tolerate. Therefore, the ZCash designers let many of the transaction parts be transparent. Zcash transactions become a game of trade-offs between privacy and cost.

Zcash has two kind of addresses: *t-addr* which is transparent and *z-addr* which is not transparent. Transparent addresses reveal all their transactions and non-transparent addresses do not reveal their transactions.

1) ***JoinSplit****:* *JoinSplit* is a type of Zcash transaction that enables private payments. A JoinSplit can perform three different operations. A *shielding* operation transfers ZEC from a *t-addr* to a *z-addr*. A *deshielding* operation transfers ZEC from a *z-addr* to a *t-addr*. A *shielded* operation transfers coins between two *z-addrs*. The different operations reveal different information.

	Source	Destination
Shielding	transparent	encrypted
De-shielding	encrypted	transparent
Shielded	encrypted	encrypted

2) ***Zero-knowledge proofs****:* Similar to Monero, Zcash needs a way of protecting against double spending. Zcash's solution uses zk-SNARKs to guarantee that encrypted transactions are valid. This paper will not cover those details.

D. Existing Monero Attacks[4]

1) Temporal Attack: In Monero, when a user attempts to send an input to another user, she has the option to choose a number of existing outputs to use as mixins. In early versions of Monero, these outputs were selected according to a uniform distribution over the indices of the outputs (Nth output created has index N-1). However, outputs being spent on Monero are heavily skewed towards the newest outputs. As a result, in 80% of transactions the true output being spent was simply the newest one[2].

The Monero development community has addressed this issue and has solved it to a degree, but their solution is not perfect[2].

2) Output merging attack: In non-RingCT transactions, Monero enforces a requirement that all outputs created adhere to a denomination standard. As a result, sending 1 input worth 1.1 XMR would result in 2 outputs created, worth 1.0 XMR and 0.1 XMR respectively, both to the same user. When that recipient would go to spend their 1.1 XMR, they would, in 1 transaction, send 2 inputs. One input would contain as a mixin the 1.0 XMR output and the other input would contain as a mixin the 0.1 XMR output.

It should be very rare that the two outputs of a transaction appear as the inputs to a later transaction unless they belong to the same user that is spending both. This leak of information led to the output merging attack which was 87% accurate[4].

This attack does not apply to RingCT transactions.

3) Zero mixin attack: Generally, the larger the number of mixins for a transaction, the better the guarantee of unlinkability of transactions. However, a larger number of mixins comes with a larger transaction fee to pay. As a result, in the past, a majority of Monero transactions used a low number of mixins. Before Monero had a minimum number of mixins required for transactions, a majority of transactions used 0 mixins. The outputs used as a source in this kind of transaction could be identified as having been spent, so they had no utility as a mixin for future transactions. This led to a snowball effect that revealed an even larger amount of transparent transactions that used these outputs as mixins[4].

This has been fixed by enforcing minimum mixin requirements for transactions.

E. Existing Z-Cash Attacks

The Zcash attack we will analyze in this paper supposes that JoinSplits are mostly used to perform a *Round Trip Transaction (RTT)*. The respective numbers of shielding and deshielding JoinSplit operations indicates this is plausible. Given a shielding operation that transfers *X* ZEC to a *z-addr*, if we find only 1 deshielding operation with a value of *X* ZEC, we can say that it is likely that these two form a RTT. This is supported by the low frequency of these kinds of pairs.

III. Anonymity Analysis

Given the above attacks, we modeled the current attacks on Monero and Z-Cash anonymity with the SASS metric. The motivation behind this analysis is to see if we can observe a correlation between the strength of the aforementioned attacks and the anonymity of transactions. The presence of this correlation suggests that we may be able to create new attacks on an arbitrary blockchain by targeting reducing the value of the SASS metric.

For Monero, we measured the SASS of the anonymity set representing the mixins of the input of a transaction using the probability distribution of each mixin being the real input for the transaction.

Similarly for Z-Cash, we measured the SASS of the anonymity set representing the potential RTT deshielding partners for a given shielding operation using the probability distribution of the fees used for the RTT.

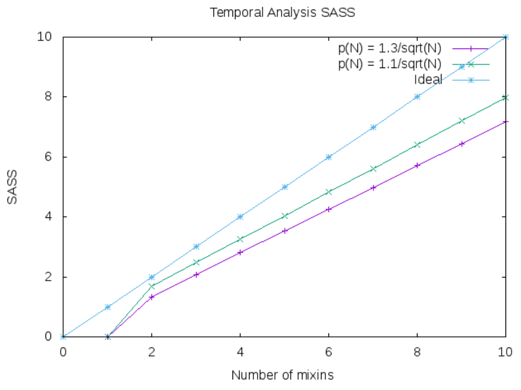
A. Monero Attacks

For each of the attacks on Monero, we used the probability distributions that were relevant when the each of the attacks were either still possible or at its strongest[4]. We did this because all three of these attacks have been addressed by the Monero developers and are either no longer possible or greatly weakened.

1) Temporal Analysis: Simulations suggest that (up to block height 1236195) in 80% of all transactions on the Monero blockchain, the newest mixin for an input is the true input being spent[2]. As the number of mixins in a transaction increases however, the probability that the newest mixin is the true input decays monotonically. However even as N increases, this probability seems to remain significantly higher than $\frac{1}{N}$. For the sake of this paper, we will estimate the accuracy of the temporal analysis to be $acc_t = \frac{some\_constant}{\sqrt{(N)}}$. Using this probability distribution, we can see that for a transaction *tx* with N+1 total mixins:

$$\begin{aligned} p_{N+1} &= acc_t = 1.3/\sqrt{N} \\ H(tx.inputs) &= -[(p_{N+1})\log_2(p_{N+1}) \\ &\quad + (1-p_{N+1})\log_2\frac{1-(p_{N+1})}{N}] \\ SASS(tx.inputs) &= 2^{H(tx.inputs)} \end{aligned}$$

We measured SASS for two different functions for acc_t for $N = 1...10$.



Does this metric work well at evaluating this attack? We know that for increasing values of N, anonymity should increase. It does in this case. We can also see that for higher calculations of acc_t, the anonymity of the system is lower because the metric is more accurate. This matches our intuition about how the parameters of the attack should affect anonymity.

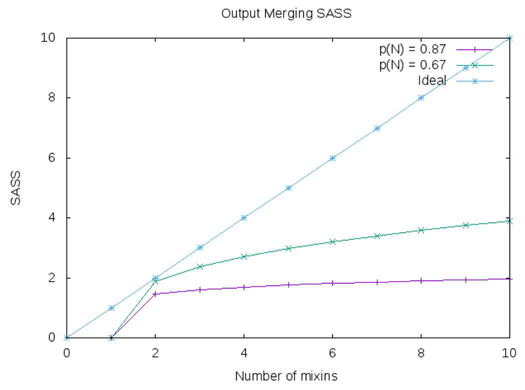
Note that it seems like our decreasing function does not seem to mirror reality [2], but it suffices for the demonstration needed for this paper.

2) Output Merging Analysis: The output merging anonymity attack on Monero had an 87% true positive rate[2]. However, the false positive rate of the attack $FP \approx 0$. Therefore, we can conservatively use 87% as the probability that one mixin is correct and distribute the rest of the probabilty distribution uniformly over the remaining mixins. Therefore, given a transaction with inputs that contain mixins that are both output mixins of a previous transaction, we can guess with 87% probability of being correct the real mixin of each input.

Our calculation in this scenario for SASS is much simpler due to the fact that the data suggests that the accuracy of this attack has no significant impact from the size of the mixin. Rather the weakness of this attack stems from the fact that a very specific scenario is needed. Our SASS calculation is as follows:

$$\begin{aligned} p_{N+1} &= acc_m = 0.87 \\ H(tx.inputs) &= -[(p_{N+1})\log_2(p_{N+1}) \\ &\quad + (1-p_{N+1})\log_2 \frac{1-(p_{N+1})}{N}] \\ SASS(tx.inputs) &= 2^{H(tx.inputs)} \end{aligned}$$

Similar to the temporal analysis, we see a marginal increase in anonymity as we increase the number of mixins. See this in **Figure 2**.



Given the extremely large number of transactions available as mixins, we should expect that the false positive rate increase at an extremely marginal rate as the number of mixins for a transaction increases. The false positive rate should increase only due to rising probability that the two inputs be randomly in the transaction as the number of mixins increases.

We see that this attack is relatively stronger than the temporal analysis attack as N increases, and this makes sense because the accuracy of the attack does not diminish as N increases.

3) Zero mixin Analysis: The zero mixin attack is no longer viable in most scenarios as zero mixin outputs cannot be spent in RingCT. However, we can still apply an anonymity analysis on the attack based on statistics of the past. At a point in time, 88% of all inputs were traceable. Interestingly, the traceability of an input affects the anonymity of all transactions that use that use that input as a mixin. Therefore, if we assume that the choice of a mixin is uniform over all inputs (no longer the case in reality, but true up until version 0.9.0 of Monero), each input had an 88% chance of not contributing to the anonymity of the transaction.

To model this, we estimated the expected value of the anonymity. Consider the following scenario:

We have a transaction tx with 4 total mixins, $\{I_1, I_2, I_3, I_4\}$. Start with the assumption that each is equally likely to be spent. Now we shall consider the existence of a prior transaction tx_0 with 2 total mixins, $\{I_1, I_5\}$. There is an 88% chance that I_5 is a traceable. We would like to not overestimate the probability that I_1 is traceable, so we shall not consider the cases where I_1 is already traced. So in $p = (0.88 - 0.88^2) \approx 0.11$ of cases, we will consider I_1 to be removed from the anonymity set. Therefore,

$$\begin{aligned} \mathbf{E}(H(tx.inputs)) &= (1-p) * (-log_2(0.25)) \\ &\quad + (p) * (-log_2(0.33)) \approx 1.96 \implies \\ \mathbf{E}(SASS(tx.inputs)) &\approx 3.88 \end{aligned}$$

We can see that adding a transaction reduces the anonymity. To consider another transaction $tx_1 = \{I_2, I_6\}$, we just need to do the same thing again.

$$\begin{aligned} \mathbf{E}(H(tx.inputs)) &= (1-p) * (1.96) \\ &\quad + (p) * (-log_2(0.33)) \approx 1.92 \implies \\ \mathbf{E}(SASS(tx.inputs)) &\approx 3.79 \end{aligned}$$

We can see that adding transactions reduces anonymity, as we expect with this attack, since these transactions necessarily give information about whether inputs have already been spent. This is what we expect from our metric.

B. Z-Cash Attack

Recall that Z-Cash has a special kind of masked transaction called a *JoinSplit*. One attack [3] seeks to link shielding and deshielding operations. The attack defines a Round Trip Transaction (RTT) as the following:

Definition 1. Let j_1 and j_2 be JoinSplits included in blocks of height h_1 and h_2 with $h_2 > h_1$. If there exists exactly one pair of transactions (t_1, t_2) with $j_1 \in t_1$, $j_2 \in t_2$, and $vpub\_new(j_2) = vpub\_old(j_1)$, then we say $(t1, t2)$ is a round-trip transaction.(RTT).

Starting with the notion that *JoinSplit* transactions

The methodology for calculating SASS for a certain shielding transaction is as follows:

For the set of fees $F = \{f_i\}$, we have a corresesoding set of frequency distribution $P = \{p_i\}$ over all Z-Cash transactions. Given a shielding transaction tx, find all deshielding transactions d_i s.t. $\exists f_i \in F$ for which $tx.value - f_i = d_i.value$. Call this set $M = \{(f_i, \{d_i\})\}$. Define $P' = \{p_i \in P | (f_i, D) \in M\}$. Normalize P' s.t. $\sum_{p_i \in P'} = 1$.

$$\begin{aligned} H(M) &= -\sum_{(f_i,D)\in M} p_i \log_2 \frac{p_i}{|D|} \implies \\ SASS(M) &= 2^{H(M)} \end{aligned}$$

We can see that if there is only one deshielding transaction, $SASS(M) = 1$, the anonymity is at its minimum. In addition to this, we can see that adding more possible deshielding transactions increases anonymity.We see that the anonymity metric can be used to accurately model this attack.

IV. Monero Input Selection Attack

Now we will use the anonymity analysis to evaluate our attack on Monero's anonymity. Our attack focuses on the presence of an attacker that holds the private key to a significant number of inputs. We show the effectiveness of this attack under different attacker ownership models and under different mixin selection algorithms. We suggest mitigation techniques to avoid this kind of attack, both at an individual level and a protocol level.

A. *Monero Mixin Selection*

The Monero protocol selects the mixins for an input to a transaction from a predefined distribution. It has used three different distributions since the start of Monero. They are the following:

- **Method 1:** Uniform distribution - Output indices selected as mixins are from uniform distribution over all outputs ($I \sim Uniform(0, max\_output\_index)$
- **Method 2:** Triangular distribution - Output indices selected as mixins are from triangular distribution with parameters ($a = 0, b = c = last\_output\_index$)
- **Method 3:** Modified triangular distribution - 50% of output indices selected as mixins are from the above triangular distribution. The other 50% of output indices are selected from a triangular distribution of all outputs from the past 1.8 days with parameters ($a = output\_created\_1.8\_days\_ago, b = c = last\_output\_index$).

B. *Attack*

Our attack takes advantage of these distributions. Information on a transaction can be gained by simply owning one of the inputs being used as a mixin. Motivated by this observation, the goal of our attack is to *sneak* transactions by having outputs that the attacker owns be selected as mixins. Therefore we need to generate outputs in a manner that maximizes the probability of this occurence.

To keep our approach directed, consider an output O generated at block height b_1 and spent at block height b_2. If Method 1 is used to select mixins for the spending transaction, the probability of *sneaking* the spending transaction is independent of how we generate outputs.

Considering Method 2 and Method 3, we observe that the probability of *sneaking* the spending transaction increases with the proximity of the attacker's outputs to b_2 since the newest outputs are most likely to be chosen as mixins. See this probability calculated below.

First we note that the problem of sampling without replacement over Monero's triangular distribution is a Hard problem. This is much easier if we sample with replacement. This does indeed remove some validity from our results.

Assuming N mixins, M attacker generated outputs right before the spending transaction, R other outputs generated in the 1.8 days before the spending transaction, and T total outputs generated in the life of Monero.

The following calculations use the triangular distribution. See [1] for details.

Method 2

$$\begin{aligned} p2_N(sneak) &= 1 \\ &\quad - p_N(choosing\_no\_attackers\_output) \\ &= 1 - (\frac{T^2 - 2TM + M^2}{T^2})^N \end{aligned}$$

Method 3

$$\begin{aligned} p3_N(sneak) &= 1 \\ &\quad - p_N(choosing\_no\_attackers\_output) \\ &= 1 - (\frac{T^2 - 2TM + M^2}{T^2})^{N/2}(\frac{R^2}{(M+R)^2})^{N/2} \end{aligned}$$

This situation represents an optimal scenario for an attacker where b_2 is known. In reality, an attacker cannot guess the value of b_2. In reality, the time between an output's generation and its spending (its *lifespan*) in aggregate follows an estimatable distribution[2]. Note that this is just a generalized case of the above scenario. By targeting the the parts of the output lifespan distribution with the highest probabilities we can optimize the attack under the generalized case.

Note here that for $T >> M$, $\frac{T^2-2TM+M^2}{T^2} \approx 1$. This implies that $p2_N(sneak) \approx 0$. However, $\frac{R^2}{(M+R)^2} \not\approx 1$ so replacing $N/2$ powers of $\frac{T^2-2TM+M^2}{T^2}$ is a significant difference. Since an attacker can get information leaked just by having one of their outputs selected as a mixin, Method 3 leaks information under this attack. An attacker performing the same aforementioned output generating behavior under Method 2 and Method 3 will gain more information when the network is using Method 3.

This is different from the previous attacks mentioned, which got their information leaks from user behavior. This sort of attack shows that we don't need user behavior to get leaked information from Monero.

C. *Prevention*

Note that this attack is not meant to be viable on Monero as is. It is meant as an example of how to get leaked information without analyzing the behavior of users. Indeed, if you keep adding mixins (i.e. increasing N), the heightened probability that a transaction will be sneaked is offset by the increased anonymity provided by having more mixins.

V. CONCLUSIONS

In this paper, we showed that we can use traditional anonymity metrics to evaluate existing attacks against the anonymity of cryptocurrencies. Given that we can do that, we can try to detect information leaks that don't necessarily have to do with user behavior, but we can detect information leaks by abusing parts of the protocol itself. One idea that can come away from this is that cryptocurrencies should not be leaking information unnecessarily. Monero does leak information to mixin owners but it is not clear why it must do so, as it may be a fault in the design of Monero or a necessary evil for maintaining a decentralized platform.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] https://learnandteachstatistics.files.wordpress.com/2013/07/notes-on-triangle-distributions.pdf

[2] Miller, A., Moeser, M., Lee, K., Narayanan, A.: An Empirical Analysis of Linkability in the Monero Blockchain (2017), https://arxiv.org/abs/1704.04299

[3] Quesnelle J.: On the linkability of Zcash transactions (2017), https://arxiv.org/abs/1712.01210

[4] Kumar A. et al: A Traceability Analysis of Monero's Blockchain (2017), https://eprint.iacr.org/2017/338.pdf

[5] Serjantov A., Danezis G.: Towards an Information Theoretic Metric for Anonymity, https://dl.acm.org/citation.cfm?id=1765303

[6] Diaz et al: Towards measuring anonymity, https://www.esat.kuleuven.be/cosic/publications/article-89.pdf

[7] Andersson C., Lundin R.: On the Fundamentals of Anonymity Metrics, https://link.springer.com/chapter/10.1007/978-0-387-79026-8_23