A Comparative Study of Privacy-Preserving Cryptocurrencies: Monero and ZCash

Sofie Christensen

ID 1820778

MSc in Cyber Security

Supervisor: Dr. David Galindo

School of Computer Science

University of Birmingham

September 2018

Abstract

Since it was discovered that Bitcoin offers limited privacy, many cryptocurrencies have emerged implementing privacy-enhancing cryptographic technologies including the use of ring signatures and zk-SNARKs. Two of the most popular cryptocurrencies using these techniques respectively are Monero and Zcash. A study has been conducted to identify the different cryptographic primitives used to achieve privacy in Monero and Zcash and to offer a direct comparison of both. The study shows that the implementation of privacy enhancing cryptographic primitives introduces new limitations to the protocols in terms of usability and efficiency. These limitations are critically assessed and evaluated.

Monero was identified to be vulnerable to two attacks that could compromise privacy. The first attack shows that transactions can be de-anonymised due to the limited number of mixins chosen for the ring signature. The second attack shows that it is possible to correctly guess which mixin the real input is the majority of the time. Based on these vulnerabilities, three improvements are suggested and tested. The first is a different ring signature scheme that allows for constant sized transactions. It was shown that this scheme offers a viable solution to incentivise the use of more mixins. The second is a different sampling algorithm that increases the effective untraceability of the real input. The implementation resulted in better effective untraceability than what is currently achieved by Monero. Finally, a more secure stealth address generation algorithm is presented and implemented. This study critically assesses the advantages and disadvantages of the proposed improvement.

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. David Galindo, for his invaluable guidance and support throughout the completion of this project.

I would also like to thank my parents and my siblings for their constant support.

Finally, I would like to thank Ollie, for once again being my rock throughout the completion of this MSc.

Contents

1. Introduction

The emergence of blockchain has introduced the possibility for currencies to be decentralized. Cryptocurrencies such as Bitcoin ensure that the entire network can control and verify the ledger on a consensus basis. This means that finances can be seen and monitored by everyone, resulting in the problem that the privacy of users is not preserved. Bitcoin attempts to mitigate this issue by introducing pseudonymous addresses, where the address is not directly linked to the owner of the funds. However, it has been shown that Bitcoin transactions can be linked back to their users and that Bitcoin thus offers limited privacy (Conti, et al., 2018). As a response, new cryptocurrencies have emerged that present a solution to the privacy problem. Among the most popular of these cryptocurrencies are Monero and Zcash, who use different cryptographic primitives to preserve privacy.

In cryptographic applications it is often necessary to consider the trade-offs between privacy, efficiency and usability. The same is true for applications such as Monero and Zcash. Both cryptocurrencies employ different cryptographic primitives to promote privacy, with different consequences for efficiency and usability. In this study the Monero and Zcash protocols will be compared and their advantages and disadvantages will be analysed. In addition to a direct comparison, improvements to the Monero protocol will be suggested and tested based on the limitations identified.

The dissertation is organised as follows. First, related works are presented in the literature review. Then the cryptographic primitives used in Monero and Zcash are described in the background chapter, followed by an outline of the Monero and Zcash protocols. Chapter 5 consists of the direct comparison of Monero and Zcash. Finally, chapter 6 presents the suggested improvements to the Monero protocol and testing results.

2. Literature Review

2.1. Monero

The first version of Monero was based on a paper by Van Saberhagen (2013) which introduces a protocol called CryptoNote. CryptoNote uses ring signatures and stealth addresses to ensure privacy of the sender and receiver respectively. Ring signatures were first introduced by Rivest, et al. (2001). The ring signature is a variant of the group signature, which was first introduced by Chaum & Van Heyst (1991). The ring signature used by CryptoNote is a variant of the traceable ring signature based on the work by Fujisaki & Suzuki (2007).

Research has shown that vulnerabilities exist in the CryptoNote protocol, which allow for transactions to be de-anonymised. Noether & Mackenzie (2014) show that CryptoNote can be de-anonymised by tracing back mixins that have already been spent. Miller, et al. (2017) performed a separate analysis that showed that 63% of Monero Cryptonote transactions can be traced and that zero-mixin transactions can be used to de-anonymise further transactions. A similar analysis on the traceability of transactions in Monero was carried out by Kumar, et al. (2017). Wijaya, et al. (2018) completed a study that found that there are two types of attacks that can be executed to reduce the anonymity set in the ring signature.

In response to the studied vulnerabilities, Monero implemented the RingCT protocol. The whitepaper for RingCT is written by Noether & Mackenzie (2016). This protocol differs from CryptoNote in two notable ways. Instead of using traceable ring signatures, RingCT uses Multi-Layered Linkable Spontaneous Anonymous Group signatures, which are a variant of the linkable ring signatures introduced by Liu, et al. (2004). Additionally, they make use of Confidential Transactions, first introduced by Maxwell (2015).

There has been research into creating ring signatures that are of a constant size, i.e. ring signatures that do not increase in size linearly to the number of mixins used in the ring signature. There are a number of papers that describe an algorithm for creating constant size linkable ring signatures, also named short linkable ring signatures. The first introduction to short linkable ring signatures came from Tsang & Wei (2005) who used accumulators as a solution to provide constant sized ring signatures. Au, et al. (2006) discovered a weakness in the previous paper and offered an improvement to their security assumptions. A constant size ring signature with escrowed linkability was introduced by Chow, et al. (2006). Au, et al. (2013) introduced an id-based short linkable ring signature as well. Finally, an improvement of the RingCT protocol used by Monero was introduced by Sun, et al. (2017), who use the same concepts as the ones used in the previous papers.

2.2. Zcash

The technologies used by Zcash are *zero knowledge succinct arguments of knowledge* (zk-SNARKs) which were first introduced by Bitansky, et al. (2012). Miers, et al. (2013) used the concept of non-interactive zero knowledge proofs to introduce a new proposal for an extension of Bitcoin, called Zerocoin. An important limitation of Zerocoin is that the proofs of knowledge are not scalable and thus do not offer a viable solution to the privacy problem that can be implemented in real world applications. In response, Zerocash, which uses zk-SNARKs, was introduced in a paper by Sasson, et al. (2014) and implemented, with some adjustments, to become what is now known as Zcash. Zcash uses a trusted setup that is needed for the zk-SNARKs to work. The trusted setup is implemented by using the multi-party protocol called the Pinnochio protocol as described by Bowe, et al. (2017).

Considering the novelty of zk-SNARKs and the Zcash cryptocurrency there has not been a lot of research into the vulnerabilities that might be present in the protocol. However, studies conducted by Kappos, et al. (2018) and Quesnelle (2017) show that it is possible to distinguish transactions made by miners, founders and "normal" users, highlighting possible traceability issues.

3. Background

3.1. Ring Signatures

Ring signatures were first introduced by Rivest, et al. (2001) in their paper *How to Leak a Secret*. They describe a scenario where a member of a cabinet wants to leak a secret without revealing who he is. In addition, he wants to prove that he is a trustworthy source by showing that he is indeed a member of cabinet. The proposed solution is that the member of cabinet produces a signature that includes his own secret key, as well as the public keys of other members of cabinet. The signature that is produced is called a ring signature and hides the actual signer whilst proving that the signer is a member of the group of signers.

Given a message m, the signer's secret key S_s, and the public keys $P_1, P_2, \dots, P_r$ of all the ring members, the ring signature is computed by the signer as follows (Rivest, et al., 2001).

1. The signer computes the symmetric key k as the hash of the message m:

$$k = h(m)$$

2. The signer picks an initialisation value v uniformly at random from $\{0,1\}^b$

3. The signer picks random x_i for all the other ring members, uniformly and independently from $\{0,1\}^b$ and computes:

$$y_i = g_i(x_i)$$

4. The signer computes the unique value for y_s and solves the following ring equation for y_s:

$$C_{k,v}(y_1, y_2, \dots, y_r) = v$$

5. The signer obtains x_s by inverting g_s on y_s using his knowledge of his trap-door:

$$x_s = g_s^{-1}(y_s)$$

6. The signature on message m is defined as:

$$(P_1, P_2, \dots, P_r; v; x_1, x_2, \dots, x_r)$$

The combination function $C_{k,v}(y_1, y_2, \dots, y_r)$ is a symmetric encryption function E_k applied to the sequence $(y_1, y_2, \dots, y_r)$ as follows:

$$C_{k,v}(y_1, y_2, \dots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\dots \oplus E_k(y_1 \oplus v) \dots))))$$

The resulting ring signature can be seen in Figure 1.

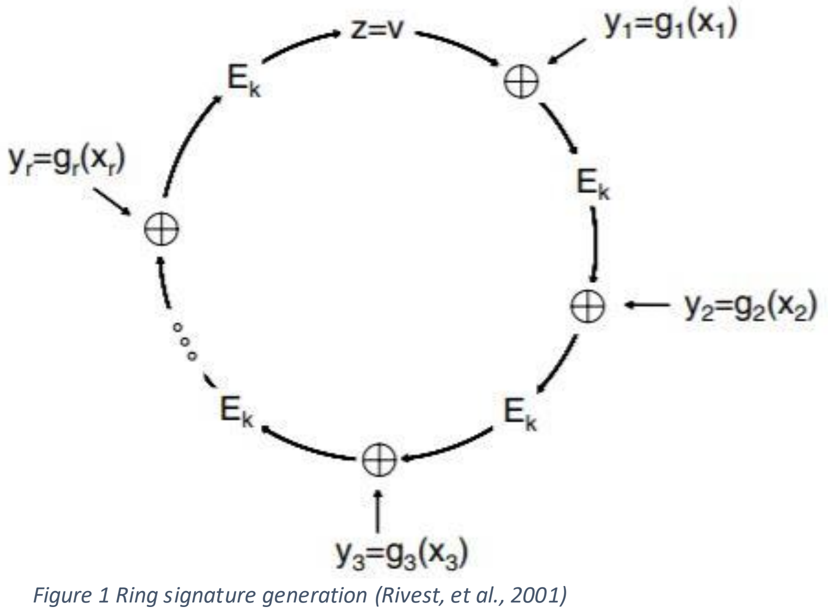

Figure 1 Ring signature generation (Rivest, et al., 2001)

A verifier can verify a signature $(P_1, P_2, \dots, P_r; v; x_1, x_2, \dots, x_r)$ on a message m as follows.

1. For $i = 1,2, \dots, r$ the verifier computes:

$$y_i = g_i(x_i)$$

2. The verifier hashes the message to obtain k:

$$k = h(m)$$

3. The verifier checks that the y_i's satisfy:

$$C_{k,v}(y_1, y_2, \dots, y_r) = v$$

4. If the ring signature is satisfied, the verifier accepts the signature as valid. Otherwise the signature is rejected.

It is important to note that a ring signature is different to a group signature as presented by Chaum & Van Heyst (1991). The main difference is that in the group signature the group of signers is managed by a group manager who distributes keys to the members of the group. Each member of the group can then produce a signature on behalf of the entire group. Crucially, the group manager can also revoke anonymity and reveal the actual signer of the signature. Another distinction to make is that the group of signers all agree to work together and that the group is predefined in the group signature scheme. In contrast, the ring signature can be generated by a signer without the knowledge or consent of the members included in the ring signature. This means that the ring signature can be constructed dynamically, which is a crucial observation for the blockchain.

3.1.1. Linkable Spontaneous Anonymous Group Signature

Monero uses a variant of the ring signature called the Linkable Spontaneous Anonymous Group/Ring Signature (LSAG) first introduced by Liu, et al. (2004). The main difference between the classic ring signature and the LSAG is that the LSAG allows for two signatures to be linked if they are signed by the same private key. This is an important observation with regard to cryptocurrencies as it will allow for double-spending attempts to be detected and rejected.

Similar to the ring signature described in the previous section, each user i has a distinct public key y_i and a private key x_i such that $y_i = g^{x_i}$ for $i = 1, \dots, n$.

Let $H_1: \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $H_2: \{0,1\}^* \rightarrow G$ be some cryptographic hash functions where $G = \langle q \rangle$ is a group of prime order q. Given a message $m \epsilon \{0,1\}^*$, list of public keys $L = \{y_1, \dots, y_n\}$, and private key x_s corresponding to $y_s \in L$ an LSAG signature generation is described as follows (Liu, et al., 2004).

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_s}$

2. Pick $u \in \mathbb{Z}_q$ at random and compute

$$c_{s+1} = H_1(L, \tilde{y}, m, g^u, h^u)$$

3. For $i = s+1, \dots, n, 1, \dots, s-1$ pick $r_i \in \mathbb{Z}_q$ at random and compute

$$c_{i+1} = H_1(L, \tilde{y}, m, g^{r_i} y_i^{c_i}, h^{r_i} \tilde{y}^{c_i})$$

4. Compute $r_s = u - x_s c_s$

The generated signature is $\sigma_L(m) = (c_1, r_1, \dots, r_n, \tilde{y})$.

Given a signature $\sigma_L(m) = (c_1, r_1, \dots, r_n, \tilde{y})$ on a message m and a list of public keys L a signature is verified as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \dots, n$ compute $z_i' = g^{r_i} y_i^{c_i}$, $z_i'' = h^{r_i} \tilde{y}^{c_i}$.

   If $i \neq n$ compute $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$

2. Check whether $c_1 = H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes accept, otherwise reject.

Two signatures $\sigma_L'(m') = (c_1', r_1', \dots, r_n', \tilde{y}')$ and $\sigma_L''(m'') = (c_1'', r_1'', \dots, r_n'', \tilde{y}'')$ are linked if $\tilde{y}' = \tilde{y}''$.

3.2. Pedersen Commitment

Commitment schemes are cryptographic primitives used to commit to a value without revealing the value and crucially, prevent the value from being changed once it has been committed to, making them binding. An important variant of the commitment scheme is the Pedersen Commitment first introduced by Pedersen (1991). The scheme is described as follows.

Let g and h be elements of G_q such that nobody knows $log_g(h)$. The committer commits himself to a value $s \in \mathbb{Z}_q$ by choosing $t \in \mathbb{Z}_q$ at random. The commitment is computed as:

$$C(s, t) = g^s h^t$$

The commitment can later be opened and revealed by revealing s and t.

The special property of this commitment scheme is that it is homomorphic, which means mathematical operations can be performed on the committed values. The idea being that these operations produce a commitment to the result of applying these mathematical operations on the non-committed values.

For example, given a commitment to the value 3, $C(3)$, and a commitment to the value 5, $C(5)$, addition can be performed with the result being a commitment to the value 8, $C(8)$:

$$C(3) + C(5) = C(3 + 5)$$

This property will prove useful in Monero, where transaction amounts are hidden by the Pederson Commitment, but where a sender needs to be able to prove that the input amounts equal the output amounts.

3.3. Zk-SNARKs

The main technology used in Zcash is the zk-SNARK (zero knowledge-Succinct Non-interactive Arguments of Knowledge), a variant of zero knowledge proofs. Classic zero knowledge proofs consist of an interaction between the prover and the verifier resulting in the verifier either accepting or rejecting the final proof provided by the prover. These types of proofs are not suited for blockchain applications such as Zcash because they are highly inefficient.

The term zk-SNARK was first coined by Bitansky, et al. (2012) and is a variant of the Succinct Non-interactive Arguments (SNARGs). A description of zk-SNARKs as they are used in the Zcash protocol is given by Sasson, et al. (2014). For an F-arithmetic circuit C, its relation R_C and its NP language $\mathcal{L}_c$, a zk-SNARK consists of three polynomial-time algorithms $(KeyGen, Prove, Verify)$, which are described as follows (Sasson, et al., 2014).

1. $KeyGen(1^\lambda, C) \rightarrow (pk, vk)$. Given a security parameter λ and an F-arithmetic circuit C, $KeyGen$ probabilistically samples a proving key pk and a verification key vk. The keys are published as public parameters and can be used, any number of times, to prove/verify membership in $\mathcal{L}_c$.

2. $Prove(pk, x, a) \rightarrow \pi$. Given a proving key pk and any $(x, a) \in R_C$, $Prove$ outputs a non-interactive proof π for the statement $x \in \mathcal{L}_c$.

3. $Verify(vk, x, \pi) \rightarrow b$. Given a verification key vk, an input x and a proof π, $Verify$ outputs $b = 1$ if the verifier is convinced that $x \in \mathcal{L}_c$.

4. Protocol Description

This chapter introduces the protocols used in Monero and Zcash, based on the building blocks presented in the previous chapter.

4.1. Monero

As seen previously, Monero provides privacy through Ring Confidential Transactions (RingCT), which were first introduced by Noether & Mackenzie (2016). RingCT combines three cryptographic primitives to achieve privacy. A variant of the LSAG ring signatures is used to obfuscate the actual sender of the transaction, Confidential Transactions are used to hide the amount sent in a transaction and stealth addresses protect the privacy of the receiver of a transaction. In the following these cryptographic primitives will be elaborated on.

4.1.1. Multi-layered Spontaneous Anonymous Group Signatures

Monero uses a variant of LSAG ring signatures called Multi-layered Linkable Spontaneous Anonymous Group (MLSAG) signatures, presented by Noether & Mackenzie (2016). The main difference between the MLSAG and LSAG signatures is that the former produces a ring signature on a set of n key-vectors, instead of a set of n keys. Noether & Mackenzie (2016) describe a key-vector as “[..] a collection $\bar{y} = (y_1, \ldots, y_r)$ of public keys with corresponding private keys $\bar{x} = (x_1, \ldots, x_r)$”.

In Monero the MLSAG is computed as follows (Noether & Mackenzie, 2016). Each signer of the ring containing n members has exactly k keys $\{P_i^j\}_{i=1,\ldots,n}^{j=1,..,k}$. A signature on a message m, is generated by the signer, π, by using the key vectors of n other signers.

1. For the signature of the actual signer, π, define

$$L_\pi^j = \alpha_j G$$

$$R_\pi^j = \alpha_j H(P_\pi^j)$$

where α_j is a random scalar and $j = 1, \ldots, k$ are the indices of the signer’s key-vector.

2. For the remaining signers, excluding the actual signer, π , define

$$c_{\pi+1} = H(m, L_\pi^1, R_\pi^1, \dots, L_\pi^k, R_\pi^k)$$

$$L_{\pi+1}^j \; = \; s_{\pi+1}^j G + c_{\pi+1} P_{\pi+1}^j$$

$$R_{\pi+1}^j \; = \; s_{\pi+1}^j H\left(P_{\pi+1}^j\right) + c_{\pi+1} I_j$$

Where s_i^j is a randomly selected scalar. This is repeated incrementing $i \; mod \; n$ until we arrive at

$$L_{\pi-1}^j \; = \; s_{i-1}^j G + c_{i-1} P_{i-1}^j$$

$$R_{\pi-1}^j \; = \; s_{i-1}^j H\left(P_{i-1}^j\right) + c_{i-1} I_j$$

$$c_\pi = H(m, L_{\pi-1}^1, R_{\pi-1}^1, \dots, L_{\pi-1}^k, R_{\pi-1}^k)$$

3. To complete the ring, the signer needs to solve s_π^j for L_π^j and R_π^j. This is done by setting the randomly chosen scalar $\alpha_j = s_\pi^j + \; c_\pi x_j \; mod \; l$ and computing $s_\pi^j = \alpha_j - c_\pi x_j \; mod \; l$

To prove the correctness of 3. recall that the actual signer's signature is

$$L_\pi^j \; = \; \alpha_j G$$

$$R_\pi^j = \; \alpha_j H(P_\pi^j)$$

Now we set $\alpha_j = s_\pi^j + \; c_\pi x_j \; mod \; l$ and show that the signer's signature is now indistinguishable from the other decoy signatures.

$$L_\pi^j \; = \; s_\pi^j G + c_\pi x_j G$$

$$L_\pi^j \; = \; s_\pi^j G + c_\pi P_\pi^j$$

and

$$R_\pi^j = \; s_\pi^j H\left(P_\pi^j\right) + c_\pi x_j H\left(P_\pi^j\right)$$

$$R_\pi^j = \; s_\pi^j H\left(P_\pi^j\right) + c_\pi I_j$$

4. The final signature is $\sigma = (I_1, \dots, I_k, c_1, s_1^1, \dots, s_1^k, s_2^1 \dots, s_2^k, \dots, s_n^1, .., s_n^k)$

Here $I_{1..k}$ is the key image for the key vector of the actual signer. The key images are computed as

$$I_j = x_j H(P_\pi^j)$$

The key image ensures linkability of the signature and allows for detecting double-spending attempts. When a coin is spent, the key image is added to a lookup table. If it already exists in the lookup table, it means that the coin has already been spent and the transaction is then discarded.

To verify the signature the verifier recomputes L_i^j, R_i^j and c_i, for $i = 1, ..n$ and $j = 1, .., k$.

Finally, the verifier verifies that

$$c_{n+1} = c_1$$

4.1.2. Confidential Transactions

In Monero Confidential Transactions (CT) are used to hide transaction amounts. CT was first introduced by Maxwell (2015) as a possible improvement to privacy in Bitcoin. The main cryptographic primitive used in CT is the Pedersen Commitment as described in section 3.2. The commitment scheme used in CT is computed as

$$C = xG + aH$$

Where x is the randomly chosen secret blinding factor, a is the amount and G and H are generators of an elliptic curve group such that no one knows the discrete log of H with respect to G (and vice versa).

Due to the homomorphic properties of the commitment scheme, anyone on the network can verify that the amounts that went into the transaction are equal to the amounts that came out of the transaction by checking that

$$\sum C(in)_i - \sum C(out)_i = 0$$

However, in Monero this mechanism does not work, because a transaction consists of multiple possible inputs, with only one of them being the real input. In order to verify the above statement, the verifier would have to know which input the real input is, thus defeating the purpose of the ring signature. The solution is to create commitments for the inputs and outputs as follows (Noether & Mackenzie, 2016).

$$C_{in} = x_c G + aH$$

$$C_{out-1} = y_1 G + \ b_1 H$$

$$C_{out-2} = y_2 G + \ b_2 H$$

Such that $x_c = y_1 + y_2 + z, y_i$ are mask values, $z > 0$ and $a = b_1 + b_2$. Here x_c is called the "amount key", which is a special private key known only to the sender. The secret amount key can be derived as follows.

$$C_{in} - \sum_{i=1}^{2} C_{out-1}$$

$$= \ x_c G + aH - y_1 G - \ b_1 H - y_2 G - b_2 H$$

$$= zG$$

The secret amount key is $sk = z$ and the corresponding public key is $pk = zG$. It can be seen that the above commitment becomes a commitment to zero $(C(0) = zG + 0H)$, which proves that the input amounts equal the output amounts.

The ring consisting of all input commitments $C_i, i = 1,..,n$ and corresponding public keys $P_i, i = 1,..n$ including the commitment and public key of the actual signer is created as follows.

$$\{P_1 + C_{1,in} - \sum_j C_{j,out}\, , \ldots, P_n + C_{n,in} - \sum_j C_{j,out}\}$$

This ring can be signed because we know one of the private keys ($z + x'$ where $x'G = P_\pi$, the public key of the signer and $zG = pk$ is the special public amount key).

4.1.3. RingCT

Combining the MLSAG as described in section 4.1.1. and the ring produced by the CT commitment scheme, the RingCT protocol as used in Monero is described as follows (Noether & Mackenzie, 2016).

1. Let $\{(P_\pi^1, C_\pi^1), \ldots, (P_\pi^m, C_\pi^m)\}$ be a collection of addresses/commitments of the real signer with corresponding secret keys $x_j, j = 1,..,m$

2. Find $q + 1$ collections of public keys and commitments $\{(P_i^1, C_i^1), \ldots, (P_i^m, C_i^m)\}, i = 1, .. q + 1$
3. Decide on a set of output addresses $(Q_i, C_{i,out})$ such that $\sum_{j=1}^{m} C_\pi^j - \sum_i C_{i,out}$ is a commitment to zero.
4. Let $R = \{\{(P_1^1, C_1^1), \ldots, (P_1^m, C_1^m), (\sum_j P_1^j + \sum_{j=1}^{m} C_1^j - \sum_i C_{i,out})\}, \ldots,$

   $\{(P_{q+1}^1, C_{q+1}^1), \ldots, (P_{q+1}^m, C_{q+1}^m), (\sum_j P_{q+1}^j + \sum_{j=1}^{m} C_{q+1}^j - \sum_i C_{i,out})\}\}$ be the generalized ring to sign.
5. Compute the MLSAG signature on R

4.1.4. Stealth Adresses

Finally Monero uses stealth addresses to provide privacy to the recipient of a transaction. In Monero each user has two key pairs. The first key pair is the viewing key pair (V, v) and is used to scan the blockchain to search for transactions destined for that user. The second key pair is called the spend key pair (B, b) and is used by a user to spend the coins sent to him. Given an ECC base point G, stealth addresses are constructed as follows.

1. A sender Alice wants to send a transaction to Bob, whose public key is of the form $(V = vG, B = bG)$.
2. Alice generates a random r and computes $R = rG$
3. Alice computes one-time public key $P = H_s(rV)G + B$
4. Alice publishes R, P
5. Given R Bob computes $P' = H_s(vR)G + B$
6. Bob checks whether $P' = P$. If not, the transaction is not meant for him and he continues to scan the ledger.
7. If yes, the transaction is meant for him and Bob computes corresponding one-time private key $x = H_s(vR) + b$ such that $P = xG$
8. Bob can now use x to send the transaction to a new address

Note that Bob can share his private viewing key with someone else, who can then scan the ledger for him. However, without the private spending key they cannot spend the transaction sent to Bob. This is a useful feature for auditing purposes for example.

4.1.5. Earlier versions of Monero

It is important to note that the first version of Monero was different to the protocol described in the previous sections. It was based on the CryptoNote protocol, introduced by Van Saberhagen (2013). The main difference is that the old protocol used a different type of ring signature, called a traceable ring signature (Fujisaki & Suzuki, 2007), and did not incorporate Confidential Transactions. The latter is an important observation because in the earlier version transaction amounts were not hidden. This posed a problem because ring signatures could only be created using inputs of the same amount. Suppose a user wants to send a transaction of an unusual amount, for example 0.8456 XMR. In such a case it would be difficult to find a large number of inputs with the same denomination. Consequently, this would prevent the user from using a large number of mixins in his/her ring signature, thus limiting the anonymity set. In section 5.5 it will be shown that the old version of Monero was vulnerable to deducibility attacks that are made possible by limited anonymity sets.

4.2. Zcash

The Zcash protocol is based on the paper presented by Sasson, et al. (2014). There are different types of addresses in Zcash, namely shielded addresses (z-addr) and transparent addresses (t-addr). Each transaction splits its input values into amounts coming from shielded addresses and amounts coming from transparent addresses. Output values are split into amounts going to shielded addresses and amounts going to transparent addresses. This means that there are different types of transactions between addresses, as visualised in figure 2. A transaction can be:

1. From z-addr to z-addr, where, for example, the input amount is split into 5 ZEC coming from a shielded address and 0 ZEC coming from a transparent address. The output amount would for example be 5 ZEC to a shielded address and 0 ZEC to a transparent address. This means that the transaction is completely shielded.
2. From t-addr to z-addr and vice versa. In this case the input amounts would come from a transparent address but would be sent to a shielded address and vice versa. Thus, there is only partial anonymity.

3. From t-addr to t-addr, where the input and output amounts come from/go to transparent addresses. This provides no anonymity and functions similarly to a normal Bitcoin transaction.

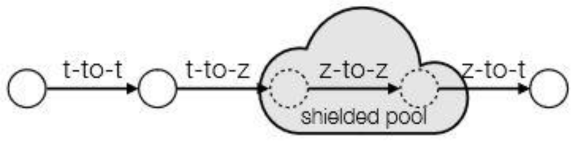

Figure 2 Different types of Zcash transactions (Kappos, et al., 2018)

In Zcash each user generates a random spending key a_{sk} with corresponding incoming viewing key a_{pk} to which incoming transactions can be sent. In addition, a user has a transmission/receiving keypair (pk_{enc}, sk_{enc}). This keypair is used for encrypting sensitive data that needs to be sent over the network.

For the purpose of this study a simplified example of a shielded transaction will be introduced, however a full overview of the Zcash protocol is presented in the protocol specification authored by Hopwood, et al. (2016). In the simplified scenario we have Alice who wants to send a *note* to Bob. A note is a tuple (a_{pk}, v, ρ, rcm), where:

- a_{pk} is Bob's viewing public key
- v is the value of the note
- ρ is the random value chosen to compute the nullifier, nf, using a pseudo-random function. The nullifier can be thought of as a serial number and will be used to prove that the note has not been spent.
- rcm is the random value chosen to compute the note commitment. The note commitment is added to a Merkle tree that holds all notes that were ever created. When the note is spent, this serves to prove that the note exists.

A transaction is made by creating a new note holding the value of the old note (for the sake of simplicity we assume that there is one input note and one output note). Alice creates a new note for Bob in the following way.

1. Choose random rcm^{new}, which will be the blinding value used in the new note commitment.

2. Compute $\rho^{new} = PRF_{\varphi}^{\rho}(h_{sig})$, where PRF_{φ}^{ρ} is a pseudo-random function using the old note's ρ and random seed φ applied to a public seed h_{sig}. ρ^{new} will be used to compute the nullifier of the new note, nf^{new}.
3. Alice sends the nullifier of the old note, nf^{old}, to all nodes. The nullifier is then added to a separate Merkle tree that holds the nullifiers of all spent notes. If the nullifier already exists in that Merkle tree it means the note has already been spent and the transaction will be discarded.
4. Compute new note commitment $cm^{new} = NoteCommit_{rcm}(a_{pk}^{new}, v^{new}, \rho^{new})$ where a_{pk}^{new} is the public key of the receiver, Bob. cm^{new} is sent to all the nodes and will be added to the Merkle tree of existing coins, such that it can be spent in a later transaction.
5. Let n^{new} =$(v^{new}, \rho^{new}, rcm^{new}, memo)$ be the new note.
6. Alice encrypts n^{new} using Bob's transmission key, pk_{enc}, and send it to Bob.
7. For each transaction, an ephemeral signing key pair is generated. Alice signs the transaction using the private signing key. The public signature verification key is added to the transaction.

In addition, Alice generates a zk-SNARK proof, π, proving that she knows:

1. $path$ a Merkle tree path to the note commitment
2. p a position where the note commitment is in the Merkle tree
3. $n^{old} = (v^{old}, \rho^{old}, rcm^{old}, memo)$
4. a_{sk}^{old} Alice's secret spending key
5. n^{new} =$(v^{new}, \rho^{new}, rcm^{new}, memo)$

Such that:

1. The Merkle path is valid
2. The balance of input notes is equal to the balance of output notes
3. $nf^{old} = PRF_{a_{sk}^{old}}^{nf}(\rho^{old})$
4. $a_{pk}^{old} = PRF_{a_{sk}^{old}}^{addr}(0)$

5. $h = PRF^{nf}_{a^{old}_{sk}}(h_{sig})$

6. $\rho^{new} = PRF^{\rho}_{\varphi}(h_{sig})$

7. $cm^{new} = NoteCommitment_{rcm^{new}}(n^{new})$

5. Comparison of Monero and Zcash

The literature review and background have established that Monero and Zcash are predominantly different due to the cryptographic primitives used to ensure privacy. These differences influence the overall performance of the cryptocurrencies in terms of actual privacy provided, efficiency and usability. In the following sections, a comparison of the two cryptocurrencies will be made and their advantages and disadvantages will be critically assessed.

5.1. Anonymity set

The anonymity set contributes to the overall anonymity provided by the cryptocurrency. In theory, the larger the anonymity set, the higher the anonymity. In Monero the anonymity set is only as large as the chosen number of mixins used to construct the ring signature. At the time of writing the enforced minimum mixin number is set to 7, however it is important to note that there has not always been an enforced minimum. In the beginning it was possible to send transactions using zero mixins. It has been found that these transactions can be used to de-anonymise further transactions (Kumar, et al. 2017; Miller, et al. 2017). A zero-mixin transaction *A* is essentially not anonymous as there is only one signer in the ring signature, namely the real signer. This means that when the transaction *A* is spent, anyone on the network can see that the transaction coming from that signer was spent. If this transaction *A* gets used as a decoy mixin in a different transaction *B*, it can be deduced that the transaction *A* cannot be the real input because everyone knows that it has already been spent. This effectively reduces the anonymity set of *B* by one. If transaction *B* was a transaction with only one decoy mixin, *B* would now be de-anonymised and could potentially de-anonymise further transactions. Figure 3 illustrates this type of attack.

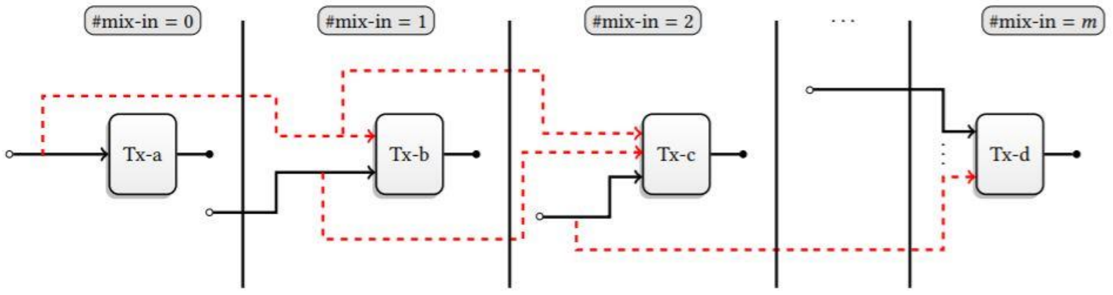

Figure 3 Illustration of the de-anonymisation attack. The dashed lines are inputs identified as decoy mixins, the bold lines are the real input (Kumar, et al., 2017)

Whilst an anonymity set of 7 is very low, it needs to be considered that transactions are sent from one-time addresses as described in section 4.1.4. Even if the real input is found, it would only show which one-time address it came from, not which user is linked to this one-time address.

In Zcash the anonymity set is equal to the entire pool of shielded transactions, which is considerably larger than Monero's anonymity set. However, it has been found that shielded transactions are not used by many users, the consequences of which will be discussed in the next section.

5.2. Opt-in privacy

As seen in section 4.2. Zcash allows for different types of transactions between shielded and transparent addresses. This means that in Zcash it is optional to use the privacy preserving features. It has been found (Kappos, et al., 2018) that only a small percentage of users use shielded transactions. Figure 4 clearly shows that the majority of transactions carried out in Zcash are transparent transactions. Whilst the anonymity set is considerably larger than in Monero, it is not as large as one would expect, due to the small amount of shielded transactions carried out in Zcash. This means that the opt-in privacy feature reduces the anonymity provided by Zcash. The reason for the limited usage of shielded transactions could be due to the fact that Zcash lacks in usability, which will be further discussed in section 5.5.

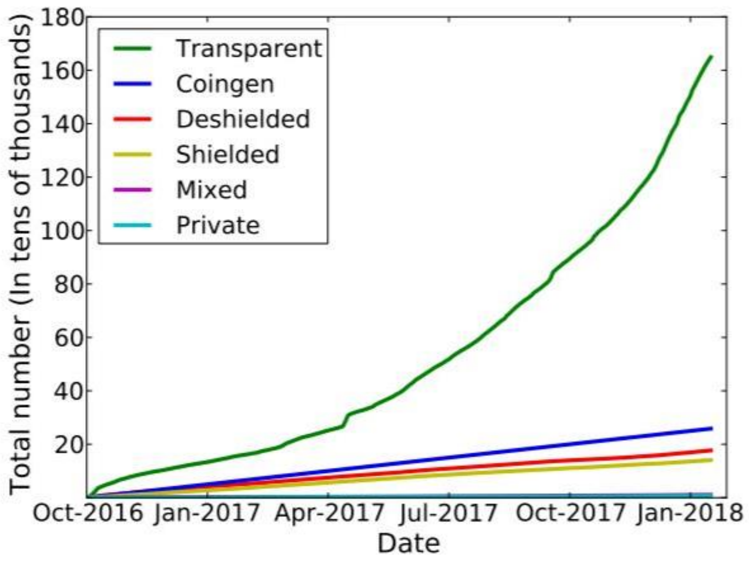

Figure 4 Total number of different types of transactions over time (Kappos, et al., 2018)

In Monero it is mandatory to use RingCT when making a transaction. In addition, the user is required to generate a minimum ring size of 7 in his/her transaction. This means that all transactions are anonymous to the degree of the anonymity set, which is equal to the number of mixins used. It is therefore in the user's best interest to increase this anonymity set by adding more mixins to his/her transactions. However, the majority of users keeps their ring size as minimal as possible. In the last month, as of the time of writing, 96.68% of users generated transactions with 3-9 mixins (Monero Blocks, 2018). One of the main reasons for this is the fact that the transaction fee increases with the expansion of the ring size, keeping users from using more mixins. This in turn leads to a smaller anonymity set, which decreases the privacy provided by Monero.

5.3. Transaction size

In Monero the average transaction size is 12.5 kb, compared to 2 kb in Zcash (Intelligent Trading, 2018). The transaction size in Monero is much larger, because the number of mixins chosen increases the size of the ring signature, which increases the size of the transaction. As Monero instates a minimum mixin of 7, it already considerably expands the transaction size. This is one of the main limitations in Monero because users need to consider the trade-off between increased privacy and lower transaction fees. In contrast, the transaction size in Zcash is much smaller, which means that users can use shielded transactions without having to pay more in transaction fees.

5.4. Trusted setup

The zk-SNARKs used in Zcash require a trusted setup to generate the public parameters that are used to create and verify proofs. This trusted setup was carried out using a multi-party generation protocol, as outlined by Parno, et al. (2013). Six different parties, each holding a part of the master secret, participated in the setup. Each party completed their portion of the protocol by using their part of the master secret, which finally produces the public parameters.

After the parameter generation is finished it is important for each party to discard their fragment of the master secret. If each part does not get destroyed the parties could collude and reassemble the master secret. Whilst this would not allow them to revoke the anonymity of transactions, it would allow them to create new notes out of nothing. It is also important to note that all parts of the master secret are needed for this to happen. This means that even if only one party properly destroyed their part of the secret the reassembly of the master secret would not be possible. This is an important topic of debate within the cryptocurrency community, because a substantial amount of people does not trust that the fragments of the master secret were destroyed. This could also be one of the reasons why adoption of Zcash is so limited. However, it needs to be noted that the Zcash team is working on a new parameter generating protocol, presented by Bowe, et al. (2017), that is scalable enough that it can be generated by hundreds of parties. The idea is that anyone could participate, and it would make it practically impossible for everyone to collude together after the generation is done.

In contrast, Monero does not need a trusted setup, thus the issue is entirely avoided.

5.5. Traceability analysis

Based on the previous sections, it can be said that none of the actual cryptographic primitives used in either Monero or Zcash are susceptible to de-anonymisation. However, there have been several studies into the traceability of both Monero and Zcash transactions. In the following, the findings of these studies will be presented and analysed.

It was found that "62% of transaction inputs with one or more mixins are vulnerable to "chain-reaction" analysis" (Miller, et al., 2017). Recall that the earlier versions of Monero did not enforce a mixin minimum, which meant that 0-mixin transactions could often be used to further de-anonymise other transactions that included these transactions as mixins. The researchers iteratively scanned through the blockchain and marked transactions that were de-anonymised. With each iteration they were able to further de-anonymise other transactions and were eventually able to trace approximately 62% of Monero transactions. However, it needs to be noted that the research was carried out on the first 1288774 blocks (April 15th, 2017) of the blockchain. Monero has since updated to include RingCT, and RingCT transactions can only use other RingCT inputs as mixins. This means that none of the inputs that were de-anonymised by the research poses a threat to the anonymity of current Monero transactions, because they cannot be used as mixins anymore.

A much more relevant problem exposed by Miller, et al. (2017) is that the algorithm used for sampling inputs when creating the ring signature is not representative of the spend-time distribution of Monero transactions. It was found that most users spend their coins within 1.8 days of receiving them. However, Monero samples decoy mixins from a triangular distribution, which means that there is a higher chance that decoy mixins are much older than the real input. The researchers found that "the Guess-Newest heuristic […] applies to 81% of these [deducible] transactions" (Miller, et al., 2017). This means that they were able to correctly guess that the newest input is also the real input 81% of the time.

Similar research was conducted by Kumar, et al. (2017). In addition, they presented a different heuristic. They hypothesise that it is possible to guess the real inputs of a transaction that takes at least two real inputs, if those inputs were both outputs in the same earlier transaction. Figure 5 illustrates this idea. The heuristic assumes that the inputs chosen as decoy mixins are not likely to all come from the same previous transaction. This heuristic is slightly weaker because it allows for false positives.

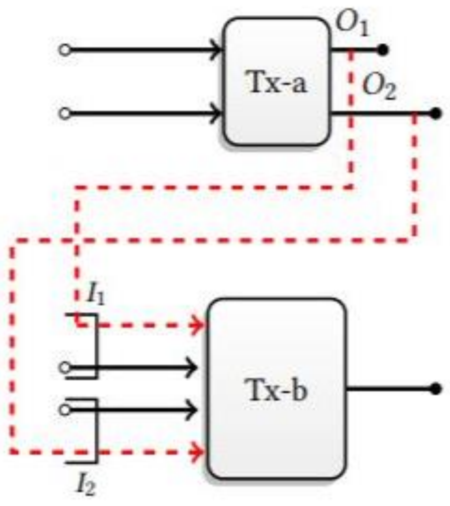

Figure 5 Heuristic proposed by (Kumar, et al., 2017)

Wijaya, et al. (2018) showed that it is also possible for an active attacker to reduce the anonymity of other transactions. The attacker creates a set number of inputs in the "setup phase" and spends them in a transaction. These are inputs that he now knows were spent. This means that, when they get used as decoy mixins in other transactions, he will know that they are not the real input for that specific transaction. The paper describes both a passive and an active attack in this scenario. The passive attack consists of the attacker simply observing the blockchain and finding transactions that use his inputs as decoy mixins. He can then reduce the anonymity set of those transactions by k. Where k is the number of the attacker's inputs used as decoys.

The active attack consists of the attacker compromising a user's wallet. In Monero the wallet is where decoy mixins are picked for the ring signature. An attacker that has compromised a wallet would be able to choose which inputs to use in a transaction. In this case, the attacker would use their own inputs as decoys, effectively reducing the anonymity set to 1, which is the actual input. These attacks are very effective against transactions using small ring sizes. However, when it comes to larger ring sizes, the attacker will have to own an increasingly large number of outputs on the blockchain. Considering the attacker would have to pay a transaction fee for each output he wants to own, this becomes a very expensive attack to carry out.

Zcash has not been around for as long as Monero, and as such there has been less research into the traceability of transactions. Important research was carried out by Kappos, et al. (2018), who, in addition to discovering the limited usage of the Zcash shielded transactions, showed that there are recognisable patterns in the transactions going into and coming out of the shielded pool. Based on these patterns they were able to identify transactions made by founders and miners and able to "reduce the size of the overall anonymity set by 69.1%" (Kappos, et al., 2018). The pattern of founder transactions showed that founders do not often put money into the pool, but when they do they transfer large amounts. It was also found that there was only ever one founder address "active" at a given moment. Once this address reached the limit of 44,272.5 ZEC the next address became the "active" address and the old address did not get used again. They also found that the deposited amount was usually always the same. These heuristics were able to identify founder deposits 74.9% of the time.

The patterns of miners showed that they are the main actors transferring money to the pool. This is not surprising as miners are obliged by the consensus rules to put their coins through the shielded pool when they receive their block rewards. The deposits by miners are therefore easy to spot, because the deposit into the shielded pool occurs directly after a new coin has been mined.

When it comes to withdrawals out of the shielded pool, miners were identified using the following heuristic.

> *If a z-to-t transaction has over 100 output addresses, one of which belongs to a known mining pool, then we label the transaction as a mining withdrawal (associated with that pool) and label all non-pool output t-addresses as belonging to miners.* (Kappos, et al., 2018)

The logic behind this heuristic is that in mining pools the block rewards are shared between the miners. In Zcash this is done by sending the block reward to a shielded address and from there dividing the reward and sending a share to each miner in the mining pool. As a result of the above heuristic, the researchers were able to tag 110,918 addresses as belonging to miners.

The above examples show that the anonymity set can be reduced by identifying miners and founders. In addition, the study presents a heuristic that could link an input address, used in a t-to-z transaction, to an output address, used in a z-to-t transaction. This heuristic is based on work by Quesnelle (2017) and allows to identify “round-trip transactions”. These transactions are used when a user wants to anonymise their coins by sending them to a new address. In this scenario a user sends their coins to the shielded pool and then back to a new transparent address. The heuristic thus claims that a transaction of a specific value v is a “round-trip transaction” if the value is found to have been spent in a t-to-z transaction, followed by the same value being spent in a z-to-t transaction shortly after. Running the heuristic resulted in linking 28.5% of t-to-z transactions, followed by z-to-t transactions. However, they also found that 87% of those transactions were transactions they had identified as being either founder or miner transactions, meaning that the heuristic is less effective when applied to “normal” user transactions.

In summary the research shows that the anonymity set of shielded transactions can be reduced by applying certain heuristics. However, it is important to note that the anonymity set is still larger than in Monero, and that transactions carried out by “normal” users have not been proven to be traceable as of yet. In addition, fully shielded transactions (z-to-z) were not affected by any of the heuristics described.

In conclusion, it can be said that there are distinct differences in the way the cryptographic primitives used in Monero and Zcash affect the cryptocurrencies. In terms of pure privacy guarantees I believe Zcash offers a more secure solution. This is because the use of zero-knowledge proofs means that no sensitive information is published when a user makes a transaction. Consequently, post-hoc de-anonymisation is much less likely and is not likely to reveal any identities or identifying data. As seen in Monero the majority of non-RingCT transactions have already fallen victim to post-hoc de-anonymisation. It is important to note that Zcash was deployed much later than Monero. This means that it has been subject to less "real world" testing and, due to the limited use of shielded transactions, it has not been tested when the network is busy. In contrast, the protocol used in Zcash is based on and backed by highly regarded academic research.

The limited number of people who actually use shielded transactions could be an indication of the fact that Zcash lacks in usability compared to Monero. The reason being that verification times are slow and that the protocol is not very scalable. Monero has better usability but loses some of its privacy guarantees in the process. The larger the ring signatures become, the less usable Monero becomes, which means Monero has to make a compensation between privacy and usability. At the moment the trade-off between privacy and usability is much more balanced than in Zcash. However, Zcash is working on improvements and is releasing an update this month that would improve usability and scalability, possibly pushing Zcash to the forefront of privacy preserving cryptocurrencies.

6. Improvement on Monero

The research done into the traceability of Monero, as described in section 5.5, has highlighted two issues. Firstly, it is important that ring sizes are big enough so that anonymity does not get compromised even if it is possible to reduce the anonymity set by process of elimination. Secondly, the sampling algorithm needs to be more representative of the spend-time distribution, so that an attacker cannot accurately guess which input the real input is. In this chapter improvements to both issues will be proposed and tested. In addition, a different more secure stealth address will be presented.

6.1. Short Linkable Ring Signatures

One of the issues previously discussed is that the ring signatures in Monero grow linearly with the number of mixins used, which means that the transaction fee grows with it. Consequently, less people are inclined to use a larger number of mixins, because they would have to pay higher transaction fees. However, using less mixins reduces the privacy guarantees offered by Monero. A solution to this problem would be to introduce ring signatures that do not grow with the number of mixins used. In the following, the short linkable ring signature will be presented, showing that it is possible to produce constant sized transactions independent of the number of mixins chosen.

The short linkable ring signature schemes presented by Au, et al. (2006, 2013), Chow, et al. (2006) and Sun, et al. (2017) make use of accumulators with one-way domain. The idea of an accumulator is that it takes a set of values and accumulates them into one value by iteratively applying the same function to each element of the set of values. The properties of an accumulator with one-way domain are the following (Sun, et al., 2017).

- Quasi-commutativity. It holds that $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. For any $X = \{x_1, x_2, .., x_n\} \subset X_\lambda$ the accumulated value of X over u is $f(\ldots f(u, x_1) \ldots x_n)$.
- Collision-resistance
- One-way domain
- Efficient generation
- Efficient evaluation

The algorithms of the accumulator with one-way domain are described by Sun, et al. (2017) as follows.

1. $ACC.Gen(1^\lambda)$: generate cyclic groups $G_1 = \langle g_0 \rangle$ and G_2 of prime order p, equipped with a bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$, and an accumulating function $g \circ f: \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow G_1$, where f is defined as $f : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ such that $f : (u, x) \rightarrow u(x + \alpha)$ for some auxiliary information α randomly chosen from $\mathbb{Z}_p^*$(for simplicity u is always set as the identity element of $\mathbb{Z}_p^*$ and g is defined as $g : \mathbb{Z}_p^* \rightarrow G_1$ such that $g : x \rightarrow g_0^x$. The domain of accumulatable elements is $G_q = \langle h \rangle$ which is a cyclic group of prime order q such that $G_q \subset \mathbb{Z}_p^*$. At last, output the description $desc = (G_1, G_2, G_q, e, g_0, g_0^\alpha, g_0^{\alpha^2}, \ldots, g_0^{\alpha^n}, g \circ f)$, where n is the maximum number of elements to be accumulated.
2. $ACC.Eval(desc, X)$: compute the accumulated value $g \circ f(1, X)$ for X by evaluating $\prod_{i=0}^n (g_0^{\alpha^i})^{u_i}$ with public information $\{g_0^{\alpha^i}\}_{i \in [n]}$, where u_i is the coefficient of the polynomial $\prod_{x \in X}(x + \alpha) = \prod_{i=0}^n (u_i \alpha^i)$.

3. $ACC.Wit(desc, x_s, X)$: the relation Ω w.r.t. this accumulator is defined as $\Omega(w, x, v) = 1\ iff\ e(w, g_0^x, g_0^\alpha) = e(v, g_0)$, a witness w_s for the element $x_s \in X := \{x_1, x_2, \ldots, x_n\}$ s.t. $s \in n$ is computed as $w_s = \ g \circ f(1, X \backslash \{x_s\}) = \prod_{i=0}^{n-1} (g_0^{\alpha^i})^{u_i}$ with public information $\{g_0^{\alpha^i}\}_{i \in [n-1]}$, where u_i is the coefficient of the polynomial $\prod_{i=1, i \neq s}^{n} (x_i + \alpha) = \prod_{i=0}^{n-1} (u_i \alpha^i)$.

Simply put, the actual signer of the ring signature wants to prove that his signature is accumulated in the value $g \circ f(1, X)$, which will be called v for simplicity. This is done by generating a new value w which is the accumulated value of all signatures except for the signature of the real signer. The signer can now prove that his signature is accumulated in v by applying the accumulator function to w and his own signature. The result of which should be equal to v if his signature is indeed accumulated in v.

A new protocol for Monero based on the short linkable ring signatures was presented in a paper by Sun, et a. (2017), which they titled *RingCT2.0*. The idea is to use an accumulator to create the ring signature and then generate a proof that the user's real input is accumulated in that value, as well as prove that the user has the private spend key to allow him to spend the coin. The result is a ring signature of size $O(1)$, instead of $O(n)$, where n is the number of signers, which is the case in the ring signature scheme used by Monero. This solution would therefore incentivise users to use more mixins because it would no longer result in higher transaction fees.

One drawback of the short linkable ring signature is that it requires a trusted setup. As discussed in the case of Zcash trusted setups can be controversial. However, with the emergence of new parameter generation protocols that allow a much larger number of participants, the trusted setup would be less contested.

Given that the main privacy problem in Monero is the small anonymity set, the solution proposed by Sun, et al. (2017) is an important one. Therefore, it would be interesting to implement the solution and thoroughly test it in further work.

6.2. Mixin Sampling Strategy

As discussed in section 5.5 it was found that the sampling strategy used by Monero is not representative of the actual spend-time distribution of transactions (Miller, et al., 2017). Temporal analysis shows that most often the most recent input is the real input as well. In Monero most transactions are usually spent within 1.8 days of receiving the transaction, whilst the Monero distribution samples mixins from a triangular distribution. Therefore, it is important to find a different sampling algorithm that represents the spend-time distribution more closely.

Miller, et al. (2017) present a different sampling strategy that samples data using a model that was constructed using blockchain data. Their idea was to estimate the spend-time distribution and use this distribution to sample mixins.

In the following a different sampling distribution is presented as a possible solution to the sampling problem. The strategy is given as follows:

> *For a ring size of n:*
>
> *Pick n/4 inputs from the blocks 10-25 at the top of the chain, i.e. the newest blocks that could contain freshly unlocked balances.*
>
> *Pick n/4 inputs biased towards recent inputs, similar to the current input selection.*
>
> *Pick n/4 inputs biased towards old inputs, with older ones being more likely.*
>
> *Pick n/4 inputs uniformly across all possible choices.*
>
> *From all the chosen inputs, replace the input that is closest to the real one with the real input.* (Distinctive Multiple, 2018)

The idea is that this sampling strategy would account for different types of transactions whilst providing a uniform mixin selection. Given the most common scenario, where a user spends his/her transaction within 1.8 days, it would be difficult to guess which input the real input is given inputs from the above sampling strategy. The "Guess Newest" heuristic would not be effective, given that there will usually be a more recent input, namely one from the blocks 10-25 at the top of the chain.

This sampling strategy also accounts for more unique cases, for example where a transaction is spent after a much longer time. If samples were only sampled from within 1.8 days, an older transaction would stand out amongst the other inputs sampled from the spend-time distribution. In this sampling algorithm, that problem is avoided because there will always be at least one more old input in the transaction.

In addition, the sampling algorithm ensures that no information can be deduced simply by looking at the mixin samples. Consider the case where there are four decoy mixins, each chosen from the four different distributions. Now if the actual input were simply added to the mixin list, there would be two inputs closely related to the same sampling distribution. This means that an onlooker can now deduce that one of those two must be the real input. In the proposed sampling strategy one of the decoy mixins is replaced by the actual input which mitigates against this type of analysis.

The above sampling strategy was implemented and can be found on the github repository, as described in the Appendix. In order to test the algorithm, the test performed by Miller, et al. (2017) for their sampling strategy was reproduced. It consisted of running the algorithm 3500 times, for ring sizes of 2-16 (1-15 decoy mixins). Then the probability of each mixin being the real mixin was calculated. The code for which (*effective_untraceability.py)* can also be found in the github repository. Using the resulting probabilities, the guessing entropy Ge was calculated as follows (Miller, et al., 2017):

$$Ge = \sum i\, p_i$$

Where p_i is the set of probabilities, sorted from highest probability to lowest.

Finally, using the guessing entropy, the effective untraceability was calculated as follows:

$$Effective\ Untraceability = 1 + 2Ge$$

The effective untraceability results can be found in Figure 6. The results for the sampling algorithm used by Monero, the sampling algorithm presented by Miller, et al. (2017) and the ideal sampling algorithm were taken from the work by Miller, et. al (2017). The new suggested sampling strategy has an overall higher effective untraceability than the sampling algorithm in Monero. However, it performs slightly worse than the suggested implementation from Miller, et al. (2017). It can be reasoned that the results of the latter are better because the sampling strategy described is fitted to the spend-time distribution of Monero. This means that the sampling algorithm is much more representative of the actual spending behaviour in Monero. In the ideal case each mixin would have an equal probability of being the real mixin. However, this is a difficult feature to achieve because of this spend-time distribution.

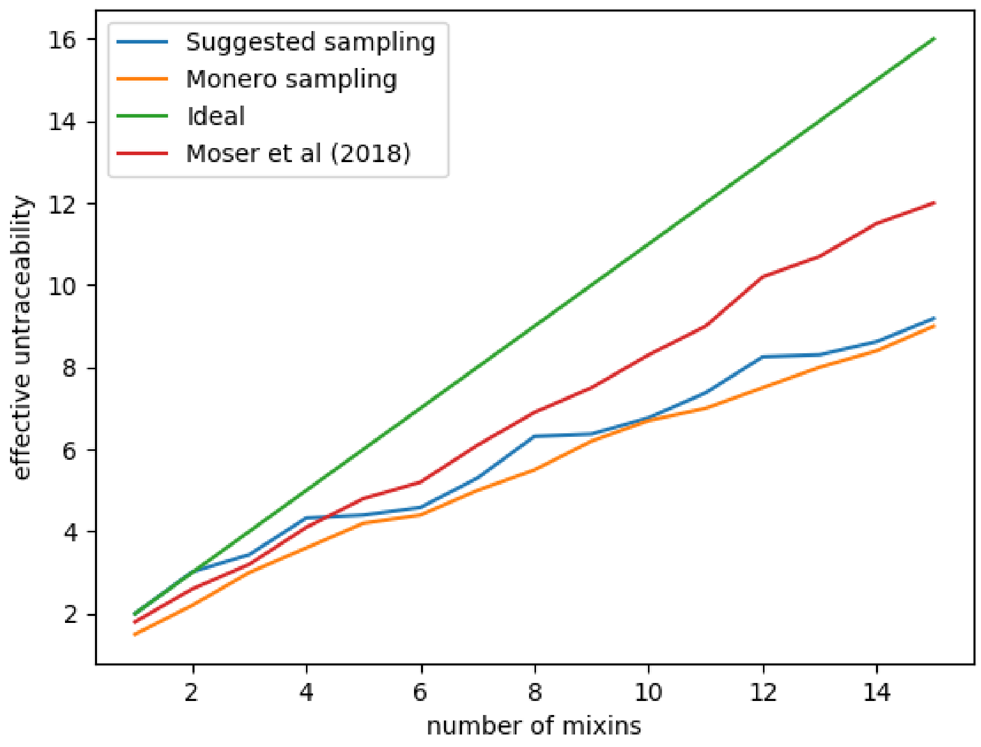

Figure 6 Effective untraceability of each sampling algorithm

6.3. Stealth Address Security Improvement

The stealth address used in Monero is vulnerable to a bad random attack (Courtois & Mercer, 2017). The attack is enabled if a user uses two identical random numbers when generating an ECDSA signature. The ECDSA signature algorithm to sign a message m is defined as:

1. Choose k randomly in $\{1, \dots, q-1\}$
2. Let $T = kP$
3. Let $r = f(T)$. If $r = 0$ start again
4. Let $e = H(m)$
5. Let $s = (e + xr)/k \bmod q$. If $s = 0$ start again
6. Return (r, s)

The bad random attack allows for an attacker to recover secret key x if the same random k is chosen for two different signatures. The attack works as follows:

1. Set $k = (e + xr)/s$ and $k' = (e' + xr')/s'$

2. Since $k = k'$ we can set $\frac{e+xr}{s} = \frac{e\prime+xr\prime}{s\prime}$

3. Now $x = \frac{(e's-es')}{(rs'-r's)}$

Since the one-time secret key x has been recovered, it can now be used to find the secret spending key b of the receiver. Recall, the one-time secret key is defined as follows:

$$x = H_s(vR) + b$$

Consequently b can now be found by setting $b = H_s(vR) - x$. An attacker can now spend all coins sent to the receiver's public address. Note that for this attack to work, the attacker would also have to have knowledge of the secret viewing key v.

The solution proposed by Courtois & Mercer (2017) suggests that each user has one view key pair (V, v) as in the current version of Monero, but several spend key pairs (B_i, b_i). The new stealth address protocol is laid out as follows:

1. Sender chooses random r and sets $R = rG$, where G is the base point of the elliptic curve.
2. Sender computes one-time public key $P = H(rV)G + \sum H(rV)B_i$ where V is the public viewing key of the receiver and B_i are the public spending keys of the receiver.
3. Sender publishes R and P on the blockchain
4. Receiver scans the blockchain and computes $P' = H(vR)G + \sum H(vR)B_i$ where v is the private view key of the receiver.
5. Receiver checks whether $P' = P$.
6. If yes, the receiver computes corresponding secret key $x = H(vR) + \sum H(vR)b_i$

The new stealth address generation algorithm was implemented and instructions on running the code can be found in the Appendix.

The new one-time secret key is more robust than the one used in Monero, because an attacker would have to recover all b_i to be able to spend the funds of the receiver. Even if an attacker were able to recover some parts of the secret spending key, he would have to solve a linear set of equations to find the spending key, which is considered a hard problem.

Whilst this solution is effective in protecting the secret spend key against a bad random attack it introduces a scalability issue. The problem being that the address size would grow linearly with the number of spend key pairs that are included. The size of addresses in Monero is already relatively large compared to other cryptocurrencies. This means that a trade-off needs to be considered between security and scalability.

The above attack only works if the attacker has knowledge of the secret viewing key v. Therefore, the above solution is not recommended to users who do not need to share their secret viewing key as their security is much less likely to be compromised with this type of attack. However, consider the case of a company that needs to be audited regularly. The company would have to share their secret viewing key, exposing themselves to the possibility of the above attack. In this case it would be recommended to construct the stealth addresses as described above.

7. Conclusion

The Monero and Zcash protocols show that there are different ways to achieve privacy. They are also a clear example of how cryptographic applications must achieve a balance between privacy, usability, and efficiency. In Monero the ring signatures provide a user-friendly way of achieving privacy. However, the privacy assumptions are less strong than in Zcash, which in turn suffers from less usability.

It has been shown that Monero's privacy heavily depends on the number of mixins used in its ring signatures. The current version prevents users from increasing their ring signature size because of the growing transaction fees. Short linkable ring signatures were presented to solve this problem and it was argued that they offer a good improvement because they decrease the signature size from $O(n)$ to $O(1)$. In further work it would be important to implement the short linkable ring signature and test it to fully assess the importance of the improvement.

Furthermore, an additional problem in Monero was identified. The fact that mixins are sampled from a distribution not representative of the spend-time distribution allows an outsider to correctly guess which input the real input is the majority of the cases. A different sampling algorithm was implemented and tested, showing that its effective untraceability was better than in the current version of Monero. In addition, it was seen that the solution proposed by Miller, et al. (2017) offered even better effective untraceability. Therefore, an extension of this study could look further into their sampling solution and attempt to improve it to further ameliorate the sampling strategy.

By implementing a more secure stealth address generation scheme, it was shown that the new scheme effectively keeps an attacker from retrieving a user's secret spending key, even in the event of a bad random attack. It was argued that the new scheme is useful in cases where a user has to share their secret viewing key, as this makes them more vulnerable to the attack. However, the suggested improvement also results in increased address sizes, which is a consequence an average user, who does not need to share her/his viewing key, may not want to incur. In further work it would be interesting to research whether there is a way of keeping the addresses constant sized, whilst incorporating the new scheme.

References

Au, M.H., Chow, S.S., Susilo, W. and Tsang, P.P., 2006, June. Short linkable ring signatures revisited. In *European Public Key Infrastructure Workshop* (pp. 101-115). Springer, Berlin, Heidelberg.

Au, M.H., Liu, J.K., Susilo, W. and Yuen, T.H., 2013. Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theoretical Computer Science*, *469*, pp.1-14.

Bitansky, N., Canetti, R., Chiesa, A. and Tromer, E., 2012, January. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (pp. 326-349). ACM.

Bowe, S., Gabizon, A. and Green, M.D., 2017. *A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK*. TR 2017/602, IACR.

Bowe, S., Gabizon, A. and Miers, I., 2017. *Scalable multiparty computation for zk-SNARK parameters in the random beacon model*. Cryptology ePrint Archive, Report 2017/1050.

Chaum, D. and Van Heyst, E., 1991, April. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques* (pp. 257-265). Springer, Berlin, Heidelberg.

Chow, S.S., Susilo, W. and Yuen, T.H., 2006. Escrowed linkability of ring signatures and its applications. In *Progress in Cryptology-VIETCRYPT 2006* (pp. 175-192). Springer, Berlin, Heidelberg.

Conti, M., Kumar, S., Lal, C. and Ruj, S., 2018. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*.

Courtois, N.T. and Mercer, R., 2017. Stealth Address and Key Management Techniques in Blockchain Systems. In *ICISSP*(pp. 559-566).

Distinctive Multiple, 2018. “Multidistributed input selection for large ring sizes” https://github.com/monero-project/monero/issues/3112 [accessed 30 august 2018]

Fujisaki, E. and Suzuki, K., 2007, April. Traceable ring signature. In *International Workshop on Public Key Cryptography* (pp. 181-200). Springer, Berlin, Heidelberg.

Hopwood, D., Bowe, S., Hornby, T. and Wilcox, N., 2016. *Zcash protocol specification*. Technical report, 2016–1.10. Zerocoin Electric Coin Company.

Intelligent Trading, 2018. *Intelligent Trading.* [Online]
Available at: https://intelligenttrading.org/wp-content/uploads/ITF-Privacy-Coins-Report.pdf [Accessed 26 08 2018].

Kappos, G., Yousaf, H., Maller, M. and Meiklejohn, S., 2018. An Empirical Analysis of Anonymity in Zcash. *arXiv preprint arXiv:1805.03180*.

Kumar, A., Fischer, C., Tople, S. and Saxena, P., 2017, September. A traceability analysis of Monero’s blockchain. In *European Symposium on Research in Computer Security* (pp. 153-173). Springer, Cham.

Liu, J.K., Wei, V.K. and Wong, D.S., 2004, July. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy* (pp. 325-335). Springer, Berlin, Heidelberg.

Maxwell, G., 2015. Confidential transactions. *URL: https://people.xiph.org/~ greg/confidential_values.txt (Accessed 09/05/2016)*.

Miers, I., Garman, C., Green, M. and Rubin, A.D., 2013, May. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on* (pp. 397-411). IEEE.

Miller, A., Möser, M., Lee, K. and Narayanan, A., 2017. An empirical analysis of linkability in the Monero blockchain. *arXiv preprint*, *1704*.

Monero Blocks, 2018. “Mixins used in transactions (%)”, https://moneroblocks.info/stats/ring-size [accessed 01 Sept 2018]

Noether, S. and Mackenzie, A., 2014. A note on chain reactions in traceability in cryptonote 2.0. *Research Bulletin MRL-0001. Monero Research Lab*, *1*, pp.1-8.

Noether, S. and Mackenzie, A., 2016. Ring confidential transactions. *Ledger*, *1*, pp.1-18.

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

Pedersen, T.P., 1991, August. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference* (pp. 129-140). Springer, Berlin, Heidelberg.

Quesnelle, J., 2017. On the linkability of Zcash transactions. *arXiv preprint arXiv:1712.01210*.

Rivest, R.L., Shamir, A. and Tauman, Y., 2001, December. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*(pp. 552-565). Springer, Berlin, Heidelberg.

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

Sun, S.F., Au, M.H., Liu, J.K. and Yuen, T.H., 2017, September. RingCT 2.0: a compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *European Symposium on Research in Computer Security* (pp. 456-474). Springer, Cham.

Tsang, P.P. and Wei, V.K., 2005, April. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*(pp. 48-60). Springer, Berlin, Heidelberg.

Van Saberhagen, N., 2013. CryptoNote v 2.0.

Wijaya, D.A., Liu, J., Steinfeld, R. and Liu, D., Monero Ring Attack: Recreating Zero Mixin Transaction Effect.

Appendix

The project was uploaded on the gitlab repository. To download the project please run

```
git clone https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2017/sxc1058
```

The repository consists of two folders. *Sampling Algorithm* holds the code for the implementation and testing of the sampling algorithm. *Stealth Address* holds the code for the implementation of the new stealth address.

1. Sampling algorithm

Tested with Python3.6 on Ubuntu 17.10

The following packages are needed to run the code:

Matplotlib (use command `pip3 install matplotlib`)

Tkinter (use command `sudo apt install python3-tk`)

Pandas (use command `pip3 install pandas`)

Instructions:

Unzip blockchain_data.zip

First run `python3 sampling.py`

The code takes as input the database and runs the sampling algorithm a specified number of times on the specified number of mixins. It outputs 16 new csv files holding the index numbers of the real inputs for each number of mixins the sampling algorithm was run on. (Due to the large size of the database, running the code can take a considerable amount of time)

Now run `python3 effective_untraceability.py`

The code takes as input the csv files from the sampling algorithm and calculates the effective untraceability for each ring signature size (2-16). Then plots the results against the results taken from (Miller, et al., 2018).

2. Stealth Address

Tested with Python 3.6 on Ubuntu 17.10

The following package is needed to run the code:

ed25519 (use command `pip3 install ed25519`)

To run the code, run `python3 address.py`

address.py simulates a stealth address generation using the ed25519 elliptic curve that is used in Monero.

sender.py simulates the sender of a transaction used in *address.py*

ed25519ietf.py implements basic functions for computations on the ed25519 elliptic curve. The code was written by Joseffson, taken from https://tools.ietf.org/html/draft-josefsson-eddsa-ed25519-00#section-4