MATHEMATICS BEHIND THE ZCASH

Nomana Ayesha Majeed
Hochschule Mittweida, Technikumplatz 17, D-09648 Mittweida

Among all the new developed cryptocurrencies, Zcash comes out to be the strongest cryptocurrency providing both transparency and anonymity to the transactions and its users by deploying the strong mathematics of zk-SNARKs. We discussed the zero knowledge proofs as a building block for providing the functionality to zk-SNARKs. It offers schnorr protocol which is further used in Zcash transactions where the validation of sent transaction is proved by cryptographic proof. Further, we deploy zk-SNARKs following common reference string that allows sender to prove that she knows a secret such that the proof is succinct, can be verified and does not leak the secret. Non-malleability, small proofs and effective verification make zk-SNARKs a classic tool in Zcash. We deal with NP problems therefore we have considered the elliptic curve cryptography to provide the security. Lastly, we explain Zcash transaction, the corresponding transaction completely hides the sender, receiver and amount of transaction using zero knowledge proof.

1. Introduction

As for prerequisites, reader is expected to familiar with the Bitcoin [1]; first decentralized cryptocurrency developed in 2008 by Satoshi Nakamoto in the world of cryptocurrencies and act as a base currency in the development of other cryptocurrencies. Bitcoin neither depend on the bank nor rely on any government instead it use a distributed ledger referred as a Blockchain which makes it completely decentralized. However, Bitcoin suffer from some limitations:

- Privacy issue.
- Functionality and scalability limitations.
- Lack of Fungibility.

Beginning from the Zerocoin; a decentralized mix which extends the Bitcoin and is the underlying academic work presented in 2014. The main flaw to Zerocoin is that it only hides the origin of the amount but do not hide the amount itself and receiver of the payment. Another complication with the Zerocoin is its performance; it depends on zero knowledge double discrete logarithm proof which is 25 kbs large and requires 10 seconds for the verification of each spending on the blockchain. It also lake functionality due to the fixed denomination of the coins.

To overcome all of the above mention problems, a team of six outstanding scientists including Zooko Wilcox developed Zcash in October 2016. The edge of using Zcash lies in the fact that Zcash provides the confidentiality and fungibility to its users and their transactions. Beside ensuring the privacy the users can split and merge their coins as well. Zcash is interchangeable due to its anonymity, so one can replace the block of Zcash with another unit of equal value of all coins.

Zcash is defined as a Decentralized Anonymous Payment scheme because it is:

- **Decentralized:** works when given any ideal global ledger like Blockchain.
- **Privacy Preserving**: Anyone can pay directly through payment transactions while keeping origin, amount and the receiver hidden.
- **Efficient:** The transactions take $< 1\,m$ to create and $< 6\,ms$ for verification of the proof.

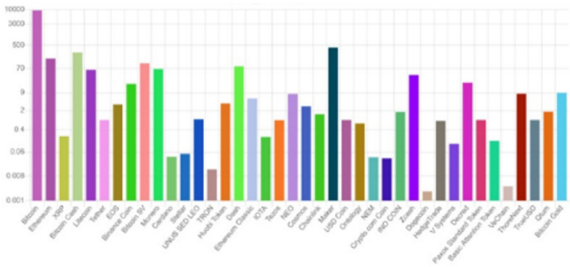
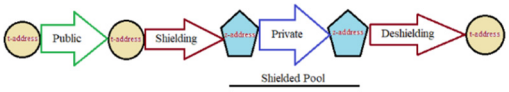


Figure 1.1: Top ranked cryptocurrencies

Beside having shielded transaction, Zcash also support transparent transaction which are same as in Bitcoin in such a way that transparent transactions reveal the pseudonymous addresses of the sender and receiver but to make sure that all the new coins have been shielded once at least it is required to pass them through the shielded pool [2].



1.1 History of Zcash

Symmetric key encryption used by the Julius Ceaser was 1950s is not useful for two parties to communicate with each other unless they are able to deliver the key securely earlier. In 1960's to 1970's Diffie, Hellman and Merkle introduced public key cryptography where Alice uses the public key of Bob to send the message while Bob uses his own private key to read it [3]**.** Later on, Goldwasser et al. (1980) came up with a third new invention of Zero knowledge Proofs. Satoshi Nakamoto (2009), developed blockchain by deploying the idea of public key cryptography where every block contains the proof-of-work and hash. The effective advantage is that one can distribute and share the data in a large group of people without relying on any central party. Finally, Zcash developed in 2016 deployed the idea of ZKP and zk-SNARKs.

1.2 Blockchain

Blockchain is a public ledger of information developed by Satoshi Nakamoto. Since we are working on digital currencies thus, the blockchain provides a digital way to store data and makes it irreversible. Every individual block consists of data, timestamp, hash value of the current and previous block.

To make the blockchain secure it is required to have the same hashes in the current and previous block which makes a chain. Following steps by [4] are required to creates the block for transaction:

1. Assume that Alice has x_1, x_2,…, x_n in her account, she wants to send a transaction to Bob let say y units.
2. Transaction is mint and represented as block.
3. The block is received by every other network.
4. The network node will check the validity of ownership, the Alice is not double spending and $y \leq \sum_{i=1}^{n} x_i$.
5. If everything works well, each participant adds this new block to their own blockchains.
6. All the network nodes make authorization.
7. Finally, transaction is added to Bob's account.

2. Zero Knowledge Proofs

Zero-Knowledge Proofs (ZKP) are used to show the verifier that some secret x belongs to some set $\mathcal{L}$ without providing any information about x except the correctness [5]. Let $\mathcal{L}$ be an NP language defined as $\mathcal{L} = \{ x \mid g^n = x \}$, it follows three properties of zero knowledge proofs as:

1. Completeness: Verifier will be convienced by the prover only if statement x ∈ $\mathcal{L}$.
2. Soundness: A prover cannot prove an honest verifier if the statement x ∉ $\mathcal{L}$ [6].
3. Zero-Knowledge: If the verifier is convienced that x ∈ $\mathcal{L}$ is correct, even though he will not gain any additional information about the secret.

2.1 From interactive to non-interactive

In 1980, Fiat and Shamir provides a method called Fiat-Shamir Heuristic where no interaction is needed between the two parties.It is simply achieved by applying the cryptographic hash function on the challenge [7].

- Alice has to prove to Bob that she knows the solution of $g^x = y$.
- Alice choose random integer $m \in \mathbb{Z}_q^*$ and compute $g^m = t$.
- She choose a challenge $c = Hash\,(g, y, t)$.
- Alice also calculate $r = m - (x * c)$ and publishes the proof (t, r).
- Bob can verify $t = g^r y^c$.

2.1.1 Schnorr Protocol

It [8] is frequently used as a proof of knowledge in DLOG problems for cyclic group $< g >$ with generator g. Suppose that the prover has to show that she knows DLOG x that belongs to $y = g^x \, mod\, p \in \mathbb{Z}_p^*$. The protocol for non-interactive version works as:

- Prover chooses $r \in \mathbb{Z}_{p-1}$ to compute message $t = g^r \,(\mathrm{mod\ p})$ and apply hash function on t to get $c = Hash(t)(mod\ p)$ and calculate $s = r + cx$.
- After hashing, prover publishes (y, c, s).
- Anyone can verify if $c = Hash\ (\ g^s\ y^{-c})$.

For interactive version, we requires interaction of two parties rather then using cryptographic hash function. Authentication of graph isomorphism problem can be taken as an example and can be prove by using [9]:

- Prover randomly chooses $\mathrm{a} \in \{1, 2\}$, a random permutation ρ and generate another graph $I = \rho\,(G_a)$. Send I to verifier.
- Verifier randomly choose $\mathrm{b} \in \{1,2\}$, send b to prover and ask him for σ with $G_b = \sigma\,(I)$.
- Consider $\mathrm{a} = \mathrm{b}$: if graph I results from G_1 or G_2 and the verifier ask prover to map graph I on either of two graphs then the prover will send $\sigma = \rho^{-1}$ to receiver.
- Consider $\mathrm{a} = 1 \,\mathrm{and}\, \mathrm{b} = 2$: if graph I is derived from G_1 and verifier ask prover to map graph I to G_2 then $\sigma = \rho^{-1}\, \mathrm{o}\, \pi$.
- Consider $\mathrm{a} = 2 \,\mathrm{and}\, \mathrm{b} = 1$: if G_2 is used to obtain graph I and verifier asks prover to send a permutation that map graph I to G_1 then prover will send $\sigma = \rho^{-1}\, \mathrm{o}\, \pi^{-1}$ to verifier.

2.1.2 Sigma Protocol

Σ-protocols are 3-move honest verifier zero-knowledge proofs based on commitment, challenge and a response. A binary relation $\mathcal{R}$ for an NP relation is a pair consist of (x, w) where x denotes the instance of statement and w is called witness. Based on $\mathcal{R}$ we define $\mathcal{L}_{\mathcal{R}} = \{ \mathrm{x} \mid \exists\, \mathrm{w} : (\mathrm{x}, \mathrm{w}) \in \mathcal{R} \}$ **[10].** Sigma protocol allows prover to manifest the verifier that statement $x \in \mathcal{L}_{\mathcal{R}}$ without revealing any knowledge about w given $(\mathrm{x}, \mathrm{w}) \in \mathcal{R}$.

Interactive version: Both prover and verifier knows the instance x but private parameter w is known to prover only such that $(\mathrm{x}, \mathrm{w}) \in \mathcal{R}$.

- Prover generates a message m and integer vectors z_1 and z_2 for $z_i \in \{0,1\}^{l_z(n)}$ i-e $P_\Sigma(x, w) \rightarrow (\mathrm{m}, z_1, z_2)$ where $l_n(\mathrm{n})$ is a polyn omial upper bound for security parameter n.
- Verifier chooses $\mathrm{e} \in \{0,1\}^n$ randomly and send challenge to prover.
- Prover respond with $\mathrm{z} = \mathrm{e}z_1 + z_2$ to verifier.
- Verifier return $V_\Sigma(x, m, e, z) \rightarrow \{0,1\}$.
- Verifier will reject the response if any entry $z_i \notin \{0,1\}^{l_z(n)}$ or $e \notin \{0,1\}^n$.

3. zkSNARKs

We begin by considering an idea of Zcash using illustration below:

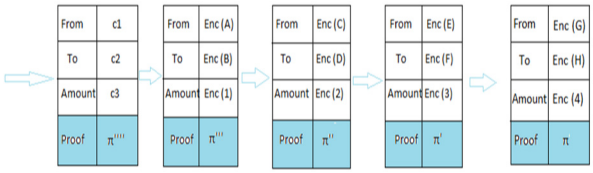


Figure 3.1: Encrypted transactions in Zcash

Suppose that Alice posted four transactions containing the ciphertexts c_1, c_2, c_3, c_4 and a proof π''' as shown in figure 3.1. Since all transactions are ciphertexts so Alice has to generate a cryptographic proof π'''' to prove that she is not double spending.

We start with the "proof" that the true statements have proofs but the false statements do not have. Then "non-interactive" providing the feature of creating & verifying the transaction without having any interaction between the two parties. "Zero knowledge" which reveal nothing else beyond the fact of statement that it is true. "Of knowledge" again which allows crypto that certain knowledge can be proven and lastly "succinct" states that proof is very brief and verification step is easy. Using all of these terms we introduce the notion of "Zero Knowledge Succinct Non-Interactive Argument of Knowledge". The strong privacy of Zcash is based on shielded transaction with are encrypted and the validation is proof by using the zk-SNARKs.

More precisely, we define an NP language as $\mathcal{L}$ [6], a non-deterministic arithmetic circuit C for an instance x. zk-SNARKs is use to prove and verify the existance of instance x in $\mathcal{L}$. Beside considering an input circuit C, a trusted setup is also needed which provides proving key pk and a verifying key vk using common reference string. Proving key enables a prover to create a proof π such that $x \in \mathcal{L}$ and vk is use to verify the proof π without making any interaction. Succinct tells that π should be verified in short time. Construction of zk-SNARKs involves the following steps:

3.1 Homomorphic encryption: For a variable x, [11] define homomorphic encryption $E(x)$ as a function which satisfy the following conditions:

- If $x \neq y \Rightarrow E(x) \neq E(y)$.
- It is very hard to find number x from its encrypted form $E(x)$.
- $E(x+y)$ can be computed from $E(x)$ and $E(y)$.

3.1.1 Constructing homomorphic encryption in $\mathbb{Z}_p^*$:

Construction of homomorphic encryption requires finite groups, finding the solution of discrete logarithm is still unknown in finite fields. Suppose that $\mathbb{Z}_p^*$ is obtained by applying multiplication operator over the prime numbers $\{1,2,\ldots,p-1\}$ and using a multiplication technique $ab = c\,(mod\,p)$ such that the result also lie inside $\{1,2,\ldots,p-1\}$ then;

- All elements of a cyclic group $\mathbb{Z}_p^*$ consisting of prime p and generator g can be written as $E(x) = g^x$.
- Follows from the third property of homomorphic encryption [11]:

  $E(x+y) = g^{(x+y)mod(p-1)} = g^x g^y = \mathrm{E(x)E(y)}$
- $E(x) = g^x$ can be use for linear combinations:

  $E(ax+by) = (g^x)^a (g^y)^b = E(x)^a E(y)^b$

3.2 Hidden evaluation of polynomials

Let $\mathbb{F}_p$ be a finite field of prime numbers consist of elements $\{0,1,\ldots,p-1\}$. We define polynomial P as $P(x) = a_0 + a_1 x + \cdots + a_m x^m$ of degree m. It can be very difficult to solve the polynomial for x since degree of $P(x)$ can take a very large number. Replace x with s and define P at point $s \in \mathbb{F}_p$ $P(s) = a_0 + a_1 s + \cdots + a_m s^m$.

- Alice has polynomial $P(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{F}_p$.
- Bob has a point $s \in \mathbb{F}_p$ and he wants to know $E(P(s))$ where $P(s) = \sum_{i=0}^{d} a_i s^i$.
- Bob send hidings $E(1), E(s), E(s^2), \ldots, E(s^d)$ instead of s to Alice.
- Alice compute $E\big(P(s)\big) = \prod_{i=0}^{d} E(s^i)^{p_i} = g^{p_i s^i}$ from hiding send by Bob and send $E(P(s))$ to Bob.

For non-interactive version, instead of sending $(s^0, s^1, \ldots, s^d)$, Bob will publish $(E(s^0), E(s^1), \ldots, E(s^d))$ on to the CRS where s is a secret parameter that needs to be destroyed.

3.3 Common Reference String

Non-Interactive proofs requires both sender and receiver to have a mutual access to string of random numbers encoded in common reference string. In Zcash, initial setup generates CRS called the public parameters of system, provides non-interactive and short proofs to publish on blockchain. Initial parameters are randomly chosen using pseudorandom generator based on the secret sharing scheme. The initial parameters used to generate CRS needs to be securely destroyed, otherwise CRS could be deceived. Select a random secret s of degree d and a random α so the encryption of secret s is $E(s^i) = g^{s^i}$ for $i = 0,1,\ldots,d$. Encryption of s is available to the prover as $E(s^0), E(s^1), \ldots, E(s^d)$. Based on the cryptographic pairing, we can setup the secure public parameters. Assume that the secret s and α-shifts is constructed by a single trusted party. Initial parameters should be destroyed after the encryption of α and all powers of s w.r.t α. Mainly, CRS is divided into two parts:

- proving key: $(g^{s^i}, g^{\alpha s^i})$
- verification key: $(g^{Z_m(x)}, g^{\alpha})$

Verifier can check the polynomials by using the verification key received by the encrypted polynomial evaluations g^P, g^H and $g^{p'=\alpha P}$. Now the verifier will run the check in two steps:

- According to [12],check if $P = Z_m(x).H$ as: $e(g^P, g^1) = e(g^{Z_m(x)}, g^H) \Leftrightarrow e(g, g)^P = e(g, g)^{Z_m(x).H}$
- Follows from Extended knowledge of coefficient test and assumption; check:

$$e(g^P, g^\alpha) = e(g^\alpha, g^P) = e(g^{\alpha P}, g^1) = e(g^{p'}, g)$$

Since the construction of CRS requires multi party involvement, therefore everyone has to trust that every party has deleted the initial parameters α, *s*.

3.4 Computational to Polynomial

zk-SNARKs cannot be applied directly on computational problem, we need to convert problem to a language of polynomials called Quadratic Arithmetic Program (QAP). We work in polynomial times because it is difficult to lie in polynomial time. For example, if you want to tell the story of execution in polynomials, the true and false statements will differ a lot. Conviencing verifier that solution to the equation $x^2 + 49 = 0$ is known to prover without revealing x we proceed as follow:

$def\ qeval(x)$:

$y = x\ **\ 2 + 49$

At the end, prover shows that she has correctly executed program. But there is no verification method to check if the prover has run a correct program and we have no idea how to observe the y without knowing x. We consider the following steps:

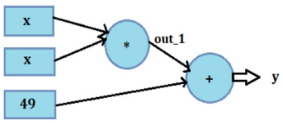
code flattering → Arithmatic Curcuit → R1CS → QAP → zk-SNARK

In the initial step, convert the original code into a sequence of statements called the code flattering as:

$x = y$

$x = y\ (op)\ z$

Where op denotes an arithmetic operation and y, z are variables. Further, consider these statements as gates of an arithmetic circuit:



The flattened code is:

$def\ qeval(x)$:

$out\_1 = x * x$

$y = out\_1 + 49$

Here, Prover has to prove that she knows the consistent assignment of the variable solving $x^2 + 49 = 0$. We start with Rank One Constraint System to check if all the steps are performed accurately. It is a list of triplets of vectors $\langle \vec{a_i}, \vec{b_i}, \vec{c_i} \rangle$ and its solution is vector $\vec{s}$, such that:

$$\langle \vec{a_i}, \vec{s} \rangle * \langle \vec{b_i}, \vec{s} \rangle - \langle \vec{c_i}, \vec{s} \rangle = 0 \qquad (3.1)$$

Each component of this vector will be one variable. Where vector $\vec{s}$ tells the state of variables being considered in the program. More generally, for $s = (one, x, out\_1, y)^T$ we can write 3.1 as:

$$cs = as * bs \qquad (3.2)$$

Second statement $y = out\_1 + 49$ can be written as:

$$\begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}one\\x\\out\_1\\y\end{pmatrix}=\begin{pmatrix}49\\0\\1\\0\end{pmatrix}\begin{pmatrix}one\\x\\out\_1\\y\end{pmatrix} * \begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}one\\x\\out\_1\\y\end{pmatrix}$$

Logic of the code has been transferred to the triplets of constraints vector $\vec{a_i}, \vec{b_i}, \vec{c_i} \in \mathbb{F}_p^n$ for n number of variables. Thus, every statement can be represent by using the three vectors $\vec{a_i}, \vec{b_i}\ and\ \vec{c_i}$. In general, there are n variables which define the length of the vector and m statements. To complete the transformation to QAP, we switch from vectorial to polynomial representation and define polynomials as:

$A_i(x),\ B_i(x),\ C_i(x)$ for all $i \in \{1, n\}$ [13][14]

QAP with m statements and n variables is represented by m triples of vector $\vec{a_i}, \vec{b_i}, \vec{c_i} \in \mathbb{F}_p^n$.

$$R_1 = (a_1, b_1, c_1), \dots, R_m = (a_m, b_m, c_m)$$

Transformation is done by using the langrange interpolation which is use to find the polynomial from the set of points (x, y) that passes through all these points, since we have n variables and three constraint vectors therefore, we obtain $3n$ polynomial:

$$A_1(x), A_2(x), \dots, A_n(x),$$
$$B_1(x), B_2(x), \dots, B_n(x),$$
$$C_1(x), C_2(x), \dots, C_n(x)\ [13][14]$$

Express polynomial as $P_s(x) = A_s(x) * B_s(x) - C_s(x)$. Thus, we have:

$$P_s(x) = \begin{pmatrix}A_1(x)\\.\\.\\.\\A_n(x)\end{pmatrix}\begin{pmatrix}s_1\\.\\.\\.\\s_n\end{pmatrix} * \begin{pmatrix}B_1(x)\\.\\.\\.\\B_n(x)\end{pmatrix}\begin{pmatrix}s_1\\.\\.\\.\\s_n\end{pmatrix} - \begin{pmatrix}C_1(x)\\.\\.\\.\\C_n(x)\end{pmatrix}\begin{pmatrix}s_1\\.\\.\\.\\s_n\end{pmatrix}$$

for $x = 1, \dots, m$. Subsequently;

$$P_s(x) = A_s(x) * B_s(x) - C_s(x) = H_s(x) * Z_m(x)$$

The value of polynomial at target polynomial $Z_m(x)$ is 0 except those included at the target point. The degree of $H_s(x) \le (m-2)$, $P_s(x) \le 2(m-1)$ and $Z_m(x) = (x-1)(x-2)\dots(x-m)$ [11]. The purpose of using $H_s(x)\ and\ Z_m(x)$ is that the prover has to show that she knows legal assignment s by solving quadratic arithmetic program hence, polynomial $A_s(x) * B_s(x) - C_s(x) = 0$ iff target polynomial $Z_m(x) | P_s(x)$:

$\exists\ H_s(x) : A_s(x) * B_s(x) - C_s(x) = H_s(x) * Z_m(x)$ [13]

$$H_s(x) = \frac{A_s(x) * B_s(x) - C_s(x)}{Z_m(x)} = \frac{P_s(x)}{Z_m(x)}$$

Hence; QAP is a 4-tuple: $(\vec{a_i}, \vec{b_i}, \vec{c_i}, Z)$ with solution:

$$s = \begin{pmatrix}s_1\\.\\.\\.\\s_n\end{pmatrix}$$

3.5 Pinocchio protocol

Bob need to know whether Alice has a valid assignment or not, Bob will use method of Pinocchio protocol introduced by [11] to test the validation:

- Bob choose random point $k \in \mathbb{F}_p$ and send $E(k)$ to Alice.
- Alice choose polynomials $A_s(k), B_s(k), C_s(k), H_s(k)$ and send the corresponding hidings $E(H_s(k)), E(A_s(k)), E(B_s(k)), E(C_s(k))$.
- Bob will check if equality $E(A_s(\mathrm{k}) * B_s(\mathrm{k}) - C_s(\mathrm{k})) = E(H_s(\mathrm{k}) * Z_m(\mathrm{k}))$ holds.

If the equality holds then Bob will accept that Alice knows the valid assignments. Pinocchio protocol follows from the method of blind evaluation of polynomials. Therefore, further calculations can be solved by using hidden evaluation of polynomials.

3.7 Elliptic Curve Pairing

In Pinocchio protocol Bob need to evaluate $E(H_s(\mathrm{k}) * Z_m(\mathrm{k}))$ based on the individual hidings $E(H_s(\mathrm{k}))$ and $E(Z_m(\mathrm{k}))$ but In general,

$$E(H_s(\mathrm{k}) * Z_m(\mathrm{k})) \neq E(H_s(\mathrm{k}))E(Z_m(\mathrm{k}))$$ [11]

We define a pairing over an elliptic curve to check the quadratic constraints of the system. [15] defined an elliptic curve E over $\mathbb{F}_p$ satisfy the following equation:

$$y^2 = x^3 + ax + b \qquad (3.4)$$

Group of elliptic curve consists of addition operation, point at infinity O and set of pairs (x, y) over F_{p^k}. The elliptic curve security relies on solving hard DLOG problem therefore elliptic curve cryptography provides the same security using secret key but with a very small parameter key size as compare to the other cryptosystems. The fastest algorithm to solve the DLOG in elliptic curve requires exponential time. For instance; consider an elliptic curve below.

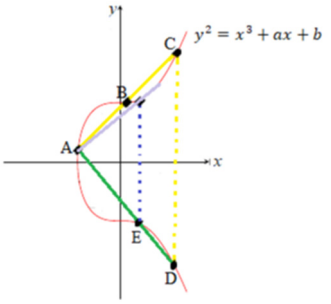


Figure 3.3: Elliptic Curve Cryptography

E over F_p from [15] showed in figure [3.3] starts with generator point $A \in \mathrm{E}/F_{p^k}$ and another point $\mathrm{Z} \in \mathrm{E}/F_{p^k}$. For an elliptic curve E, $ECDLOG$ is to find an integer n for given element A and another element Z such that $n.A = Z$. We prefer an elliptic curve over the other public key cryptosystems because it provides high security with a small bit size then that of RSA which rely on large number factorization.

As proposed by [16], prover has to show the verifier that she knows such n that results $Z = n.A$ without revealing n. The algorithm work as following:

- Prover calculate another $B = r.A$ after choosing a random $\mathrm{r} \in F_{p^k}$ and send to verifier.
- Verifier respond with binary choice $b \in \{1,2\}$.
- Prover will send r if $b = 1$ otherwise $m = n + r\ (mod\ p)$.
- Verifier will check $B = r.A$, for $b = 1$ otherwise $m.A = (n + r)(A) = Z + B$.

The algorithm will keep on repeating unless the verifier get to know that prover knows n. The number of iterations will be $\frac{1}{2^k}$. Now if the prover is a fake then she can generate a random message m and compute $m.A - Z = B$ and send B for verification but if in other case she need instance of ECDLOG r to generate B therefore, prover will be able to perform this case only if she is honest.

Definition 3.7.2 [17] defined Pairing as a bilinear and non-degenerate mapping over an elliptic curve:

$$\mathrm{e}: G_1 * G_2 \rightarrow \mathrm{G} \in F_{p^k}$$

where $G_1 \in \mathbb{F}_p$ denotes subgraph generated by a rational point on elliptic curve with order r. $G_2 \in \mathbb{F}_p$ is another subgraph obtained by a twisted curve over an elliptic curve. G define the subgroup of the field that belongs to non-zero elements of a finite field of p^k where k is an embedded degree or the minimum integer such that $\frac{(p^k-1)}{r}$. Based on information provided by pairing with elliptic curve, let G_1 and G_2 be the cyclic subgroup over F_{p^k} and G be the subgroup of $\mathbb{F}^*_{p^k}$ with order of $|G_1| = |G_2| = |G| = r$. For fix generators $g_1 \in G_1, g_2 \in G_2, g \in G$ and given hidings $E_1(\mathrm{x}) = x.g_1, E_2(\mathrm{x}) = x.g_2, E(x) = x.g$ we have:

$$E(xy) = Tate(E_1(x), E_2(y))$$

4. Zcash Transactions

For an arithmetic circuit C, [17] defined zk-SNARK as a triple of polynomial time algorithm:

- $KeyGen(1^\lambda, C) \rightarrow (a_{pk}, a_{vk})$

  Given the λ security parameter and circuit, KeyGen algorithm will generate the proving key and a verifying key called public parameter. These pp are accessible to every participant and can be use over and over to prove and verify that the statement x belongs to $\mathcal{L}_c$.

- $Prove(a_{pk}, x, w) \rightarrow \pi$:

  Prover will create a proof π and binary relationship $(x, w) \in \mathcal{R}$ such that instance $x \in \mathcal{L}_c$.

- $Verify(a_{vk}, x, \pi) \rightarrow b$:

  For the given statement x, vk along with the $non - interactive$ proof π, the receiver will return bit $b = 1$ if $x \in \mathcal{L}_c$.

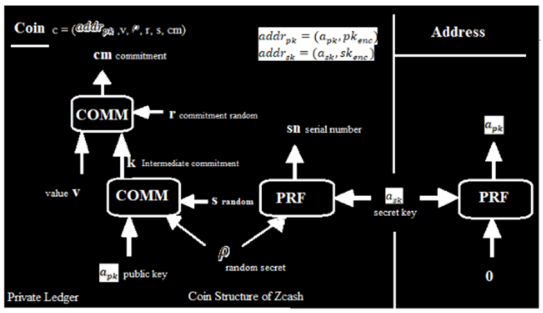


Figure 4.1: Coin Structure of Zcash

Setup: For figure [4.1], public parameters are created by the trusted setup, based on security parameters λ which is use to create the pp we have:

$$pp = \{pk_{POUR}, vk_{POUR}, pp_{enc}, pp_{sig}\}$$

Coin c consist of coin commitment cm, a string available inside the Merkle tree that makes commitment to the serial number sn of the coin when the coin is minted, value v which is the denomination of the coin ranges from 0 to v_{max}. Serial number sn is singular string devoted to the coin to avoid double spending and is hidden. Trapdoors r and s are to enhance the security. $addr_{pk}$ is use by others participats to send the transactions to the user. $addr_{sk}$ is use to obtain the payments sent to the address public key. Random secret ρ is use to obtained the sn and cm. Each user can create any number of address key pair (a_{pk}, a_{sk}). Using the pp we obtain the address key pair $(addr_{pk}= (a_{pk}, pk_{enc}), addr_{sk}= (a_{sk}, sk_{enc}))$ where (pk_{enc}, sk_{enc}) = κ_{enc} (pp_{enc}). Beside transparent transactions like in Bitcoin, Zcash mainly deals with two type of transactions:

4.1 Mint Transacton:

Mint transaction itself is a cryptographic commitment to the new coin *c*. Let say, Alice spend v BTC to create a value v coin with coin commitment cm.

Input: $pp, v, addr_{pk}$ to send the transaction.

Output: *c* = $(addr_{pk}, v, \rho, r, s, cm)$

for $cm = COMM_r(v \parallel k)$ and $k = COMM_s(a_{pk} \parallel \rho)$ with mint transaction $tX_{MINT} = (cm, v, k, r)$.

Verification: Since the unique sn of coin c is hidden so we need to validate the transaction:

Input: $pp, tX_{MINT}, Merkle\ tree$

Output: Verifier will return bit *b = 1* if:

$$\text{cm} = cm' = COMM_r(v \parallel k).$$

4.2 Pour Transaction

This algorithm [6] enables the user to split and merge the coins. User can use this method to spend the coins, to send the coins, can make change etc. It also contains the information string which tells that who is the recipient of public value and when the transaction was made.

Input: $pp, Merkle\ root\ rt$, two old Zcash coins to be consumed c_1^{old},c_2^{old} with address secret keys $addr_{sk,i}^{old}$, $path_i$ from c_i^{old}, desired input value of new ZEC coin v_i^{new}, destination $addr_{pk,i}^{new}$ for each $i \in \{1,2\}$ and public value v_{pub} that lies between $0 \leq v(c) \leq v_{max}$.

Algorithm: Alice want to consume old coins $c_i^{old} = (addr_{pk,i}^{old}, v_i^{old}, \rho_i^{old}, r_i^{old}, s_i^{old}, cm_i^{old})$ in order to create the new coins c_1^{new} and c_2^{new}. The algorithm works as follow:

- Parse c_i^{old} and yields new coins $c_i^{new} = (addr_{pk,i}^{new}, v_i^{new}, \rho_i^{new}, r_i^{new}, s_i^{new}, cm_i^{new})$ for $i \in \{1,2\}$ with $addr_{sk,i}^{old} = (a_{sk,i}^{old}, sk_{enc,i}^{old})$ and $addr_{pk,i}^{new} = (a_{pk,i}^{new}, pk_{enc,i}^{new})$.
- $tX_{POUR} = (\text{rt}, sn_1^{old}, sn_2^{old}, cm_1^{new}, cm_2^{new}, v_{pub}, \text{info}, pk_{sig}, \pi_{POUR}, h_1, h_2, C_1, C_2, \sigma)$.

Where $C_i = \mathcal{E}_{enc}(pk_{enc,i}^{new}, (v_i^{new}, \rho_i^{new}, r_i^{new}, s_i^{new}))$ is ciphertext computed by Alice to transfer the ownership of coin to Bob. $h_i = PRF_{a_{sk,i}^{old}}^{pk}(i \parallel h_{sig})$ for $i \in \{1,2\}$ works as a MACs that associates h_{sig} with address secret keys using pseudo random function. $\pi_{POUR} = Prove(pk_{POUR}, x, w)$ is prove from Alice that computations are done correctly using pk of pour transaction computed at the setup phase on instance x and witness w. Alice use the sk_{sig} to sign each value that is connected with the pour operation denoted by message $m = (x, \pi_{POUR}, info, C_i)$ to obtain sigma signature $\sigma = \mathcal{S}_{sig}(sk_{sig}, m)$. In order to deploy the efficiency, Zcash use zk-SNARKs and an additional requirement for avoiding the malleability attack is by using digital signature. Zcash compute the MACs to combine the secret keys with signing key, then update the instance x by adding signature verification key and MACs. Finally, each transaction is signed. Also, Bob can decrypt this message using his encrypted secret key by scanning the tX_{POUR} on public ledger.

Transaction verification:

- sn_1^{old}, sn_2^{old} do not already in ledger otherwise $b = 0$.
- If $sn_1^{old} \neq sn_2^{old}$ then $b = 1$.
- Compute $b = \mathcal{V}_{sig}(pk_{sig}, m, \sigma)$. From the given digital signature based public key, message *m* and the σ signature, if sigma is valid signature for m then return $b = 1$.
- Finally, for the given verification key derived from pp, instance x and proof π verify:

$$b' = verify(vk_{POUR}, x, \pi_{POUR}) = \begin{cases} 1, & x \in \mathcal{L}_c \\ 0, & otherwise. \end{cases}$$

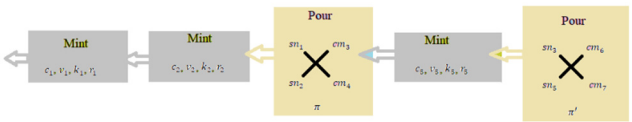


Figure 4.2: Pour transaction blockchain View

zk-SNARK proof: Alice consume two input coins c_1^{old},c_2^{old} with serial numbers s_1^{old}, s_2^{old} in order to create two output coins c_1^{new}, c_2^{new} with cm_1^{new}, cm_2^{new}

and public output v_{pub}. For an $NP\ statement, zk-SNARK\ proof\ \pi$ that Alice know secret is:

Given the Merkle tree root *rt* containing *cm* of all minted coins, serial number *sn* of old coins and cm_1^{new}, cm_2^{new} of new coins. Alice know $c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}$ and the secret key a_{sk}^{old} such that:

- Coins $c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}$ are well formed.
- $a_{pk,i}^{old} = PRF_{a_{sk,i}^{old}}^{addr}(0)$.
- Revealed s_1^{old}, s_2^{old} are of old coins and are calculated correctly as $s_i^{old} = PRF_{a_{sk,i}^{old}}(\rho_i^{old})$.
- cm_i^{old} of c_i^{old} appears on CRH-based Merkle tree with rt.
- Revealed cm_1^{new}, cm_2^{new} are of c_1^{new}, c_2^{new}.
- $v_1^{new} + v_2^{new} + v_{pub} = v^{old}$.

If all the conditions stated above hold then witness w is valid for instance x.

Fetching coins: This algorithm provides the list of all those coins whose sn do not appear already on the blockchain $\mathcal{L}$.

Input: address key pair:$(addr_{sk}, addr_{pk})$ and current ledger $\mathcal{L}$.

Output: receive coins if:

- $sn_i = PRF_{a_{sk,i}^{sn}}(\rho_i)$ does not appear in ledger.
- $cm_i^{new} = COMM_{r_i^{new}}(v_i^{new} \parallel k_i^{new})$.

Transaction Anonymity:

- If Alice do not know secret key $addr_{sk}$ then based on this knowledge she cannot spend the coin. The core security lies in the secret key since $addr_{pk}$ generated by $addr_{sk}$ is required in order to create an address of coin. Given secret key $addr_{sk}$ is also apply in witness w which is requisite in creating a zk−SNARK proof π so if Alice do not know the secret then she cannot create a proof π.

5. Future Considerations

Besides having the strong mathematical algorithms used in the development of Zcash there are still some flaws in term of privacy, decentralization and efficiency that can be improve in Zcash.

- The factorization and DLOG problems on EC are at risk by the quantum computer. It is very challenging to design quantum resistant anonymous algorithms that can protect user's privacy and run anonymous authentication in quantum systems. Currently, Zcash rely on one-way-hash where user creates the digital signature using these trapdoors in order to validate the transaction. Quantum computers can be use to break these cryptographic codes as trapdoors rely on large prime numbers factorization. Based on this knowledge, we can consider two main features "trapdoors, digital signature" in Zcash. Either replace the trapdoor with other unbreakable system or upgrade to trapdoor free structure. Secondly, digital signature can be exploit by using the shor's algorithm which also deploy the idea of Fourier transform for prime number factorization based on quantum computers. One solution to secure the digital signature can be using the same idea deployed by the Monero which works using ring signature.
- Privacy can be a risk to governments as well. Anyone can use this channel for money laundering. Still considering the dark aspect of transparency everyone wants to move to privacy, a possible solution to stop using privacy for illicit acts can be by adding a security check in the pipeline connecting the two parties.
- Another possibility to raise the privacy can be by expiration of particular secret key after a certain time limit.
- The user need to pay more when select a private transaction, 20% reward of it goes to the founder which is expensive. It is required to make improvements in the shielded transaction to make it more efficient beside consuming less memory space and enhancing the security.
- Comparison between Zcash, Monero and Ethereum:

Features	**Zcash**	**Monero**	**Ethereum**
Type	Digital currency	Digital currency	DC/Blockchain platform
Supplies	21 Million	18.4 Million	18 Million annual
Block confirmation time	2.5 min	2 min	15 sec
Block size	2 MB	Dynamic	1 MB
Hashing alogorithm	Zk-SNARKs	CryptoNight	Ethash
Rank	28	10	2
Transaction per sec.	6-25	>1700	Approx. 4

Table 4.1: Zcash Comparison with other cryptocurrencies [18][6][19]

- Promotion in the decentralization is still required by allowing multiple parties to participate in Zcash development. As mentioned in CRS that initial parameters are completely destroyed but there is no way to verify it. There should be a verification method so the user can invest in Zcash without any doubt.
- Scalability need to enhance in Zcash, it should be in the access of every person. Also, it takes 40 seconds to create the transaction. Zcash need to develop a network that can handle several transactions in seconds but currently private transactions slow down the process. To make it possible, main network can be split into subnets so that

only the relevant network work when requires instead of running the whole setup. New block creation time in Zcash is 2.5 minutes while that is Monero is 2 minute and the block size is 2MB in Zcash which is high in case of private transactions. Monero takes the advantage here in term of scalability due to its dynamic block size.

Acknowledgement

This present survey is based on the author's Master's thesis. The author expresses her gratitude to Professor Klaus Dohmen for his kind supervision.

References

[1] B. S. d. Albuquerque and M. d. C. Callado, “Understanding Bitcoins: Facts and Questions,” Revista Brasileira de Economia, vol. 69, pp. 3--16, 2015.

[2] G. Kappos, H. Yousaf, M. Maller and S. Meiklejohn, An empirical analysis of anonymity in zcash, 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 463--477.

[3] S. D. Galbraith, Mathematics of public key cryptography, Cambridge University Press, 2012.

[4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Manubot, 2019.

[5] K. Balasubramanian and M. Rajakani, Algorithmic strategies for solving complex problems in cryptography, IGI Global, 2017.

[6] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in IEEE Symposium on Security and Privacy, San Jose, CA, 2014.

[7] T. Koens, C. Ramaekers and C. v. Wijk, “Efficient zero-knowledge range proofs in ethereum,” ING, 2018.

[8] R. Cramer, I. Damgård and B. Schoenmakers, “Proofs of partial knowledge and simplified design of witness hiding protocols,” in Annual International Cryptology Conference, 1994.

[9] E. Ayeh, “An Investigation Into Graph Isomorphism Based Zero-knowledge Proofs,” PhD thesis, 2009.

[10] P. Chaidos and J. Groth, “Making sigma-protocols non-interactive without random oracles,” in IACR International Workshop on Public Key Cryptography},, 2015.

[11] A. Gabizon, What are zk-SNARKs?, 2017.

[12] M. Petkus, “Why and How zk-SNARK Works,” arXiv preprint arXiv:1906.07221, 2019.

[13] V. Buterin, Zk-SNARKs: Under the Hood, Medium, 2017.

[14] S. D. Valentin Ganev, Introduction to zk-SNARKs, 2018.

[15] C. Paar and J. Pelzl, Understanding cryptography: a textbook for students and practitioners, Springer Science & Business Media, 2009.

[16] I. Chatzigiannakis, A. Pyrgelis, P. G. Spirakis and Y. C. Stamatiou, “Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices,” in IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, 2011.

[17] E. Ben-Sasson, A. Chiesa, E. Tromer and M. Virza, “Scalable zero knowledge via cycles of elliptic curves,” Algorithmica, vol. 79, pp. 1102--1160, 2017.

[18] KOE, K. M. ALONSO and S. NOETHER, Zero to Monero: Second Edition, April 4, 2020.

[19] D. Vujičić, D. Jagodić and S. Ranđić, “Blockchain technology, bitcoin, and Ethereum: A brief overview,” in 17th international symposium infoteh-jahorina (infoteh), 2018.