ZeroCT: Improving Zerocoin with Confidential Transactions and more

Alex Vazquez
alex@encrypt-s.com

January 17, 2019

Abstract

The Zerocoin protocol [4] is a set of cryptographic algorithms which embedded in a cryptocurrency provide anonymous swap of tokens in a mathematically provable way by using cryptographic accumulators. Functionally it can be described as a black box where an actor can introduce an arbitrary number of coins, and later withdraw them without leaving evidence of connection between both actions. The withdrawing step admits a destination for the coins different from the original minter, but unconditionally requires a previous mint action and does not accept the transfer of coins without leaving the accumulator, thus exposing the traceability of the coins. We propose an alternative design which for the first time combines the virtues of Zerocoin [4] with those of Confidential Transactions [17] offering fully-featured anonymous transactions between individuals with private amounts.

1 Introduction

We can find implementations of Zerocoin in a "production" environment in active projects like ZCoin [1] or PIVX [2]. Those stick to the original protocol [4] where the set of actions a user can execute are limited to minting and spending coins, working the system as a simple on-chain mixer where a previous step of *coin laundering* is necessary before the transfer of value is anonymously possible. Other alternative variations of the protocol [6] operate in a similar way, while some newer variations [8] introduce the concept of sending Zerocoins to an external party but still require an initial interactive setup step and only allow one deposit per key. Although the system satisfies the necessary conditions to consider it functionally anonymous, we would like to point out some drawbacks from the original implementation for which we propose a solution in this paper.

- Because only transparent addresses exist, it promotes the use of the Zerocoin accumulators as purely transitional for the laundering of coins between transparent addresses, being transaction traceability and address linkability through chain analysis moderately plausible. Even if a mechanism consisting of rewarding users for keeping coins in the Zerocoin pool [7] is an example of good action to increase the anonymity set, it does not prevent the fact that coins need to leave the anonymity pool in order to be transferred, which is the final utility of a currency, to be transferred and used. The anonymity of the Zerocoin protocol is upper-bounded by the size of the pool of coins, while the size of the anonymity pool is linearly related to the amount of coins sitting in the accumulators but inversely correlated to the number of transactions between users.

  We introduce the use of Anonymous Identities, similar to the concept of Stealth Addresses existent in other cryptocurrencies, allowing the private transfer of coins between different entities without the requirement of using transparent addresses and incorporating the size of the transactional ledger to the anonymity pool.

- Privacy concerns aside, the use of *clear-text* denominations reduces the usability of the system, by increasing the number of required coins and therefore the total size of the necessary proofs. Let $e_0, \cdots, e_z$ the set of different denominations supported by a Zerocoin implementation, the transaction amount can be decomposed as $\sum_{i=0}^{z} a_i e_i$. For a single spend proof message size W, the full communication cost for the spend proofs of a transaction is a function of its value and can be expressed as

$$W \sum_{i=0}^{z} a_i$$

  By applying variations of known methods inspired by [17], our implementation allows the transfer of divisible amounts to be expressed as a secret value only known to the participants of the transaction with the use of just two Accumulators.

2 Notation

Let us define some notation and variables that will be used through this paper. Let $l \leq k$ two security parameters determining the security of the zero knowledge proofs and $u \leq (\log_2 q) - 2$ the number of bits necessary to have enough precision for transaction amounts. The concatenation of two bit arrays of arbitrary length α and β is denoted by $\alpha||\beta$. The binary operation *XOR* will be denoted with the operator $\oplus$. Let $\mathrm{H} : \{0,1\}* \rightarrow \{0,1\}^l$ a one-way cryptographic function taking a bit array of arbitrary length as input and outputting a bit array of length l. The function H_s takes an EC point as input and outputs the result of

feeding its binary representation into H. The bit in the position i of a bit array a is denoted $a[i]$, considering $a[0]$ the bit in the left-most position of the array. When describing Zero Knowledge proofs we will use the notation of Camenisch and Stadler [15]. For instance, ZKSoK$[m]\{(x) : h = g^x\}$ denotes a signature of knowledge on message m of the element x that satisfies $h = g^x$, where all values not enclosed in () are assumed to be known to the verifier. If A is a set, $a \leftarrow A$ means that a is chosen at random from A according to the uniform distribution. If A is a function, $a \leftarrow A(\cdots)$ means a is assigned the value returned from executing A with the given parameters.

3 A mathematical introduction to the Mint operation

The Zerocoin protocol [4] defines the Mint operation as the operation of minting new private tokens (Zerocoins in the original definition). As in a regular Bitcoin transaction [3], it requires that the amount of inputs used to feed the transaction is equal to the value of the minted private tokens plus any fees defined by the network policies.

"To mint a zerocoin c of denomination d, Alice runs Mint(*params*) $\rightarrow$ (*c,skc*) and stores *skc* securely. She then embeds c in the output of a Bitcoin transaction that spends d + fees classical bitcoins. Once a mint transaction has been accepted into the block chain, c is included in the global accumulator A, and the currency cannot be accessed except through a Zerocoin spend" [4]

Mathematically minting a coin means calculating a Pedersen Commitment [12] which value will be later accumulated in the accumulator of the corresponding denomination. A Pedersen Commitment is a one-way function where you can commit to a value $\mathfrak{v}$ under a blinding factor s without revealing the value $\mathfrak{v}$ until a later time:

$$c = g^{\mathfrak{v}} h^s \pmod{p}$$

Additionally this structure admits commitments to $\mathfrak{n}$ different values at the same time in the form $c = h^s \prod_{i=0}^{\mathfrak{n}} g_i^{\mathfrak{v}_i}$. Each additional generator g_i can be calculated as $g_i = \mathrm{H}(g_{i-1}) \pmod{p}$ for $i \geq 1$. For simplicity, we will denote $\mathfrak{v}_0$ and g_0 as simply $\mathfrak{v}$ and g.

Given c, finding s and $\mathfrak{v}_\mathfrak{i}$ for $0 \leq i \leq \mathfrak{n}$ is known as the Discrete Logarithm Problem, it's "hard" to solve and there isn't currently any known efficient method for computing the solution in reasonable time even if some but not all of the values of the set (s, $\mathfrak{v}_1$, ..., $\mathfrak{v}_\mathfrak{n}$) are known. Because of the hardness of finding suitable values for $\mathfrak{v}_i$ and s, a Pedersen Commitment $c \leftarrow \mathrm{C}(\mathfrak{v}, s, \mathfrak{v}_1, \ldots, \mathfrak{v}_\mathfrak{n})$ is both hiding (the Commitment c does not reveal the value it commits to) and

binding (having made the Commitment c it's not possible to open it with different values for $\mathfrak{v}_i$ or s) as long as $\log_{\mathfrak{g}_{i'}} \mathfrak{g}_i$ is unknown with $\mathfrak{g} = (h, g_i, \ldots, g_n)$ for $0 \leq i \leq \mathfrak{n}$ and $0 \leq i\prime \leq \mathfrak{n}$.

Pedersen Commitments also have homomorphic properties: The product of two commitments is equal to the commitment to the sum of its values as in

$$\mathrm{C}(\mathfrak{v}_a, s_a) \cdot \mathrm{C}(\mathfrak{v}_b, s_b) = \mathrm{C}(\mathfrak{v}_a + \mathfrak{v}_b, s_a + s_b) = g^{\mathfrak{v}_a} h^{s_a} g^{\mathfrak{v}_b} h^{s_b} = g^{\mathfrak{v}_a + \mathfrak{v}_b} h^{s_a + s_b}$$

In the same fashion, the division of two commitments equals the commitment to the subtraction of its values as in

$$\frac{\mathrm{C}(\mathfrak{v}_a, s_a)}{\mathrm{C}(\mathfrak{v}_b, s_b)} = \mathrm{C}(\mathfrak{v}_a - \mathfrak{v}_b, s_a - s_b) = \frac{g^{\mathfrak{v}_a} h^{s_a}}{g^{\mathfrak{v}_b} h^{s_b}} = g^{\mathfrak{v}_a} h^{s_a} g^{-\mathfrak{v}_b} h^{-s_b} = g^{\mathfrak{v}_a - \mathfrak{v}_b} h^{s_a - s_b}$$

The original Zerocoin protocol [4] uses a RNG to generate different values for $S \leftarrow \mathbb{Z}_q^*$ (*serial number* of the minted coin) and $r \leftarrow \mathbb{Z}_q^*$ (*randomness* used as a blind factor) to be used in the computation of $c \leftarrow \mathrm{C}(S, r)$ until $\{c$ prime $— c \in [A, B]\}$ [14]. The future spender of the minted coin is required to prove knowledge of both values S and r constraining the spending action to the original minter. Our contribution allows an actor to commit in zero knowledge to secret values only known to an external party, even if those are publicly disclosed later.

4 Constructing a transaction

We will start defining how an anonymous identity is constructed. Let B the public part of a Elliptic Curve key $B = bG$, $j_1 \leftarrow \mathbb{Z}_q^*$, $k_1 \leftarrow \mathbb{Z}_q^*$, $j_2 \leftarrow \mathbb{Z}_q^*$, $k_2 \leftarrow \mathbb{Z}_q^*$, $z_1 \leftarrow \mathrm{C}(j_1, k_1)$ and $z_2 \leftarrow \mathrm{C}(j_2, k_2)$. The triplet (B, z_1, z_2) is known as the anonymous identity $\mathcal{I}$, which can be publicly shared and used as an address where users can receive private coins. The tuple (b, j_1, j_2, z_1, z_2) is considered a private view key PK_{view} and allows the wallet to identify which outputs contain spendable private coins and when those coins are spent. PK_{view} can be handed to an accountant to prove an account's history of private transactions without compromising the spending rights exclusivity of funds. The tuple (b, j_1, j_2, k_1, k_2) is considered a private spend key PK_{spend} and allows to construct the cryptographic proofs necessary to spend private coins. Anonymous identities admit receiving as a single output an arbitrary and divisible amount in the range $[0, 2^u)$ denoted as $\mathfrak{w}$.

We redefine the Mint algorithm as $\mathrm{Mint}(params, \mathcal{I}, \mathfrak{w})$, so when Alice wants to send coins to Bob's Private Identity $\mathcal{I}_{\mathcal{BOB}}$ she:

1. Extracts B, z_1 and z_2 from $\mathcal{I}_{\mathcal{BOB}}$.

2. Generates a new EC key $A = aG$ and calculates a Diffie-Helman secret χ using Bob's EC public key B.

$$\chi = \mathrm{H}_s(aB) \pmod{q}$$

3. Uses H as a Pseudorandom Number Generator to compute σ and ϱ taking the shared secret χ as the initial seed.

$$\sigma = \mathrm{H}(\chi) \pmod{q}$$

$$\varrho = \mathrm{H}(\sigma) \pmod{2^u - 1}$$

4. Lets $c = z_1^{\chi} z_2 \pmod{p}$ and $\epsilon \leftarrow C(\mathfrak{w}, c, \sigma)$.

5. Verifies c and ϵ are prime numbers and within the allowed range required in the accumulator proof [14]. If the test fails, she repeats the process going back to the second step. If it passes, she continues with the next step.

6. Includes a zero knowledge range proof that the value committed in ϵ is a positive number and lies in the range $[0, 2^u)$.

$$\mathrm{NIZKPoK}\{(v, \sigma) : \epsilon = g^{\mathfrak{w}} h^c g_1^{\sigma} \wedge 0 \leq v \leq 2^u - 1\}$$

   Methods like Bulletproofs [16] allow provers to bundle many range proofs in one of compressed size, making it possible to compute one proof per transaction instead of using the more expensive model of one-proof-per-output.

7. Lets $\mathfrak{W} = \mathfrak{w} \oplus \varrho$ the amount obfuscated with ϱ.

8. Reveals $(A, c, \epsilon, \mathfrak{W})$ in the output of a transaction.

Considering the following equality is satisfied

$$c = z_1^{\chi} z_2 = (g^{j_1})^{\chi} (h^{k_1})^{\chi} g^{j_2} h^{k_2} = g^{(j_1\chi + j_2)} h^{(k_1\chi + k_2)} \pmod{p}$$

we can claim c is equivalent to a Pedersen Commitment with one secret and one randomness value. Alice knows z_1, z_2 and χ but she does not have knowledge of j_1, j_2, k_1 or k_2 because of the properties of the Pedersen Commitment and under the assumption of the hardness of the Discrete Log Problem, thus she would be committing without retaining the ability of later opening the commitment by using the serial number $S = j_1\chi + j_2$ or the randomness $r = k_1\chi + k_2$ in the construction of the proofs that are necessary to spend the coins. This scheme retains the perfectly hidden property from the Pedersen Commitment construction as j_1, j_2, k_1 and k_2 are uniformly drawn from $\mathbb{Z}_q^*$ while χ is calculated *mod* q, being the distribution of the resulting $j_1\chi + j_2$ and $k_1\chi + k_2$ equally uniform.

An actor observing the chain and acting as a validator would accumulate c and ϵ in different accumulators A and V respectively.

The private key a will be stored by Alice and used to prove the minting of specific coins without revealing Alice's whole transaction history or identity.

Due to the use of only one anonymous identity to receive coins, this scheme does not facilitate the use of short-lived addresses to identify individual payments, which is a common use case for merchants in other cryptocurrencies like Bitcoin. To solve this we propose the calculation of an extra parameter $o = \mathrm{H}(\varrho)$ used to obfuscate a Payment ID/Message $\mathbb{M}$ as in $\mathbb{M}' = \mathbb{M} \oplus o$, being the maximum admitted length for $|\mathbb{M}|$ the bit length of the output from the chosen hash function H. $\mathbb{M}'$ can be attached to an extra metadata parameter of a transaction, as an additional byte array in the output's scriptPubKey or as an *OP_RETURN OP_PAYID* script in a 0-value output from the transaction.

If Alice wants to anonymously spend private coins to fund the transaction, she will need to construct and attach as inputs a set of spend proofs for each of the outputs she wants to spend.

Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge and Dominique Schrder published a paper [5] describing a cryptographic denial-of-spending attack against the original Zerocoin protocol where it would be possible to block a transaction from being propagated in blocks and reusing its serial number S to create a new Zerocoin mint. If this new Zerocoin mint is spent earlier than the honest coin, the honest coin's serial number would be marked as spent making the honest coin thus unspendable.

They propose to "use (as a serial number) a fresh verification key of an ordinary signature scheme, which is strongly existentially unforgeable under chosen message attacks. The spender will additionally sign spend transactions under this verification key, and verifiers will additionally verify these signatures using the verification key revealed as serial number." [5]

This solution, already implemented in other cryptocurrencies, is not completely compatible as it is with the changes in the Zerocoin protocol proposed in this paper, as the coin's serial number is calculated by the sender in zero knowledge.

As an alternative we propose the following scheme to achieve serial number unforgeability:

- When computing a coin spend proof for a transaction's input, we consider S a private key and provide the serial number's public key $\mathbb{S}$ instead as in

$$\mathbb{S} = g^S \pmod{p}$$

- Alice will also include an extra zero knowledge proof of knowledge based on a Schnorr identification protocol [9] transformed in a non-interactive signature of knowledge using the Fiat-Shamir heuristic [10]:

$$\text{ZKSoK}[m]\{(S) : \mathbb{S} = g^S\}$$

This scheme removes an attacker's ability to reuse a serial number to mint a new coin and later proceed with a Denial-Of-Spending attack, as even if he could mint a new coin with the serial number public key $\mathbb{S}$, he'd be unable to spend it without knowledge of the serial number private key S.

Further modification of the Spend algorithm is required to accommodate a new transaction's value commitment $\mathcal{W}$.

$$\mathcal{W} = g^{\mathfrak{w}} g_1^r = g^{\mathfrak{w}} g_1^{(k_1\chi + k_2)} \pmod{p}$$

The description of the original algorithm in [4, Appendix B] defines π as a signature of knowledge "composed of two proofs that (1) a committed value c is accumulated and (2) that c is a commitment to S". A prover using our implementation will need to extend (1) with an extra proof of the accumulation of ϵ in V using the accumulation witness $w\prime$, and substitute (2) with a new proof to prove in zero knowledge that he knows the secrets of both c and ϵ, that ϵ commits to c as an exponent of h and that $\mathcal{W}$ and ϵ commit to the same transaction amount w:

$$\pi = ZKSoK[m]\{(c, w, S, r, v, \sigma) :$$

$$\text{AccVerify}((N, u), A, c, w) = 1 \wedge \text{AccVerify}((N, u), V, \epsilon, w\prime) = 1 \wedge$$

$$\mathbb{S} = g^S \wedge c = \mathbb{S}h^r \wedge \epsilon = g^{\mathfrak{w}} h^c g_1^{\sigma} \wedge \mathcal{W} = g^{\mathfrak{w}} g_1^r\}$$

As a quick draft, we propose the following protocol in order to produce a proof to mathematically convince a verifier of the aforementioned statement. Taken $y = \vartheta^c \beta^v = \vartheta^{(g^S h^r)} \beta^v$ and $\mathcal{Y} = \vartheta^{\epsilon} \beta^{\varsigma} = \vartheta^{(g^{\mathfrak{w}} h^{\sigma} g_1^c)} \beta^{\varsigma}$ from the transcripts of the AccVerify algorithm (used to prove the accumulation of c and ϵ in the accumulators A and V as described in [14]), let a and b be generators of a group whose order equals the modulus of the group used for the Pedersen Commitment c. Let $v\prime \leftarrow \mathbb{Z}_n^*$, $\varsigma\prime \leftarrow \mathbb{Z}_n^*$, $y\prime = a^{(g^S h^r)} b^{v\prime}$ and $\mathcal{Y}\prime = a^{(g^{\mathfrak{w}} h^{\sigma} g_1^c)} b^{\varsigma\prime}$. Using standard and well known techniques, Alice will first prove with a discrete log equality proof that both y and $y\prime$, and $\mathcal{Y}$ and $\mathcal{Y}\prime$, open to the same values.

Then, inspired by the double discrete log proof described in [4, Appendix B], Alice will prove she knows how to open $y\prime$, $\mathcal{Y}\prime$ and $\mathcal{W}$ and will reuse the challenges from the zero knowledge proof to argue for the fulfilment of the rest of conditions:

- She will compute for each $1 \leq i \leq l$:

$$\rho_i, \tau_i, \alpha_i, \gamma_i \in \mathbb{Z}_q$$

$$\zeta_i, \varphi_i, \varpi_i \in \mathbb{Z}_n$$

$$t_i = a^{(\mathbb{S}h^{\rho_i})} b^{\zeta_i}$$

$$\upsilon_i = a^{(g^{\tau_i} h^{\gamma_i} g_1^{\alpha_i})} b^{\varpi_i}$$

$$\mu_i = g^{\tau_i} g_1^{\rho_i}$$

$$\kappa_i = a^{\gamma_i} b^{\varphi_i}$$

$$\omega = \mathrm{H}(m||y||y\prime||a||b||g||h||g_1||\mathcal{W}||\mathbb{S}||t_1|| \ldots ||t_l$$
$$||\upsilon_1|| \ldots ||\upsilon_l||\mu_1|| \ldots ||\mu_l||\kappa_1|| \ldots ||\kappa_l)$$

- For every bit $\omega[i]$, when its value equals 0, let

$$\xi_i = \rho_i$$

$$\iota_i = \tau_i$$

$$\delta_i = \alpha_i$$

$$\psi_i = \zeta_i$$

$$\nu_i = \gamma_i$$

$$\Omega_i = \varpi_i$$

$$\eta_i = \varphi_i$$

  If $\omega[i]$ equals 1, let

$$\xi_i = \rho_i - r$$

$$\iota_i = \tau_i - \mathfrak{w}$$

$$\delta_i = \alpha_i - \sigma$$

$$\psi_i = \zeta_i - \upsilon\prime h^{(\rho_i - r)}$$

$$\nu_i = \gamma_i - c$$

$$\Omega_i = \varpi_i - \varsigma\prime g^{(\tau_i - \mathfrak{w})} h^{(\gamma_i - c)} g_1^{(\alpha_i - \sigma)}$$

$$\eta_i = \varphi_i - \upsilon\prime$$

- The proof

  $(\omega, \xi_1, \ldots, \xi_l, \iota_1, \ldots, \iota_l, \delta_1, \ldots, \delta_l, \psi_1, \ldots, \psi_l, \nu_1, \ldots, \nu_l, \Omega_1, \ldots, \Omega_l, \eta_1, \ldots, \eta_l)$

  is sent to the verifier.

- For every $\omega[i]$ he will check if it equals 0. In that case, let

$$t\prime_i = a^{(\mathbb{S}h^{\xi_i})} b^{\psi_i}$$

$$\upsilon\prime_i = a^{(g^{\iota_i} h^{\nu_i} g_1^{\delta_i})} b^{\Omega_i}$$

$$\mu\prime_i = g^{\iota_i} g_1^{\xi_i}$$

$$\kappa\prime_i = a^{\nu_i} b^{\eta_i}$$

  otherwise

$$t\prime_i = y\prime^{(h^{\xi_i})} b^{\psi_i}$$

$$\upsilon\prime_i = \mathcal{Y}\prime^{(g^{\iota_i} h^{\nu_i} g_1^{\delta_i})} b^{\Omega_i}$$

$$\mu\prime_i = \mathcal{W} g^{\iota_i} g_1^{\xi_i}$$

$$\kappa\prime_i = y\prime a^{\nu_i} b^{\eta_i}$$

- He can now compute

$$\omega\prime = \mathrm{H}(m||y||y\prime||a||b||g||h||g_1||\mathcal{W}||\mathbb{S}||t\prime_1||\ldots||t\prime_l$$

$$||\upsilon\prime_1||\ldots||\upsilon\prime_l||\mu\prime||\ldots||\mu\prime||\kappa\prime_1||\ldots||\kappa\prime_l)$$

- The proof is valid iff $\omega \equiv \omega\prime$.

We point the interested reader to [13, Appendix A] in order to find a full security proof of the original zero knowledge proof which served as an inspiration to construct this.

This proof clearly increases the communication overhead compared with the original proof. Considering $\mathbb{A}$ the size of an accumulation proof, $\mathbb{E}$ the size of a discrete logarithm equality proof and $\mathbb{C}$ the size of a challenge used in the double logarithm proof, a transaction's input communication cost of the original protocol can be approximately denoted as

$$W = \mathbb{A} + \mathbb{E} + 2l\mathbb{C}$$

while the cost of the cryptographic proofs for an input in our proposal would be

$$W = 2\mathbb{A} + 2\mathbb{E} + 7l\mathbb{C}$$

For a default $l = 80$, let $e = \frac{2+2l}{4+7l} = \frac{162}{564} \approx \frac{1}{4}$ the efficiency of our implementation, we can use

$$(\sum_{i=0}^{z} a_i \geq e^{-1}) \stackrel{?}{=} \text{True}$$

to determine if this protocol has a communicational cost advantage for a concrete transaction. Even if our proposal offers better anonymity properties and

shows itself more efficient than Zerocoin transactions with 4 or more inputs, we strongly encourage research in the direction of designing more efficient zero knowledge proofs.

The following table shows the count of single- and multi-exponentiation operations needed to construct and verify the different cryptographic proofs which are part of the coin spend algorithm. Count of scalar arithmetic operations, multiplicative inverse calculations, hash functions or other operations out of the exponentiation realm are intentionally excluded from the scope of the table for simplicity, as their computational cost is considered marginally low.

Table 1: Count of operations of exponentiations of n powers

		n=1	n=2	n=3
Accumulation Proof	Prove	1	8	2
	Verify	0	0	7
DL Equality Proof	Prove	0	2	0
	Verify	0	0	2
Ext. Double DL Proof	Prove	2l	4l	2l
	Verify	l	4l	l
Coin Serial Signature	Prove	2	0	0
	Verify	2	0	0

5 A transaction's amount signature

We substitute the *public* amounts from transactions with secret values hidden in the coin and spend proof commitments. The amounts being publicly verifiable is a key part of how traditional blockchains work to confirm all value transfers occur inside of a constrained money supply limit and that no user is able to spend more coins than those he proved ownership of.

For a transaction $\mathcal{T}$ with m inputs and n outputs we will also require the transaction fee (following strict network policies) to appear explicit as the last output at index n with transparent amount f. This output can be denoted with a special un-spendable script like *OP_RETURN OP_FEE*.

Once the explicit-fee output is added to the output's array of the transaction, Alice will be able to sign the transaction using the public key $\mathcal{N}$ as in

$$\mathcal{N} = \frac{\prod_{i=0}^{m} \mathcal{W}_i}{g^f \prod_{i=0}^{n-1} \epsilon_i h^{-c_i}} = \frac{g^{(\mathfrak{w}\prime_0 + \cdots + \mathfrak{w}\prime_m)} g_1^{(r_0 + \cdots + r_m)}}{g^{(f + \mathfrak{w}_0 + \cdots + \mathfrak{w}_{n-1})} g_1^{(\sigma_0 + \cdots + \sigma_{n-1})}} \pmod{p}$$

only if the committed amounts in $\mathcal{W}_i$ and the committed amounts in $\epsilon_i + f$ match

$$\sum_{i=0}^{m} \mathfrak{w}\prime_i - f - \sum_{i=0}^{n-1} \mathfrak{w}_i \stackrel{?}{=} 0$$

by using

$$\sum_{i=0}^{m} r_i - \sum_{i=0}^{n-1} \sigma_i \pmod{q}$$

as a private key.

6 Validating transactions

Bob will scan all the incoming new transactions (as he already does) and for every output containing a Zerocoin mint, he will:

1. Reject the transaction if:
   - The range proof for the outputs' amount is not valid or
   - The fee is not explicitly included or does not strictly meet the network policies or
   - Broadcasted values c and ϵ are not prime numbers or in the required range or
   - The transaction is not signed by $\mathcal{N}$.
2. Extract b, z_1 and z_2 from his own PK_{view}.
3. Calculate a Diffie-Helman secret $\chi\prime$ using his own EC private key b and Alice's EC public key A.

$$\chi\prime = \mathrm{H}_s(bA) \pmod{q}$$

4. Derive $\sigma\prime$ and $\varrho\prime$ from $\chi\prime$.

$$\sigma\prime = \mathrm{H}(\chi\prime) \pmod{q}$$

$$\varrho\prime = \mathrm{H}(\sigma\prime) \pmod{2^u - 1}$$

5. Decode the transaction amount into $\mathfrak{w}\prime$.

$$\mathfrak{w}\prime = \mathfrak{W}\prime \oplus \varrho\prime$$

6. Reconstruct $\epsilon\prime$ and $c\prime$.

$$c\prime = z_1^{\chi\prime} z_2 \pmod{p}$$

$$\epsilon\prime = g^{\mathfrak{w}\prime} h^{c\prime} g_1^{\sigma\prime} \pmod{p}$$

7. Iff $c\prime$ and $\epsilon\prime$ equals the values of c and ϵ submitted by Alice, Bob recognises the output as spendable and securely stores it, so he can later calculate the spend proofs.

As an improvement to the original specification, Bob or an accountant will be able to reconstruct his whole transaction history of private coins by simply using his private view key PK_{view} with very low computing costs. He will need to keep PK_{view} on memory to verify outputs and calculate an unspendable private coin *pc*. This is considered safe, as an adversary accessing the memory resources of Bob's system won't be able to steal the funds. Only when a Spend action is performed, the private spend key PK_{spend} is unencrypted and temporarily stored in memory while the proofs are constructed, reducing the likeliness of an unauthorised access to the coins in the same manner as in the regular spending of Bitcoin occurs.

However compromising PK_{view} from the memory space of the wallet, or compromising access to the wallet's database local file, would entirely compromise the privacy and act as a source of evidence for an adversary as he would be able to undoubtedly identify previous and future transactions. We encourage to implement full encryption for the whole wallet database to prevent those leakages.

Acknowledgement

We would like to specially thank Samuel Dobson, Guy Kloss, the Veil development team, Jonathan Cressman and Sarang Noether for reviewing the soundness of this paper and providing their constructive input. Marcus Chan for reviewing the copywriting of this paper. Craig MacGregor for coordinating reviews and overseeing the production of this paper. Please note that reviewers of this paper have not been commercially engaged, nor should their review of this paper be considered an endorsement of the papers content or imply any liability whatsoever regarding the application of the private transaction methods the paper describes.

References

[1] *Zcoin.* https://zcoin.io

[2] *PIVX.* https://pivx.org

[3] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system, 2009.* 2012. http://www.bitcoin.org/ bitcoin.pdf

[4] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin: *Zerocoin: Anonymous Distributed E-Cash from Bitcoin.* http://zerocoin.org/media/pdf/ZerocoinOakland.pdf

[5] Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge, Dominique Schrder: *Burning Zerocoins for Fun and for Profit A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol.* https://www.chaac.tf.fau.de/files/2018/04/attack-cryptocur.pdf

[6] Jens Groth, Markulf Kohlweiss. *One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin* https://eprint.iacr.org/2014/764.pdf

[7] *zPOS / zPIV Staking Rewards* https://www.reddit.com/r/pivx/comments/82w7s0/

[8] The NIX Developer Team: *Pedersen Anonymous Deposits: Commitment Key Packs* https://nixplatform.io/wp-content/uploads/2018/10/Commitment_Key_Packs_v1-0-1.pdf

[9] Claus P. Schnorr. *Efficient signature generation for smart cards.* Journal of Cryptology, 4(3):239252, 1991.

[10] Amos Fiat and Adi Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems.* CRYPTO 1986: pp. 186-194

[11] Christina Garman, Matthew Green, Ian Miers, and Aviel D. Rubin *Rational Zero: Economic Security for Zerocoin with Everlasting Anonymity.* https://www.ifca.ai/fc14/bitcoin/papers/bitcoin14_submission_12.pdf

[12] Pedersen T.P. (1992) *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing.* In: Feigenbaum J. (eds) Advances in Cryptology CRYPTO 91. CRYPTO 1991. Lecture Notes in Computer Science, vol 576. Springer, Berlin, Heidelberg

[13] Ian Miers. *Decentralized Anonymous Payments.* 2017

[14] J. Camenisch and A. Lysyanskaya, *Dynamic accumulators and application to efficient revocation of anonymous credentials.* in CRYPTO 02, 2002, pp. 6176.

[15] J. Camenisch and M. Stadler, *Efficient group signature schemes for large groups.* in CRYPTO 97, vol. 1296 of LNCS, 1997, pp. 410424.

[16] Bunz, B., Bootle, J., Boneh, D., Poelstra, A., Maxwell, G.: *Bulletproofs: short proofs for confidential transactions and more.* Cryptology ePrint Archive, Report 2017/1066 (2017).

[17] Greg Maxwell, *Confidential Transactions* https://people.xiph.org/g̃reg/confidential_values.txt