Zyop Technology

WHITE PAPER

Version 1.0

CONTENTS

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Introduction

1. Introduction

We propose Zyop, a new anonymous transaction and payment system which ensures privacy and anonymity with small proof sizes, short verification times and best of all does not require a trusted setup.

1.1. Abstract

Inspired by the Zerocoin protocol, Zyop extends and improves on the original Zerocoin fuctionallity to support confidential transactions. Zyop also greatly improves the original Zerocoin performance and maintaining smaller proof sizes. Zyop builds on the base techniques of Confidential Transactions, Zerocoin and One-out-of-Many proofs and its efficiency is ideal for enabling private and secure blockchain transactions with minimal trust.

1.2. Background

For blockchain/cryptocurrency payments to be truly private transactions have to have two properites.

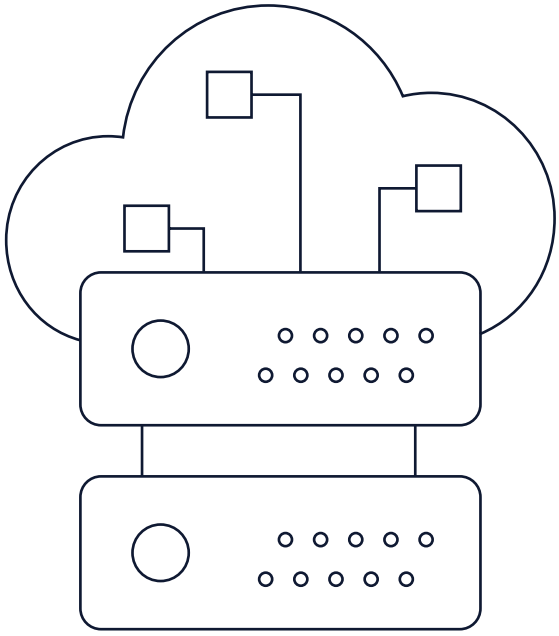
Confidentiality (hiding transferred amounts)
Anonymity (hiding identities of the sender/receiver)

To address the lack of confidentiality in transactions, Gregg Maxwell introduced the concept of Confidential Transactions (CT) in which all amounts are hidden from public view using a commitment amount. This design, however, does not ensure transaction anonymity, a highly desirable privacy feature for financial transactions. Other cryptographic constructions which do offer transaction anonymity, such as the Zerocoin and Zerocash protocols, bring with them significant drawbacks. Zerocoin enables users to generate coins with no prior transaction history which can then be spent anonymously without disclosing the source.

However, this construction works only with fixed denominated coins and hence does not hide transaction amounts. Zerocash provides a very efficient private transaction system which is capable of hiding transaction values, their origins, and destinations. Its hard-to-beat efficiency and advanced privacy features, though, come at the price of reliance on knowledge of exponent assumptions and a trusted setup process, necessitating the user's trust in the correctness of this setup.

1.3. Goal

The goal of this paper is to provide a practical transaction scheme ensuring both anonymity and confidentiality, based on an efficient implementation which relies only on standard cryptographic assumptions and does not require a trusted setup process.



CHAPTER 2

Technologies

2. Technologies

2.1. What is Zyop?

Our work builds upon the Confidential Transaction protocol of Greg Maxwell, the Zerocoin Protocol. We achieve transaction anonymity with a Zerocoin setup which is implemented through one-out-of-many proofs over generalized Pedersen commitments. Except for the coin unique serial number, which is explicitly revealed during the spend operation in order to prevent the double-spending of the coin, the commitment will also hide the monetary value assigned to that coin. The user will be able to sum up the transactions using the homomorphic properties of the underlying commitment scheme. Next, we provide a transaction balance proof which ensures that the transaction's input and output values add up and no coins are generated out of thin air. The main challenge in this type of setup is that the transaction input commitments which help to provide a balance proof (as it is done in CTs), cannot be explicitly exposed. We have observed that one-out-of-many proofs used to generate the proofs of valid spends without revealing the transaction origins already encode necessary information about the coin values in order to generate a zero-knowledge balance proof. The resulting scheme has numerous advantages over the original Zerocoin protocol:

- **It does not require a trusted setup process and is still based on standard cryptographic assumptions**
- **The need for fixed denominations is removed. In fact, the protocol allows mints of arbitrary amounts and partial spends of any amount less or equal to the amount minted.**
- **Transaction amounts are hidden.**
- **A single transaction can contain simultaneous spends and output multiple coins.**
- **Reduction of proof sizes and proof generation times.**
- **It enables efficient batching of the verification of transaction proofs.**
- **Enables direct payments to recipients.**

2.2. Comparisons

The below table illustrates how Zyop compares with other mainstream cryptocurrency protocols, namely with Monero, Zerocoin and Zerocash.

	Anon Set Size	Trusted Setup	Proof Size (KB)	Proof Time (s)	Verification Time (s)
Monero	10	**No**	2.1	1	47
Zerocash	2^32	Yes	0.3	1 - 20	8
Zerocoin	10,000	Yes	25	0.2	200
Zyop	2^14	**No**	1.5	1.5	13

Zyop markedly improves the original Zerocoin protocol's performance while enabling a key feature for anonymous transactions: confidentially transferring arbitrary amounts. From the table above it becomes evident that the proof sizes and verification times of SNARK-based constructions are hard to beat. This unmatched performance, however, is dearly bought with having to rely on knowledge of exponent assumptions. Zyop provides strong privacy and competitive performance while still relying on standard cryptographic assumptions.

2.4. Coverage

A non-interactive commitment scheme is a pair of probabilistic polynomial time algorithms $(G;Com)$. All algorithms in our schemes get a security parameter λ as input written in 1^λ. The setup algorithm $ck \leftarrow G(1^\lambda)$ generates a commitment key ck which specifies the message space Mck, a randomness space Rck and a commitment space Cck. The commitment algorithm combined with the commitment key specifies a function $Comck : Mck \times Rck \rightarrow Cck$. Given a message $m \in Mck$ the sender picks uniformly at random $r \; R \leftarrow Rck$ and computes the commitment $C = Comck(m;r)$. We require the commitment scheme to be both hiding and binding. Informally, a non-interactive commitment scheme $(G;Com)$ is hiding if a commitment does not reveal the committed value. Formally, we require for all probabilistic polynomial time state full adversaries A

$$Pr(ck \leftarrow G(1^\lambda);(m_0,m_1) \leftarrow A(ck);b \leftarrow\{0,1\};c \leftarrow Comck(mb) : A(c) = b) \approx 1/2$$

where A outputs $(m_0,m_1) \leftarrow Mck$. If the probability is exactly 1/2 we say the commitment scheme is perfectly hiding. A non-interactive commitment scheme (G;Com) is strongly binding if a commitment can only be opened to one value. Formally we require

$$Pr(ck \leftarrow G(1^\lambda);(m_0,r_0,m_1,r_1) \leftarrow A(ck) : (m_0,r_0)\ b= (m_1,r_1) \wedge Comck(m_0;r_0) = Comck(m_1;r_1)) \approx 0$$

where A outputs $(m_0,m_1) \leftarrow Mck$ and $(r_0,r_1) \leftarrow Rck$. If the probability is exactly 0 we say the commitment scheme is perfectly binding. The Pedersen commitment scheme is perfectly hiding and computationally strongly binding additively homomorphic commitment scheme under the discrete logarithm assumption. The key generation algorithm G outputs a description of a cyclic group G of prime order p and random generators g and h. The commitment key is $ck = (G,p,g,h)$. To commit to $m \in Zq$ the committer picks randomness $r \in Zp$ and computes $Comck(m;r) = g^m \cdot h^r$. The Pedersen commitment scheme can be generalized for multiple messages, i.e. given messages $m_1,m_2,...,m_n$ one can create a single commitment of the form

$$Com(m_1,m_2,\cdots m_n;r) = hrgm_1\ 1\ gm_2\ 2\ \cdots gm_n\ n$$

In our protocol, we use a private case of generalized Pedersen commitment scheme referred as doubleblinded commitment which utilize three different group generators g,h_1,h_2 and uses two different random factors r_1,r_2 for committing to the given message m as $Commck(m;r_1,r_2) = gmhr_1\ 1\ hr_2\ 2$
Generalized Pedersen commitment scheme is computationally strongly binding, perfectly hiding and has homomorphic properties. In particular, for all correctly generated parameters the following equation holds

$$Commck(m;r_1,r_2) + Commck(m_0;r_0\ 1,r_0\ 2) = Commck(m + m_0;r_1 + r_0\ 1,r_2 + r_0\ 2)$$

Note: We will henceforth refer to the Pedersen commitment for value m using randomness r as $Com(m;r)$. A double-blinded commitment for value m using random values r_1 and r_2 is denoted as $Comm(m;r_1,r_2)$.

2.5. 1-out-of-N (Σ) Proofs for a Commitment Opening to 0

A Σ-protocol is a special type of 3-move interactive proof system that allows a prover to convince a verifier that a statement is true. In the first move the prover sends an initial message to the verifier, then the verifier picks a random public coin challenge $x \leftarrow 1\lambda$ and next, the prover responds to the challenge. Finally, the verifier takes the initial message, the challenge, and the challenge response to check the transcript of the interaction and decide whether the proof should be accepted or rejected.

Jens Groth provided a Σ-protocol for knowledge of 1-out-of-N commitments $C_0,...,C_{N-1}$ being a commitment to 0, or more precisely a Σ-protocol for the relation

$$R = \{(ck,(C_0;...;C_{N-1});(l,r) \mid \forall i : C_i \in Cck \ \wedge l \in \{0,...,N-1\} \wedge r \in Zp \ \wedge C_l = Comck(0,r))\}$$

This protocol was further optimized to reduce proof sizes and fasten proof generation. We will leverage the construction provided to build a Σ-proof for 1-out-of-N double-blinded commitments opening to 0, which can be formalized as follows:

$$R = \{(ck,(C_0;...;C_{N-1});(l,r1,r2) \mid \forall i : C_i \in Cck \ \wedge l \in \{0,...,N-1\} \wedge r \in Zp \ \wedge C_l = Comck(0,r1,r2))\}$$

- **Perfect Completeness:** If the prover knows a witness w for the statement s then they should be able to convince the verifier. Formally, for any $(s,w) \in R$ we have $Pr[Verify(ck,s,Prove(ck,s,w))) = 1$ meaning that the verifier will accept all valid transcripts.
- **Special honest verifier zero-knowledge (SHVZK):** The Σ-protocol should not reveal anything about the prover's witness. This is formalized as saying that given any verifier challenge x it is possible to simulate a protocol transcript.
- **n-Special Soundness:** If the prover does not know a witness w for the statement, they should not be able to convince the verifier. This is formalized as saying that if the prover can answer n different challenges satisfactorily, then it is possible to extract a witness from the accepting transcripts. For any statement s and from n accepting transcripts $\{a,x_i,z_i\}_{i=1}^{n}$ for the $s \in L_R$ with distinct challenges x_i, the witness w can be extracted $s.t. (s;w) \in R$.

2.6. Bulletproofs

Bulletproofs are a powerful scheme for providing short and aggregatable range proofs.
Formally, let $v \in Zp$ and let $V \in G$ be a Pedersen commitment to v using randomness γ. Then the proof system will convince the verifier that $v \in (0,2^n - 1)$. In other words, the proof system proves the following relation $R = \{g,h \in G,V,n; v,\gamma \in Zp \mid V = g^v h^\gamma \ \wedge v \in (0,2^n-1)\}$
Bulletproofs are interactive protocols which can be made non-interactive by using the Fiat-Shamir heuristic in the random oracle model. We will show how Bulletproofs can work with V being a double-blinded commitment to the value v using two random values γ_1 and γ_2. Or in other words, we will provide a proof system for the following relation.

$$R = \{g,h \in G,V,n; v,\gamma_1,\gamma_2 \in Zp \mid V = g^v h_{\gamma_1}^{1} \ 1 \ h_{\gamma_2}^{2} \ 2 \ \wedge v \in (0,2^n-1)\}$$

The resulting protocol also provides perfect completeness, SHVZK, and witness-extended emulation.

2.7. Generalized Schnorr Proofs

Generalized Schnorr proofs are zero-knowledge arguments for the following relation

$R = \{g,h \in G, y ; s,t \in Zp \mid y = g^s h^t\}$

The protocol is depicted in the diagram below.

Prover(g,h,y,(s,t))		Verifier(g,h, y)
Computes		
$s_0, t_0, \leftarrow_R \mathbb{Z}_p$	u	
$u = g^{s_0} h^{t_0} \in G$	$\longrightarrow$	
	$x \leftarrow \{0,1\}^\lambda$	
	$\longleftarrow$	
$s_1 = s_0 - x \cdot s \in \mathbb{Z}_p$		Accepts if and only if
$t_1 = t_0 - x \cdot t \in \mathbb{Z}_p$	$\xrightarrow{s_1,t_1}$	$u = y^x g^{s_1} h^{t_1}$

Verifying the completeness of the protocol is straightforward. It can be converted into a noninteractive protocol that is secure and special honest-verifier zero-knowledge in the random oracle model using the Fiat-Shamir heuristic.

CHAPTER 3

Zyop

3. Zyop

3.1. Construction

Zyop is a decentralized anonymous payment scheme that allows direct anonymous payments of arbitrary amounts. In this section we provide overview of the underlying building blocks, data structures and algorithms used to construct our confidential payment system.

Coins: A coin is the encrypted representation of abstract value to which we associate the following data:

- A coin value v.
- A public key Q which corresponding witness q is referred as a spending key. When the coin is spent or used as an input to a future join-split transaction, the prover proves his ownership over the coin by signing the transaction by this spending key.
- A coin serial number S, which is generated from the coin's public key Q. The coin's serial number is revealed during the spend in order to prevent double-spending of the coin.
- A coin commitment denoted as C. This commitment is a double-blinded commitment to the coin serial number S using the coin value V and also a randomly generated blinding factor R. The coin commitment appears on the ledger as soon as the coin is minted

Transactions: Zyop is implementing three specific transactions.

- **Mint** Transaction: Mint transaction creates and records on the ledger a new coin with commitment $cm(C)$ and value v.
- **Spend** Transaction: Spend transaction redeems a previously minted coin to transfer the shielded value back to the base cryptocurrency. Spend is the private case of JoinSplit transaction which allows more generic functionality.
- **JoinSplit** Transaction: JoinSplit is used to merge, split or redeem coins. We assume JoinSplit transaction can spend multiple input coins and output multiple new coins and also a transparent public value (which may be 0). A special zero-knowledge proof is provided by the transaction owner convincing the legitimacy of the transaction.

Lists of all minted coin commitments and serial numbers of spent coins. For any given moment

- **CMList** denotes the list of all coin commitments appearing in the Mint and JoinSplit transactions.
- **SNList** denotes the list of all serial numbers revealed after Spend or JoinSplit transactions

3.2. Overview

Zyop can be integrated with any blockchain-based currency, such as Bitcoin. To give a sense on how Zyop works, we outline our construction in four incremental steps starting from the original Zerocoin construction.

Step 1: Transaction anonymity with fixed-value coins.

The Zerocoin protocol is one of the first and most widely-used anonymous payment protocols in cryptocurrencies today. It uses coins of fixed denomination, e.g., 1 BTC. The protocol enables the users to destroy these coins in their possession and redeem a new coin with no prior transaction history. Each coin has a unique serial number assigned, which is revealed in the spending process to prevent double-spending of the coin. The original Zerocoin construction was build on RSA accumulators and redeeming Zerocoin required double-discrete-logarithm proofs of knowledge, which have a size exceeding 20 kB.

A new Zerocoin design has been proposed based on 1-out-of-N proofs which results in smaller proof sizes and faster verification. In this construction, the user's spendable coin C_i is a Pedersen commitment to a secret random serial number S that only he knows the opening of. It leverages the fact that the serial number S can be homomorphically subtracted from all coins from the CMList by multiplying them with $Comck(S;o)^{-1}$ so that the commitment with this serial number S turns into a commitment to 0. When a user wants to spend the coin, he first reveals the coin's serial number S, then form a statement for the 1-out-of-N protocol consisting of the commitments $C_0 \cdot Comck(S)^{-1},...,C_N \cdot Comck(S)^{-1}$ and lastly prove that they know an opening to zero for one of these commitments. To prevent doublespending of the coin, the verifier accepts the proof only if S has not previously been recorded in the SNList. Another important benefit of this construction is that in contrast to existing Zerocoin implementations based on RSA accumulators, it does not rely on a trusted setup process, assuming the commitment parameters ck have been generated in a way that is publicly verifiable and excludes backdoors. The Zerocoin scheme consists of a quadruple of PPT algorithms (Setup, Mint, Spend, Verify) for generating a common setup available to all users, creating new coins, spending the coins while simultaneously providing proof of the validness of the spend transaction, and verifying proofs of spending.

Step 2: Enabling to mint, merge, split and redeem coins of arbitrary values.
Zerocoin makes transaction history private, but does not support payments of arbitrary values. A scheme for Confidential Transaction has been proposed. This scheme hides transaction amounts while preserving the ability of the public network to verify that the transaction entries still add up. The construction utilizes coins as Pedersen commitments of the form $C = g^r h^v$, where v is the transaction amount that the transaction owner is committing to, and r is a secret blinding factor to hide the value. The construction implies each transaction can spend Nold inputs denoted as $C_{i1} = g^{r_{i1}} h^{v_{i1}},...,C_{iNold} = g^{r_{iNold}} h^{v_{iNold}}$ to output Nnew new coins $C_{o1} = g^{r_{o1}} h^{v_{o1}},...,C_{oNnew} = g^{r_{oNnew}} h^{v_{oNnew}}$ and for each such transaction a special proof should be provided claiming that the transaction balance is preserved, i.e. $v_{i1} + \cdots + v_{iNold} = v_{o1} + \cdots + v_{oNnew}$

R is known only to the transaction owner who can provide a balance proof by simply signing the transaction with this private key. The signature can then be verified by all network verifiers, as they can compute the public key from the public input and output commitments. Signature verification will pass only if the transaction balance is preserved. Confidential Transactions also use range proofs to convince the verifier that the transaction outputs are not negative and there will be no value overflow. However this construction is possible only because of the fact that all transaction input commitments $C_{i1},...,C_{iNold}$ are public, while our goal is to ensure both transaction anonymity and confidentiality.

Step 3: Ensuring non-malleability and enabling direct anonymous payments.
To prevent malleability attacks on a spend transaction (e.g., malicious assignment by re-targeting the recipient address of the transaction public output) we generate the coin serial number from the public key Q associated with the coin using a cryptographically secure hash function. The coin spend transaction should be signed with the coin's spending key. When the coin is spent, its corresponding public key Q is published on the blockchain instead of the coin serial number S, which still allows all network verifiers to derive the coin serial number S and verify both the transaction signature and provided 1-out-of-N proof. The fact that each coin has an associated spending key, also allows for the creation of a Diffe-Hellman-like authentication system between the coin owner and the targeted recipient and mint the new coins in way that only the intended recipient will possess the spending key.

Step 4: Performing batch verification of transaction proofs.
In the blockchain application the verifier needs to verify multiple separate 1-out-of-N proofs simultaneously. 1-out-of-N proof verification complexity is linear of the commitment list size N and takes hundreds of milliseconds to verify within a set of few dozen thousand commitments. We will illustrate important batch verification techniques which enables verification of proofs in batches and lowers the average cost of a single proof verification to a few dozens of milliseconds within large commitment sets.

3.3. Transactions

Mint Transactions
The Mint algorithm generates a coin of a given value V and a mint transaction $txmint$.

Inputs:
– Public parameters pp
– Coin value $V \in (0,vmax)$
Outputs:
– Randomly generate the coin spending key q and computes the corresponding public key as $Q = gq$
– Compute the coin's serial number from the public key Q as $S = Hash(Q)$
– Randomly sample a commitment blinding factor R
– Compute a double-blinded coin commitment as $C = Comm(S;V,R) = gShV \cdot hR$

Spend Transactions
The Spend transaction enables a user to redeem previously-minted coins without revealing their origin. Although Spend is a private case of a more generic transaction type JoinSplit, we discuss it beforehand for simplifying the design rationale of JoinSplit. Similar to the Spend transaction in the original Zerocoin construction, for each input coin the transaction owner.

– First reveals the coin serial number S
– Next subtracts S homomorphically from all commitments in the CMlist
– Then provides a zero-knowledge proof of their knowledge of one out of these N-formed commitments opening to 0.

Inputs:
– Input coins $C1,...,Cold$
– Private coin $data n(q1,V1,R1),...,(qold,Vold,Rold)o$ and their corresponding indexes in the CMlist.
Outputs: Redeemed value Vout and Spend transaction - txspend
– For each input coin $C1,...,Cold$:
• Prover proves that he knows an index $l \in (0,..N)$ and the values q, V , R of the coin Cl , so that $S = Hash(gq)$ and $Cl = gShV \cdot hR$ 2 .
This is done via Σ-proofs for a double-blinded commitment opening to 0
– Provides a correct output value proof Proofvalue that the transparent input value corresponds to the sum of all spent input coins.
– Signs the transaction with each spending key qi.

3.4. Conclusion

In this paper we have presented a new private cryptocurrency scheme which meets the requirements of a good privacy protocol, namely a high anonymity set, minimal trust required, scalability, ease of use and implementation. We presented formal security proofs for all cryptographic building blocks utilized in our system leaving the formal proof of the payment system security to be discussed in the extended version of this paper.

CHAPTER 4

The Team

Mykel Zhao

Lead Blockchain Developer

Mykel is the founding lead blockchain developer for the Zyrk Project. With a curious interest in Bitcoin since the release way back in 2009, Mykel has watched and participated in many network launches until the idea of Zyrk was created as a movement to change and improve on features he thought was missing in most competing cryptocurrencies.

Jackson Friel

Software Engineer

Jackson was recruited once development for Zyrk was already well underway, Jackson currently works part time on the project to give a helping hand to Mykel due to him just having a new born addition to his family. We hope to see more of Jackson over the coming years and potentially move in to a full time role.

Oscar Goodwin

Systems Admin/Analyst

Oscar was a bit late to the cryptocurrency world, only finding out about Bitcoin around the Mt Gox crash. Oscar has been a Systems Administrator and Analyst for a few high profile technology companies including a large scale hosting company. Oscar looks after all website, server and hosting related parts of the project.

Wayne Ahl

Cryptography Expert

Wayne has always had an eagerness to solve things, which lead him to cryptography. He first learnt of cryptography by subscribing to the Metzdowd mailing list around the time Satoshi released the first version of Bitcoin. Wayne is responsible for the creation of Zyop - making trustless, anyonymous and secure transactions.

Francisco Ribeiro

Business Development and Marketing

Francisco has been an entrepreneur since a young age, managing and owning multiple businesses around the world. He got involved with crypto currencies in late 2015 and has been involved in multiple projects since including being a co-founder in 2017. He is a creative individual with a passion to solve problems and overcoming difficulties that arise.

Zyop Technology

Written By:
Wayne Ahl
waye@zyrk.io

Zyrk Project
https://zyrk.io

team@zyrk.io

Revision 1.0