Lelantus: Private transactions with hidden origins and amounts based on DDH

Aram Jivanyan

Zcoin
aram@zcoin.io
https://zcoin.io
Version: 2018.12-beta-22

Abstract. We propose a new protocol for private financial transactions on the blockchain which ensures the privacy of both transaction values and their origins without a trusted setup. By combining different techniques from the confidential transaction protocol of Greg Maxwell, the Zerocoin Protocol and One-out-of-Many Proofs of Groth et al., we propose a new system, *Lelantus*[1], that provides both anonymity and confidentiality of transactions.

Keywords: Zero-knowledge Proofs, Confidential Transactions, Zcoin, One-out-of-Many Proofs, Double-blinded commitments, Bulletproofs

1 Introduction

For cryptocurrency payments to be truly private, transactions have to have two properties: (a) confidentiality, i.e., hiding the transferred amounts, and (b) anonymity, i.e., hiding the identities of the sender and/or receiver in a transaction. To address the lack of confidentiality in transactions, Gregg Maxwell [4] introduced the concept of *Confidential Transactions* (CT) in which all transaction amounts are hidden from public view using a commitment to the 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 [13] and Zerocash [11] 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.

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.

Our work builds upon the Confidential Transaction protocol of Greg Maxwell, the Zerocoin Protocol and One-out-of-Many Proofs (Sigma) by Groth and Kohlweiss [2]. 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 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 transaction value. 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

[1] In Greek mythology, Lelantus was one of the younger Titanes who was moving unseen

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.

Organization of the paper: In Section 2 we provide a basic cryptographic background. Next, we describe our idea evolution by providing a naïve confidential transaction protocol, which can hide both the transaction values and origins but is not sufficiently secure. In Section 4 we describe, in detail, the full protocol and corresponding cryptographic building blocks. Section 5 provides a performance analysis of the scheme.

2 Cryptographic Background

2.1 Commitments

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 M_{ck}, a randomness space R_{ck} and a commitment space C_{ck}. The commitment algorithm combined with the commitment key specifies a function $Com_{ck} : M_{ck}$ x $R_{ck} \rightarrow C_{ck}$. Given a message $m \in M_{ck}$ the sender picks uniformly at random $r \xleftarrow{R} R_{ck}$ and computes the commitment $C = Com_{ck}(m; r)$. 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

$$Pr[ck \leftarrow G(1^\lambda); (m_0, m_1) \leftarrow A(ck); b \leftarrow \{0,1\}; c \leftarrow Com_{ck}(m_b) : A(c) = b] \approx \frac{1}{2}$$

where A outputs $(m_0, m_1) \leftarrow M_{ck}$. If the probability is exactly $\frac{1}{2}$ 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

$$Pr[ck \leftarrow G(1^\lambda); (m_0, r_0, m_1, r_1) \leftarrow A(ck) : (m_0, r_0) \neq (m_1, r_1) \wedge Com_{ck}(m_0; r_0) = Com_{ck}(m_1; r_1)] \approx 0$$

where A outputs $(m_0, m_1) \leftarrow M_{ck}$ and $(r_0, r_1) \leftarrow R_{ck}$. If the probability is exactly 0 we say the commitment scheme is perfectly binding.

The Pedersen commitment scheme [6] 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 $m \in Z_q$ the committer picks randomness $r \in Z_p$ and computes $Com_{ck}(m; r) = g^m \cdot h^r$.

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

$$Com(m_1, m_2, \cdots m_n; r) = h^r g_1^{m_1} g_2^{m_2} \cdots g_n^{m_n}$$

In our protocol, we use a private case of generalized Pedersen commitment scheme referred as double-blinded 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 $Comm_{ck}(m; r_1, r_2) = g^m h_1^{r_1} h_2^{r_2}$

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

$$Comm_{ck}(m; r_1, r_2) + Comm_{ck}(m'; r_1', r_2') = Comm_{ck}(m + m'; r_1 + r_1', r_2 + r_2')$$

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 $Comm(m; r_1, r_2)$.

2.2 One-out-of-Many (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 [2] provided a Σ-protocol for knowledge of one out of N commitments $c_0, \ldots, c_N$ being a commitment to 0, or more precisely a Σ-protocol for the relation

$$R = \{(ck, (c_0; \ldots; c_N); (l, r) \quad | \quad \forall i : c_i \in C_{ck} \wedge l \in \{0, \ldots, N-1\} \wedge r \in Z_p \wedge c_l = Com_{ck}(0, r))\}$$

This protocol was further optimized and generalized by [3], so the resulting protocol has smaller proof sizes and lower proof computational time. We, in turn, will make some proprietary modification to this protocol for empowering our confidential protocol. We require Σ-protocol to be complete, sound and zero-knowledge in the following sense: [2]

- **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 $\Pr[\text{Verify}(ck, s, \text{Prove}(ck, s, w))] = 1$ 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. $(s; w) \in R$.

2.3 Bulletproofs

Bulletproofs are a powerful scheme for providing short and aggregatable range proofs.

Formally, let $v \in Z_p$ 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

$$R = \{g, h \in G, V, n; \quad v, \gamma \in Z_p \quad | \quad V = g^v h^\gamma \wedge v \in [0, 2^n - 1]\}$$

Bulletproofs are interactive protocols which can be made non-interactive by using the Fiat-Shamir heuristic in the random oracle model. For the original protocol details, we refer to the paper [5].

In section 4.3, 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.

$$R = \{g, h \in G, V, n; \quad v, \gamma_1, \gamma_2 \in Z_p \quad | \quad V = g^v h_1^{\gamma_1} h_2^{\gamma_2} \wedge v \in [0, 2^n - 1]\}$$

The resulting protocol also provides perfect completeness, SHVZK, and witness-extended emulation as defined in [5].

2.4 Generalized Schnorr Proofs

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

$$R = \{g, h \in G, y \quad ; \quad s, t \in Z_p \quad | \quad y = g^s h^t \quad \}$$

The protocol is depicted in the diagram below.

Prover(g,h,y,(s,t))		Verifier(g,h, y)
Computes		
$s_0, t_0, \leftarrow_R \mathbb{Z}_p$		
$u = g^{s_0} h^{t_0} \in G$	$u \longrightarrow$	
	$\longleftarrow x \leftarrow \{0,1\}^\lambda$	
$s_1 = s_0 - x \cdot s \in \mathbb{Z}_p$		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. We can convert the protocol into a non-interactive protocol that is secure and special honest-verifier zero-knowledge in the random oracle model using the Fiat-Shamir heuristic [10].

3 Idea Evaluation

Here we describe how we can utilize a Sigma protocol's internal design elements to create a Sigma-based Zerocoin scheme capable of hiding transaction values.

3.1 Zerocoin

The Zerocoin protocol is one of the most widely-used anonymous payment protocols in cryptocurrencies today. It enables the users to destroy coins in their possession and redeem a new coin with no prior transaction history. These coins can then be sent anonymously i.e., without revealing the inputs. To prevent double spending, a secret serial number is revealed in the spending process.

A Zerocoin scheme can be obtained from a Σ-protocol for 1-out-of-N commitments containing 0. A user's spendable coin C_i is a commitment to a secret random serial number S that only he knows the opening of. Note that the serial number S can be homomorphically subtracted from all coins used in a statement by multiplying them with $Com_{ck}(S;0)^{-1}$ before computing the proof so that the commitment with this serial number S turns into a commitment to 0. When a user wants to spend the coin, they first reveal the coin's serial-number S, then form a statement for the one-out-of-many protocol consisting of the commitments

$$C_0 \cdot Com_{ck}(S)^{-1}, \ldots, C_N \cdot Com_{ck}(S)^{-1}$$

and lastly prove that they know an opening to zero for one of these commitments. To prevent double-spending of the coin, the verifier accepts the proof only if S has not previously been recorded. An 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, generating coins, generating proofs that a coin was spend to pay for a transaction and verifying proofs of spending.

3.2 Confidential Transactions

Unlike Zerocoin, which makes transaction history private, *Confidential Transactions* (CT) hide transaction amounts, while preserving the ability of the public network to verify that the transaction entries add up. Confidential Transactions are possible due to the cryptographic technique of additively homomorphic commitments. Given two generators, CT builds a Pedersen commitment scheme as $c = g^s h^v$, where s is the secret blinding factor, and v is the transaction amount that the transaction owner is committing to.

Let us assume the transaction spends N_{old} inputs denoted as $\{c_{i1} = g^{s_1} h^{v_{i1}}, \cdots, c_{iN_{old}} = g^{s_{N_{old}}} h^{v_{iN_{old}}}\}$ and a transparent net value V_{IN} to output N_{new} outputs $\{c_{o1} = (g^{s_{o1}} h^{v_{o1}}, \cdots, c_{oN_{new}} = g^{s_{oN_{new}}} h^{v_{oN_{new}}}\}$ at the transaction fee f. If the transaction balance is preserved, the following equation holds

$$v_{i1} + \cdots v_{iN_{old}} = v_{o1} + \cdots + v_{oN_{new}} + f$$

This is equivalent to having

$$c_{i1} \cdot \ldots \cdot c_{iN_{old}} / (c_{o1} \cdot \ldots \cdot c_{oN_{new}} \cdot h^f) = g^S$$

where g^S is a valid public key corresponding to the private key

$$S = (s_{i1} + \cdots + s_{iN_{old}}) - (s_{o1} + \cdots + s_{oN_{new}})$$

known only to the transaction owner. The latter can use this private key to sign the transaction. This signature can then be verified by the network verifiers which compute the public key from the input and output commitments. Signature verification will pass only if the transaction balance is preserved.

Providing this balance proof, however, requires the transaction to explicitly show which input commitments have been utilized in the transaction, breaking the privacy of the transaction's history.

CT also employs range proofs to convince the verifier that the transaction outputs are not negative and there will be no value overflow.

3.3 Background on Hiding Transaction Amounts and Their Origin

Our goal is to hide both, the transaction amounts and origins. We show how we can commit to transaction values like in CT and, at the same time, exploit specific design elements of the one-out-of-many(Σ)-proofs along with the homomorphic properties of the commitment scheme, to prove that the transaction balance is preserved without revealing either the input coin origins or the transaction amounts. Also, we want to enable generic transactions, where a single transaction can contain simultaneous spends and output multiple coins.

Let us assume transaction spends N_{old} input coins $(c_{I1}, \ldots, c_{N_{old}})$ and outputs N_{new} coins $(c_{O1}, \ldots, c_{ON_{new}})$. All coins are of the form $C = g^s h^v$ where v is the coin's hidden value and s is its unique serial number.

For all N_{old} input coins the transaction owner first generates separate Σ-proofs (one out of many proofs) which will prove that each spend is valid. Then, let us see how the transaction owner can provide a transaction balance proof with help of these provided Σ-proofs.

We will describe the Σ-proofs in details in section 4.3, but the important thing to observe is that each such proof already contains information hiding the coin's committed value. This special value provided in the proof transcript is composed as $z_d = vx^n - \sum_{k=0}^{n-1} \rho_k x^k$ which is indeed the blinded version of the transaction value v. After relevant modifications in the original Σ-protocol we can also explicitly reveal the commitments of these blinding factors ρ_k as $Com(0, \rho_k)$. So we can consider that the one out of many proofs provided for the N_{old} input coins incorporate the following values

$$(z^1, z^2, \ldots, z^{N_{old}}) \text{ where } z^t = V_t x^n - \sum_{k=0}^{n-1} \rho_k x^k \text{ for } t \in [1, \ldots, N_{old}]$$
$$Com(0, \rho_k^t) \text{ for } k \in [0, \ldots, n-1] \text{ and } t \in [1, \ldots, N_{old}]$$

Now one can observe that each verifier can perform the following computations

$$A = (C_{O1} \cdot \ldots \cdot C_{ON_{new}})^{x^n} = g^{(s_{O1} + \cdots + s_{ON_{new}})x^n} h^{(v_{O1} + \cdots + v_{ON_{new}})x^n}$$

Here x is the challenge parameter generated for the non-interactive one out of many (Σ) protocols. Later, we will discuss how the prover can generate a single challenge value x for multiple Σ proofs.

Next, the verifier can compute

$$B = Com(0, z^1 + \ldots + z^{N_{old}}) \cdot \prod_{t=1}^{N_{old}} (\prod_{k=1}^{n} Com(0, \rho_k^t)^{x^k}) =$$
$$= h^{(v_{i1} + \cdots + v_{iN_{old}})x^n}$$

If the balance equation holds, then the value A/B will be a valid public key of the form

$$\frac{(C_{o1} \cdot \ldots C_{oN_{new}})^{x^n}}{Com(0, z^1 + \ldots + z^{N_{old}}) \cdot \prod_{t=1}^{N_{old}} (\prod_{k=1}^{n} Com(0, \rho_k^t)^{x^k})} = g^{(s_{o1} + \cdots + s_{oN_{new}})x^n}$$

Now it becomes evident that, along with the Σ-proofs, the prover has to additionally prove the knowledge of the exponent value

$$S = (s_{o1} + \cdots + s_{oN_{new}})x^n$$

To do so, the transaction owner can just sign the transaction with the private key $(s_{o1} + \cdots + s_{oN_{new}})x^n$ and all verifiers will be able to check if the transaction balance is preserved or not without seeing any information about the input coin origins.

Note that the verifier needed the values $Com(0, \rho_k^t)$ explicitly to compute the value B. Although the original proof transcript from [2] does not reveal these values, we can modify the original proofs to output them explicitly as is described in section 4.3.

Unfortunately, without additional measures this approach is insecure. The problem is that coin values v used as a the blinding factor for the commitment are from the small range such as $[0, 2^{64} - 1]$. As the coin serial number s is revealed during the spend to prevent double spending, one can take the revealed value s and then brute-force over all values v, which are used as the opening of the commitment, to identify the original coin. As one can easily identify both the spent coin and its hidden value this way, transaction privacy is broken.

Original Pedersen commitments cannot be used in our required setup and we need a more robust cryptosystem which can ensure the privacy of transactions. We show how double-blinded commitments described in Section 2 can be used to build a private transaction system which can securely hide any transaction's values and history.

4 Confidential Payments Hiding Transaction Values and Origins

We consider a blockchain design where users own coins and issue transaction by spending old coins and creating new coins.In Lelantus, we will exploit coins of the form $C = Comm(S; V, R) = g^S h_1^V h_2^R$ where g, h_1, h_2 are group generators orthogonal to each other, S is the coin serial number , V is the coin value and R is an extra random blinding value. Here the component h_2^R will ensure that it will be computationally unfeasible to identify the commitment given both the values S and V.

We assume each transaction can simultaneously spend $N_{old} \geq 0$ input coins and a net transparent input value $V_{in} \geq 0$ to output $N_{new} \geq 0$ new output coins and a net transparent output value

$V_{out} \geq 0$ at the cost of transaction fee f. Each transaction will be comprised of corresponding spend descriptions, output descriptions and the transaction balance proof.

For each input coin, the corresponding spend description is associated with an instance of Σ-proof and the revealed coin serial number, which allows detection of the double-spending attempts of coins.

An output transfer creates a new coin as a double-blinded Pedersen value commitment to the coin value. Output description contains the new created coin along with a valid range-proof, which ensures that the committed value is not negative.

Let's see that in case $N_{\text{old}} = 0$, we should have $V_{\text{in}} > 0$ and this operation is equivalent to the Mint operation from the original Zerocoin protocol. In case $N_{\text{new}} = 0$, we do not have any new outputs during the transaction and only the transparent output value $V_{\text{out}} > 0$ will be cached out.

In general, we require the special zero-knowledge proof of transaction validity to convince that.

- All N_{old} spend transactions are valid.
- All N_{new} output coins are valid and do not contain any negative values: $\forall j : V_{Oj} > 0$
- The transaction balance is preserved and no value is created out of thin air.

Let us define the input coins of the transaction as

$$C_{I1} = g^{S_{I1}} h_1^{V_{I1}} h_2^{R_{I1}}, \ldots, C_{IN_{old}} = g^{S_{IN_{old}}} h_1^{V_{IN_{old}}} h_2^{R_{IN_{old}}}$$

and the output coins as

$$C_{O1} = g^{S_{O1}} h_1^{V_{O1}} h_2^{R_{O1}}, \ldots, C_{ON_{new}} = g^{S_{ON_{new}}} h_1^{V_{ON_{new}}} h_2^{R_{ON_{new}}}$$

The transaction proof is generated via the following steps:

1. For each input coin
   - Prover proves that he knows an index $l \in [0, ..N]$ and the values S, V, R of the coin C_l , so that $C_l = g^S h_1^V h_2^R$ with the help of Σ-protocol for a double-blinded commitment opening to 0, which in detail description is provided in section 4.1. The process steps are the following
     (a) Prover reveals the serial number S.
     (b) Prover parses the initial set of all commitments $C = (C_0, C_1, .C_{N-1})$ and computes $C_i := C_i \cdot Comm(S, 0, 0)^{-1}$
     (c) Prover provides a non-interactive Σ-proof for a double-blinded commitment opening to 0 for the new set $C_i := C_i \cdot Comm(S, 0, 0)^{-1}$.
2. For each output coin
   - Prover provides a zero-knowledge range proof, showing that the coin does not hide a negative value. This is done with the help of Bulletproofs for double-blinded commitments described in the section 4.3.
3. Prover provides a zero-knowledge proof that

$$V_{IN} + V_{I1} + \ldots + V_{IN_{old}} = V_{OUT} + V_{O1} + \ldots + V_{ON_{new}} + f$$

Next, we will describe step-by-step the following cryptographic primitives. (a) Σ-proof for double-blinded commitment opening to 0, (b) zero-knowledge balance proof and (c) Bulletproof for double-blinded commitments. These building blocks are necessarily enough to compose the proof described above.

4.1 Σ-Protocol for One out of N Double-Blinded Commitments Opening to 0

We now describe a Σ-protocol to prove that the list of N double-blinded commitments $C_{ck} = (c_0, \ldots, c_N)$ includes a commitment to 0. More precisely, we give a Σ-protocol for the following relation

$$R = \{(ck, (c_0; \ldots; c_N), (l, r) \quad | \quad \forall i : c_i \in C_{ck} \wedge l \in \{0, \ldots, N-1\} \wedge v, r \in Z_q \wedge c_l = Comm_{ck}(0, v, r))\}$$

The protocol described below is the modified version of the Σ-protocol of [3]. Assuming that $N = n^m$, the idea behind the Σ-protocol is to prove knowledge of an index l for which the product $\prod_{i=0}^{N} c_i^{\sigma_{l,i}}$ is a double-blinded commitment to 0. Here $\sigma_{l,i} = 1$ when $i = l$ and $\sigma_{l,i} = 0$ otherwise and $\sigma_{l,i} = \prod_{j=0}^{m-1} \sigma_{l_j,i_j}$ where $l = \sum_{j=0}^{m-1} l_j n^j$ and $i = \sum_{j=0}^{m-1} i_j n^j$ are the n-ary representations of l and i respectively. In the protocol, the prover first commits to m sequences of n bits $(\sigma_{l_j,0}, \cdots, \sigma_{l_j,n-1})$ and then proves that each sequence contains exactly one 1. On receiving the challenge x, the prover discloses the elements $f_{j,i} = \sigma_{l_j,i} x + a_{j,i}$ where $a_{j,i}$ are randomly generated and committed by the prover. For each $i \in \{0, \cdots, N-1\}$ the product $\prod_{j=0}^{m-1} f_{j,i_j}$ is the evaluation at x of the polynomial $p_i(x) = \prod_{j=0}^{m-1} (\sigma_{l_j,i_j} x + a_{j,i_j})$. So for $0 \leq i \leq N-1$ we have

$$p_i(x) = \prod_{j=0}^{m-1} \sigma_{l_j,i_j} x + \sum_{k=0}^{m-1} p_{i,k} x^k = \sigma_{l,i} x^m + \sum_{k=0}^{m-1} p_{i,k} x^k$$

The coefficients $p_{i,k}$ are depending on the l and $a_{j,i}$ and can be computed by the prover independently of the challenge value x. All polynomials $p_0(x), \cdots, p_{N-1}(x)$ are of degree $m-1$ except $p_l(x)$. The overall protocol is described in detail below.

$P(gk, crs, (C_0, \ldots, C_{N-1}), l, V, R)$		$V(gk, crs, (C_0, \ldots, C_{N-1}))$
Compute		Accept if and only if
$r_A, r_B, r_C, r_D, a_{j,1}, \ldots, a_{j,n-1} \leftarrow_R \mathbb{Z}_q$		
for $j \in [0, \cdots, m-1]$		
$\quad a_{j,0} = -\sum_{i=1}^{n-1} a_{j,i}$		
$B := Com_{ck}(\sigma_{l_0,0}, \ldots, \sigma_{l_{m-1},n-1}; r_B)$		
$A := Com_{ck}(a_{0,0}, \ldots, a_{m-1,n-1}; r_A)$		
$C := Com_{ck}(\{a_{j,i}(1 - 2\sigma_{l_j,i})\}_{j,i=0}^{m-1,n-1}; r_C)$		
$D := Com_{ck}(-a_{0,0}^2, \ldots, -a_{m-1,n-1}^2; r_D)$		
$For \quad k \in 0, ..., m-1$	$A, B, C, D,$ $\{G_K, Q_K\}_{k=0}^{m-1}$ $\longrightarrow$	
$\quad \rho_k, \tau_k \leftarrow_R \mathbb{Z}_q$		
$\quad G_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}}$		
$\quad$ computing $p_{i,k}$ as is described above		The values
$\quad Q_k = Comm(0, \rho_k, \tau_k)$		$A, B, C, D, G_0, Q_0 \ldots, G_{m-1}, Q_{m-1} \in G$
	$\xleftarrow{x \leftarrow \{0,1\}^\lambda}$	$\{f_{j,i}\}_{j,i=0,1}^{m-1,n-1}, z_A, z_C, z_V, z_R \in \mathbb{Z}_q$
$\forall j \in [0, m-1], i \in [1, n-1]$		
$\quad f_{j,i} = \sigma_{l_j i} x + a_{j,i}$		$\forall j : f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i}$
$z_A = r_B \cdot x + r_A$		$B^x A = Com(f_{0,0}, \ldots, \ldots f_{m-1,n-1}; z_A)$
$z_C = r_C \cdot x + r_D$		$C^x D = Com(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C)$
$z_V = V \cdot x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k$	$f_{0,1}, \ldots f_{m-1,n-1}$	
$z_R = R \cdot x^m - \sum_{k=0}^{m-1} \tau_k \cdot x^k$	z_A, z_C, z_V, z_R $\longrightarrow$	$\prod_{i=0}^{N} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} (G_k \cdot Q_k)^{-x^k} =$ $= Comm(0, z_V, z_R)$

Our protocol differs from the original protocol described in [3] in a few different ways. First, it exploits double-blinded commitments, and thus the transcript reveals two different values z_V and z_R for the two random values used in the commitment. Next, instead of revealing the values G_k as a product of $\prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot Comm(0, \rho_k, \tau_k)$, we split this product and explicitly reveal a pairs of values $G_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}}$ and $Q_k = Comm(0, \rho_k, \tau_k)$. The Q_k values are instrumental for verifying the transaction balance proof.

Next, we will prove the following lemma.
Lemma: The Σ-protocol for knowledge of one out of N double-blinded commitments opening to 0 is perfectly complete. It is (m + 1)-special sound if the commitment scheme is binding. It is (perfect) special honest verifier zero-knowledge if the commitment scheme is (perfectly) hiding.
Proof:To see that the protocol is complete observe that the correctness of the equations $B^x A = Com(f_{0,0}, \ldots, \ldots f_{m-1,n-1}; z_A)$ and $C^x D = Com(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C)$ follows by inspection. The first equation proves that the values of each sequence sum up to one and the second equation proves that each sequence is consisting of bits only.

The correctness of the last verification equation follows from the homomorphic properties of the double-blinded commitments since

$$\begin{aligned}
\prod_{i=0}^{N} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} (G_k \cdot Q_k)^{-x^k} &= \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \Big(\prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot Comm(0, \rho_k, \tau_k) \Big)^{-x^k} = \\
&= \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \Big(\prod_{i=0}^{N-1} C_i^{-p_{i,k} \cdot x^k} \cdot Comm(0, -\rho_k x^k, -\tau_k x^k) \Big) \\
&= \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \Big(\prod_{i=0}^{N-1} C_i^{-\sum_{k=0}^{m-1} p_{i,k} \cdot x^k} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big) \\
&= \prod_{i=0}^{N} C_i^{\sigma_{l,i} x^m} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big) \\
&= C_l^{x^m} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big) \\
&= Comm(0, V \cdot x^m, R \cdot x^m) \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big) \\
&= Comm(0, V \cdot x^m - \sum_{k=0}^{m-1} \rho_k x^k, R \cdot x^m - \sum_{k=0}^{m-1} \tau_k x^k) = Comm(0, z_V, z_R)
\end{aligned}$$

Now let us describe a special honest verifier zero-knowledge simulator that is given a challenge $x \in \{0,1\}^\lambda$. It starts by picking the elements of the response uniformly at random as $B, C, G_1, \ldots G_{m-1}$; $Q_0, Q_1, \ldots Q_{m-1} \leftarrow G$; $f_{0,1}, \ldots f_{m-1,n-1}, z_A, z_C, z_V, z_R \leftarrow Z_q$. Next it computes $f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i}$; $A = Com_{ck}(f_{0,0}, \ldots, f_{m-1,n-1}, z_A) B^{-x}$ and $D = Com_{ck}(f_{i,j}(x - f_{i,j}), z_C) C^{-x}$. The simulator com-

putes G_0 from the last verification equation as

$$G_0 = \frac{Q_0^{-1} \cdot Comm(0, z_V, z_R)}{\prod_{i=0}^{N} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=1}^{m-1} (G_k \cdot Q_k)^{-x^k}}$$

By the DDH assumption, $Q_0, Q_1, \ldots Q_{m-1}, G_1, \ldots G_{m-1}$ in a real proof are indistinguishable from picking random group elements as was done in the simulation. We get independent, uniformly random B, z_V and z_R in both real proofs and simulations. Also in both simulations and real proofs, the elements $f_{0,1}, \ldots f_{m-1,n-1}, z_A, z_C$ and C are independent, uniformly random and uniquely determine the values A, D and $\{f_{0,j}\}_{j=0}^{m-1}$. Finally, G_0 is uniquely determined by the last verification equation in both real proofs and in simulations, so the two are indistinguishable. Observe that two different valid answers $f_{0,1}, \ldots f_{m-1,n-1}, z_A, z_C$ and $f'_{0,1}, \ldots f'_{m-1,n-1}, z'_A, z'_C$ to one challenge would break the binding property of $B^x A$ and $C^x D$ so the simulation is perfect.

Now we prove the protocol is $(m+1)$-special sound. Suppose an adversary can produce $(m+1)$ different accepting responses $\Big((f_{j,i}^{(0)}, z_V^{(0)}, z_R^{(m)}), \ldots, (f_{j,i}^{(m)}, z_V^{(m)}, z_R^{(m)})\Big)$ with respect to $m+1$ different challenges $x^{(0)}, \ldots, x^{(m)}$ and the same initial message. Assume that m > 1. As is described in the original paper [7], it is possible to extract the openings $\sigma_{l_j,i}, a_{j,i}$ for B and A with $\sigma_{l_j,i} \in \{0,1\}$ and $\sum_{i=0}^{n-1} \sigma_{l_j,i} = 1$. This opening will define the index $l = \sum_{j=0}^{n-1} l_j n^j$, as l_j is the index of the only 1 in the sequence $\sigma_{l_j,0}, \ldots, \sigma_{l_j,n-1}$. Following the proof, all answers satisfy $f_{j,i}^{(e)} = \sigma_{l_j,i} x^{(e)} + a_{j,i}$ for $0 \le e \le m$ with overwhelming probability due to the binding property of the commitment scheme. Having the values $\sigma_{l_j,i}$ and $a_{j,i}$, we can compute the polynomials $p_i(x) = \prod_{j=0}^{m-1} \Big(\sigma_{l_j,i} + a_{j,i}\Big)$. Here the value $p_l(x)$ is the only polynomial with degree m in x and we can write the last equation of the protocol as $c_l^{x^m} \cdot \prod_{k=1}^{m-1} G_k'^{x^k} = Comm(0, z_V, z_R)$. The values $G_k'^{x^k}$ are derived from the initial statement and the values $\sigma_{l_j,i}$ and $a_{j,i}$ and the equation holds for all $x^{(0)}, \ldots, x^{(m)}$. Consider the Vandermonde matrix with the eth row given by $\Big(1, x^{(e)}, \ldots, (x^{(e)})^m\Big)$. As all $x^{(e)}$ are district, this matrix is invertible and we can obtain a linear combination $\theta_0, \ldots, \theta_n$ of the rows producing the vector $(0, \ldots, 0, 1)$. Hence we can deduce $c_l = \prod_{e=0}^{m} \Big(c_l^{(x^{(e)})^m} \cdot \prod_{k=1}^{m-1} G_k'^{(x^{(e)})^k}\Big)^{\theta_e} = Comm\Big(0, \sum_{e=0}^{m} \theta_e z_V^e, \sum_{e=0}^{m} \theta_e z_R^e\Big)$, which provides an opening of double-blinded commitment c_l to the plain-text 0 with the randomnesses $V = \sum_{e=0}^{m} \theta_e z_V^e$ and $R = \sum_{e=0}^{m} \theta_e z_R^e$.

4.2 Balance Proof for Transactions with Multiple Spend and Output Transfers

For the N_{old} input coins we will have N_{old} separate Σ-proofs published by the spender each proving the knowledge of a unique double-blinded commitment opening to 0 in the list $C_0, C_1, .C_{N-1}$. As follows from the Σ-proof description, the proof transcripts contain the following elements

$$(z_{V_1}, \cdots, z_{V_{N_{old}}}) \text{ and } (z_{R_1}, \cdots, z_{R_{N_{old}}}) \text{ where } z_{V_t} = V_t \cdot x^N - \sum_{k=0}^{n-1} \rho_k^t x^k \text{ and } z_{R_t} = R_t \cdot x^N - \sum_{k=0}^{n-1} \tau_k^t x^k$$
$$\{Comm(0, \rho_0^t, \tau_0^t), \cdots, Comm(0, \rho_{m-1}^t, \tau_{m-1}^t)\} \qquad \text{for } t \in 1, ..., n$$

To verify the validness of the transaction, the verifier should first check the provided *Sigma*-proofs for each spend transfer and the range-proof for all output transfers. Next, in order to ensure that the transaction balance is preserved, the verifier should go over the following steps

1. Takes all output coins, the net output value V_{OUT}, the transaction fee f, and the *Sigma*-proof challenge value x and computes the following element

$$A := (C_{O1} \cdot \ldots \cdot C_{ON_{new}})^{x^n} \cdot h_1^{(V_{OUT}+f)x^n} =$$
$$= g^{(S_{O1}+\ldots+S_{ON_{new}})x^n} h_1^{(V_{OUT}+V_{O1}+\ldots+V_{ON_{new}}+f)x^n} h_2^{(R_{O1}+\ldots+R_{ON_{new}})x^n}$$

2. Second, taking the transaction net input value V_{IN}, the elements $z_{V1}, \ldots, z_{VN_{old}}, z_{R1}, \ldots, z_{RN_{old}}$ and $\{Comm(0, \rho_k^t, \tau_k^t)\}_{k=0}^{m-1}$ from the corresponding Σ-proof transcripts, the verifier computes the element

$$B := h_1^{V_{IN}x^n} \cdot Comm(0; z_{V1} + \ldots + z_{VN_{old}}, z_{R1} + \ldots + z_{RN_{old}}) \cdot \prod_{t=1}^{N_{old}} (\prod_{k=0}^{m-1} Comm(0; \rho_k^t, \tau_k^t)^{x^k})$$
$$= h_1^{(V_{IN}+V_{I1}+\ldots+V_{IN_{old}})x^n} h_2^{(R_{I1}+\ldots+R_{IN_{old}})x^n}$$

3. It becomes evident that, if the balance transaction holds, the h_1 exponents in A and B will cancel each other out and we will have

$$\frac{A}{B} = g^X h_2^Y$$

where

$$X = (S_{O1} + \ldots + S_{ON_{new}})x^n \text{ and } Y = ((R_{O1} + \ldots + R_{ON_{new}}) - (R_{I1} + \ldots + R_{IN_{old}}))x^n$$

Observe that in order to complete the balance proof, it is sufficient for the prover to provide a generalized Schnorr proof of knowledge of the exponent values X and Y described above which he just attaches to the transaction along with the corresponding Σ-proofs for spends and Bulletproofs for outputs.

To help explain how this transaction mechanism works, we elaborate on two scenarios with different combinations of input and output parameters.

1. There are no spend transfers in the transaction and only a net input value V_{IN} is minted.

Let us assume a scenario when the user mints the net input value into two different coins denoted as $C_{O1} = g^{S_{O1}} h_1^{V_{O1}} h_2^{R_{O1}}$ and $C_{O2} = g^{S_{O2}} h_1^{V_{O2}} h_2^{R_{O2}}$ of values V_{O1} and V_{O2} correspondingly at a transaction cost of f. The transaction will be valid if $V_{IN} = V_{O1} + V_{O2} + f$. In order to prove this, the user should first provide two range proofs $BP1$ and $BP2$, ensuring that neither coin encodes a negative value. Next, they should provide a generalized Schnorr signature which will ensure that the output coins sum up to the net value. As there are no spend transfers in the transactions, we can take $x = 1$ and provide a proof of knowledge of exponents for the following value

$$\frac{C_{O1} \cdot C_{O2} \cdot h_1^f}{h_1^{V_{IN}}} = g^{S_{O1}+S_{O2}} \cdot h_2^{R_{O1}+R_{O2}}$$

Observe that in the case of a single output coin, there is no need to provide the range-proof and the Schnorr signature will be enough to guarantee that the output coin corresponds to the minted

amount and so no value is created out of thin air.

2. There are no output transfers and the input coins are spent to output a net value V_{OUT}.

Let us assume the user spends two input coins denoted as $C_{I1} = g^{S_{I1}} h_1^{V_{I1}} h_2^{R_{I1}}$ and $C_{I2} = g^{S_{I2}} h_1^{V_{I2}} h_2^{R_{I2}}$ and corresponding to the values V_{I1} and V_{I2} at a transaction cost of f. The transaction will be valid only if $V_{OUT} + f = V_{I1} + V_{I2}$. In order to prove this, the user first reveals the input coin serial numbers S_{I1} and S_{I2} and then provides two one-out-of-many proofs $\Sigma 1$ and $\Sigma 2$ of knowledge of the commitment opening to 0 in the corresponding sets $\left(C_1 \cdot g^{-S_{I1}}, \ldots, C_N \cdot g^{-S_{I1}}\right)$ and $\left(C_1 \cdot g^{-S_{I2}}, \ldots, C_N \cdot g^{-S_{I2}}\right)$. Next, the spender provides a generalized-Schnorr signature ensuring that the input coin values sum up to the output net value and the transaction fee. Intuitively, as there are no output transfers in the transaction, the spender does not have to generate any range-proof. The challenge value x in the proofs is generated as the hash of the two initial transcript messages of the $\Sigma 1$ and $\Sigma 2$. The spender should provide a proof of knowledge of the exponent for the following value.

$$\frac{h_1^{(V_{OUT}+f)x^n}}{Comm(0; z_{V1} + z_{V2}, z_{R1} + z_{R2}) \cdot \prod_{k=0}^{m-1} \left(Comm(0; \rho_k^1, \tau_k^1)^{x^k}) \cdot Comm(0; \rho_k^2, \tau_k^2)^{x^k}\right)} = h_2^{-(R_{I1}+R_{I2})x^n}$$

While other scenarios using different combinations of input and output parameters such as transactions without any input net value, transactions without any output net value or transactions having multiple spend and output transfers along with transaction net values are possible, the generic approach of the transaction proof generation described above can address all possible scenarios.

4.3 Bulletproofs for Double-Blinded Commitments

Let $v \in \mathbb{Z}_p$ and $V \in G$ be a double-blinded Pedersen commitment to v using the random values γ_1 and γ_2. The Bulletproof system will convince the verifier that $v \in [0, 2^{n-1}]$. In other words, the proof system proves the following relation

$$L = \{(g, h \in G, V, n; \quad v, \gamma_1, \gamma_2 \in Z_p) : V = g^v h_1^{\gamma_1} h_2^{\gamma_2} \wedge v \in [0, 2^n - 1]\}$$

We make appropriate modifications to the original Bulletproofs protocol to support double-blinded commitments. The steps required for proving the relation are the following.

Prover on the inputs v, γ_1, γ_2 computes

$\mathbf{a_L} \in \{0,1\}^n \text{s.t.} \langle \mathbf{a_L}, \mathbf{2^n} \rangle = v, \quad \mathbf{a_R} = \mathbf{1} - \mathbf{a_L} \in Z_P^n$
$\alpha, \rho \leftarrow Z_P, \mathbf{s_L}, \mathbf{s_R} \leftarrow Z_P^n,$
$A = h_1^{\alpha} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}}$
$S = h_1^{\rho} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}}$
Prover $\rightarrow Verifier : A, S$
Verifier $\rightarrow Prover : y, z \leftarrow \mathbf{Z_P}$

Next, as described in the original paper, two linear vector polynomials $l(x), r(x) \in Z_P^n[X]$ and a quadratic polynomial $t(x) \in Z_P[X]$ are defined as follows:

$$
\begin{aligned}
l(x) &= (\mathbf{a_L} - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X && \in Z_P^n[X] \\
r(x) &= y^n \circ (\mathbf{a_R} + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \mathbf{2}^n && \in Z_P^n[X] \\
t(x) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 && \in Z_P[X]
\end{aligned}
$$

The remaining steps required to complete the proof are:

Prover computes
$\tau_1^1, \tau_2^1, \tau_1^2, \tau_2^2 \leftarrow Z_P^*$
$T_1 = g^{t_1} h_1^{\tau_1^1} h_2^{\tau_2^1}, \quad T_2 = g^{t_2} h_1^{\tau_1^2} h_2^{\tau_2^2} \in G$
Prover sends T_1 and T_2 to verifier
Verifier generates a challenge $x \leftarrow Z_P^*$ and sends it to prover
Prover computes
$\mathbf{l} = l(x) = \mathbf{a_L} - z \cdot \mathbf{1^n} + \mathbf{s_L} \cdot x \in Z_p^n$
$\mathbf{r} = r(x) = \mathbf{y^n} \circ (\mathbf{a_R} + z \cdot \mathbf{1^n} + \mathbf{s_r} \cdot x) \in Z_p^n$
$\widehat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in Z_P$
$\mu = \alpha + \rho \cdot x \in Z_p$
$\tau_x^1 = \tau_1^2 x^2 + \tau_1^1 x + z^2 \gamma_1, \quad \tau_x^2 = \tau_2^2 x^2 + \tau_2^1 x + z^2 \gamma_2 \in Z_p$
Prover sends the values $\mathbf{l}, \mathbf{r}, \mu, \tau_x^1, \tau_x^2, \widehat{t}$ to verifier
Verifier computes:
$h_i' = h_i^{(y^{-i+1})} \in G \text{ for } j \in [1, ..., n]$
$P = A \cdot S^x \cdot \mathbf{g}^{-\mathbf{z}} \cdot (\mathbf{h}')^{\mathbf{zy^n} + \mathbf{z^2 2^n}} \in G$
Verifier checks if
$g^{\widehat{t}} h_1^{\tau_x^1} h_2^{\tau_x^2} \equiv V^{z^2} \cdot g^{\sigma(y,z)} \cdot T_1^x \cdot T_2^{x^2}$
$P \equiv h_1^{\mu} \mathbf{g}^{\mathbf{l}} \mathbf{h}'^{\mathbf{r}}$
$\widehat{t} \equiv \langle \mathbf{l}, \mathbf{r} \rangle \in Z_P$

The blinding vectors $\mathbf{s}_L$ and $\mathbf{s}_R$ ensure that the prover can publish $l(x)$ and $r(x)$ for any $x \in Z_P^*$ without revealing any information about $\mathbf{a}_L$ and $\mathbf{a}_R$. The prover needs to convince the verifier that the constant term of $t(x)$ denoted t_0 is equal to $v \cdot z^2 + \sigma(y,z)$. To do so, the prover commits to the remaining coefficients of $t(x)$, namely $t_1, t_2 \in Z_p$ through double-blinded commitments. It then convinces the verifier that it has a double-blinded commitment to the coefficients of $t(x)$ by checking the value of $t(x)$ at a random point $x \in Z_P$.
In the range proof protocol, the prover transmits l and r, with sizes linear in n. The transfer of l and r can be eliminated using the inner-product argument from the actual paper [5]. To use the inner-product argument observe that verifying $P \equiv h_1^{\mu} g^{\mathbf{l}} \mathbf{h'}^{\mathbf{r}}$ and $\widehat{t} \equiv \langle \mathbf{l}, \mathbf{r} \rangle \in Z_P$ is the same as verifying that the witness l and r satisfies the inner product relation

$$\{(\mathbf{g}, \mathbf{h}' \in G^n, P' = P h_1^{-\mu} \in G, \widehat{t} \in Z_p; \mathbf{l}, \mathbf{r} \in Z_p^n) : P' = g^l h^r \wedge \widehat{t} = \langle \mathbf{l}, \mathbf{r} \rangle\}$$

That is, $P \in G$ is a commitment to two vectors $\mathbf{l},\ \mathbf{r} \in Z_p^n$ whose inner product is $\widehat{t}$.
Therefore, instead of transferring the values $\mathbf{l}, \mathbf{r}, \mu, \tau_x^1, \tau_x^2, \widehat{t}$ to the verifier, the prover can transfer $\mu, \tau_x^1, \tau_x^2, \widehat{t}$ and an execution of the inner product argument described in the original paper.

Lemma: The presented range proof for the double-blinded commitment has perfect completeness, perfect special honest verifier zero-knowledge, and computational witness extended emulation.

Proof: For proving the perfect completeness, let us check that

$$\begin{aligned} V^{z^2} \cdot g^{\sigma(y,z)} \cdot T_1^x \cdot T_2^{x^2} &= g^{v \cdot z^2} h_1^{\gamma_1 \cdot z^2} h_2^{\gamma_2 \cdot z^2} \cdot g^{\sigma(y,z)} \cdot g^{t_1 x} h_1^{\tau_1^1 x} h_2^{\tau_2^1 x} \cdot g^{t_2 x^2} h_1^{\tau_1^2 x^2} h_2^{\tau_2^2 x^2} \\ &= g^{v \cdot z^2 + \sigma(y,z) + t_1 x + t_2 x^2} h_1^{\gamma_1 \cdot z^2 + \tau_1^1 x + \tau_1^2 x^2} h_2^{\gamma_2 \cdot z^2 + \tau_2^1 x + \tau_2^2 x^2} \\ &= g^{t_0 + t_1 x + t_2 x^2} h_1^{\tau_x^1} h_2^{\tau_x^2} \\ &= g^{\widehat{t}} h_1^{\tau_x^1} h_2^{\tau_x^2} \end{aligned}$$

as we have $\widehat{t} = \langle \mathbf{l}, \mathbf{r} \rangle = t_0 + t_1 x + t_2 x^2$ and also $t_0 = \sigma(y,z) + v \cdot z^2$ for all valid witnesses v. The equation $g^{\widehat{t}} h_1^{\tau_x^1} h_2^{\tau_x^2} \equiv V^{z^2} \cdot g^{\sigma(y,z)} \cdot T_1^x \cdot T_2^{x^2}$ is the only place where the verifier uses the committed value V, the double-blinded commitments T_1 and T_2 and the values τ_x^1, τ_x^2.
Also, it is evident that other verification checks as

$$\begin{aligned} A \cdot S^x \cdot \mathbf{g}^{-z} \cdot (\mathbf{h}')^{z \mathbf{y}^n + z^2 \mathbf{2}^n} &= h_1^{\alpha} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}} \cdot h_1^{\rho x} \mathbf{g}^{\mathbf{s_L x}} \mathbf{h}^{\mathbf{s_R x}} \cdot \mathbf{g}^{-z} \cdot (h')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} = \\ &= h_1^{\alpha + \rho x} \cdot \mathbf{g}^{\mathbf{a_L} - \mathbf{1^n} \cdot \mathbf{z} + \mathbf{s_L} \cdot \mathbf{x}} \cdot (\mathbf{h}')^{\mathbf{y}^n \circ (\mathbf{a_R} + \mathbf{s_R x} + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n} \\ &= h_1^{\mu} \mathbf{g}^{\mathbf{l}} (\mathbf{h}')^r \end{aligned}$$

In order to prove the perfect honest-verifier zero-knowledge, let us construct the simulator that produces a distribution of proofs for a given statement $\left(g, h \in G; \mathbf{h}, \mathbf{g} \in G^n; V \in G\right)$ that is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. The simulator chooses all proof elements and challenges according to the randomness supplied by the adversary from their respective domains or computes them directly as described in the protocol. Particularly it generates random witness $\mathbf{l}$ and $\mathbf{r}$, chooses $A, T_2 \leftarrow G, \mu \leftarrow Z_P$ and then computes S

and T_1 according to the verification equations as

$$S = \left(h_1^{-\mu} \cdot A \cdot \mathbf{g}^{-z \cdot \mathbf{1}^n - \mathbf{l}} \cdot \mathbf{h}'^{-z \cdot \mathbf{y}^n - z^2 \cdot \mathbf{2}^n - \mathbf{r}}\right)^{-x^{-1}}$$

$$T_1 = \left(h_1^{-\tau_x^1} \cdot h_2^{-\tau_x^2} \cdot g^{\sigma(y,z) - \widehat{t}} \cdot V^{z^2} \cdot T_2^{x^2}\right)^{-x^{-1}}$$

The simulator can run the inner product argument with the simulated witness **(l,r)** and the verifier randomness. All elements in the proof are either independently randomly distributed or their relationship is fully defined by the verification equations. Revealing the inner product witness or leaking information about it does not change the zero-knowledge property of the overall protocol. It is straightforward to check that the simulator is efficient.

In order to prove the knowledge-soundness of Bulletproofs, the witness-extended emulation techniques defined in [9] and [8] is used which shows that whenever an adversary produces an argument satisfying the verifier with some probability, then there exists an emulator producing an identically distributed argument with the same probability, but also a witness. The emulator is permitted to rewind the interaction between the prover and verifier to any move and resume with the same internal state for the prover, but with fresh randomness for the verifier. Whenever the adversary makes a convincing argument when in state s, the emulator can extract the witness, and therefore, we will have an argument of knowledge of witness w. The techniques to build witness-extended emulation techniques for Bulletproofs with double-blinded committed value are identical to the techniques for building an emulator for Bulletproofs described in the original paper, apart from a minor argument that two random values used in the calculation of the values T_1 and T_2 should be extracted from the transcripts. For the details of the emulator construction process, we refer to the original paper for interested readers.

5 Batching & Precomputation Techniques for One-Out-Of-Many Proofs in the Zerocoin Setup

In the blockchain application the verifier needs to verify multiple separate One-out-of-Many Proofs simultaneously. For example the blockchain nodes receiving a block of transactions needs to verify all transactions and thus the one-out-of-many proofs in parallel. We describe an important optimization concerning to the simultaneous verification of multiple one-out-of-many proofs in the Zerocoin setup.
Lets assume the Verifier has to verify M different Spend transfers from the initial set of all commitments $C = (C_0, C_1, .C_{N-1})$. Each Spend description contains the associated Σ proof and the coin serial number s_t For the t-th Spend a new commitment set is calculated as $C_i^t := C_i \cdot g^{-s_t}$ and the Σ-proof is generated, which convinces the verifier that the set $\{C_0^t, \ldots, C_{N-1}^t\}$ contains a commitment to 0. The verification of single one-out-of-many proof boils down to a large multi-exponentiation to check the following equivalency.

$$\prod_{i=0}^{N} C_i^{t^{\prod_{j=0}^{m-1} f_{j,i_j}^t}} \cdot \prod_{k=0}^{m-1} (G_k^t \cdot Q_k^t)^{-x^k} = Comm(0, z_V^t, z_R^t)$$

This require $\sim N$(as we have $N >> m$) exponentiations. For simplifying the notations further, lets make the following assignments.

$$f_i^t = \prod_{j=0}^{m-1} f_{j,i_j}^t \qquad D_t = \prod_{k=0}^{m-1} (G_k^t \cdot Q_k^t)^{-x^k} \qquad E_t = Comm(0, z_V^t, z_R^t) \tag{1}$$

So for each transaction the multi-exponentiation equation can be rewritten as

$$\prod_{i=0}^{N} \Big(C_i^t\Big)^{f_i^t} \cdot D_t = E_t \tag{2}$$

We want to benefit of batch verification techniques based on the observation that checking if $g^x = 1$ and $g_y = 1$ can be checked by drawing a random scalar α from a large domain and checking if $g^{\alpha x+y} = 1$. But in our case the generators C_i^t used in the multi-exponentiation are proof dependent and differs for each transaction.

In order to benefit of batching techniques we can use the following trick. Considering the fact that $C_i^t = C_i \cdot g^{(-s_t)}$ and the values s_t are explicitly revealed during the Spend transaction, we can rewrite the equation (2) as

$$\prod_{i=0}^{N} \Big(C_i^t\Big)^{f_i^t} \cdot D_t = \prod_{i=0}^{N} \Big(\frac{C_i}{g^{s_t}}\Big)^{f_i^t} \cdot D_t = E_t \tag{3}$$

or equivalently

$$\prod_{i=0}^{N} C_i^{f_i^t} = \frac{E_t}{D_t} \cdot g^{s_t \cdot (\sum_{i=0}^{N} f_i^t)} \tag{4}$$

The Verifier can do this for all proofs, as the the values s_t are public and are part of the Spend proof. So for all M transactions with different s_t, we will get the same generator values C_i for the left side of all M equations. This will enable to

1. Do verification of proofs in batches
2. Make pre-computations for the exponent values of C_i (resulting to another 25-60% performance optimization)

So for verifying all M spend proofs in batch, the verifier can generate M random values $y_1, \ldots, y_t$ and do the following computations

$$\prod_{t=1}^{M} \Big(\prod_{i=0}^{N} C_i^{f_i^t}\Big)^{y_t} = \prod_{t=1}^{M} \Big(\frac{E_t}{D_t} \cdot g^{s_t \cdot (\sum_{i=0}^{N} f_i^t)}\Big)^{y_t} \tag{5}$$

which in turn is equivalent to verifying that

$$\prod_{i=0}^{N} C_i^{\sum_{t=1}^{M} y_t \cdot f_i^t} = g^{\sum_{t=1}^{M} \Big(y_t \cdot s_t \cdot \sum_{i=0}^{N} f_i^t\Big)} \cdot \prod_{t=1}^{M} \Big(\frac{E_t}{D_t}\Big)^{y_t} \tag{6}$$

This helps to save N exponentiation for each extra proof.

N	n	m	Proof Size (bytes)	Proof Time (ms)	Verification Time (ms)
8192	2	13	1564	1636	243
16384	4	7	1412	1464	491
32768	8	5	1724	1895	1002
65536	4	8	1576	6584	1992
65536	16	4	2456	2940	1997
262144	8	6	2016	19218	8315
262144	64	3	6516	10401	8343

Table 1. One-out-of-Many Proofs Performance

6 Implementation and Performance

For a commitment set of $N = n^m$ coins, a single One-out-of-Many Proof requires the prover to send $2m + 4$ Pedersen commitments and $m(n - 1) + 4$ elements of Z_P in total.

The proof can be computed using $mN + 2mn + 2m + 6$ group exponentiation, as the computation of the values A, B, C and D in the bit proof requires $2mn+4$ exponentiation since exponentiation by $(1 - 2\sigma_{l_i,j})$ amounts to a multiplication. Computing the elements Q_k costs $2m$ exponentiation. The computation of all G_k requires mN exponentiation. Proofs can be verified using $N + 2mn + 2m + 15$ group exponentiations as follows: $N + 2m + 2$ exponentiation for the last verification equation and $2mn + 4$ for the remained.Our schemes can be instantiated over any group G where the DDH problem is computationally hard. To evaluate the performance of our proofs, we implemented a reference implementation in C++ over the popular library libsecp256k1 which uses the elliptic curve secp256k1 with 128-bit security and is used in numerous cryptocurrency projects. In the compressed

Batch Size	Verification Time	Marginal cost per verification
5	623	124.6
10	636	63.6
50	1125	22.5
100	1759	17.6
500	6978	14
1000	13719	13.7

Table 2. Batch Verification Timing for the Anonymity Set of 16384

Batch Size	Verification Time	Marginal cost per verification
5	1090	218
10	1186	118.6
50	1970	39.4
100	2967	29.7
500	11098	22.2
1000	21825	21.8

Table 3. Batch Verification Timing for the Anonymity Set of 32384

Batch Size	Verification Time	Marginal cost per verification
5	2162	432.5
10	2317	232
50	3691	73.8
100	5342	53.4
500	19660	39.3
1000	38192	38.2

Table 4. Batch Verification Timing for the Anonymity Set of 65536

Batch Size	Verification Time	Marginal cost per verification
5	9310	1862
10	10024	1000
50	16737	335
100	24995	250

Table 5. Batch Verification Timing for the Anonymity Set of 262144

form, secp256k1 points are stored as 33 bytes. In the table below we bring the proof size and performance parameters for different anonymity set size and configurations. All experiments were performed on an Intel I7-4870HQ system with a 2.50 GHz processor. The verification time has a critical importance for the cryptocurrency application, as the verifiers need to check all confidential transactions with all associated proofs. Next we experiment with batching of the verification of multiple Σ-proofs, so that the cost of verifying every additional proof will be significantly reduced. Our modifications to the original Bulletproofs protocol add an additional two exponentiations to the proof generation efforts and one extra exponentiation to the verification process. Also, we add an additional Z_P element to the proof. These modifications, in general, will result in a negligible performance overhead to the original protocol. For the Bulletproofs performance estimation, we refer to the implementation analysis provided in the original paper. The proof size for a single 64-bit range proof is 675 byte which takes 37ms to generate and 3.9ms to verify. In the transaction with multiple spend and output transfers, the proof and verification times will be dominated by the Σ-proofs and Bulletproofs. The generalized Schnorr proof's impact on the overall transaction performance will be negligible, as the generation requires only two exponentiations and verification needs just three exponentiations. The signature will be comprised of one group element and two scalars which is a small overhead to the overall storage requirements.

Acknowledgments. The author thanks Benedict Bünz for his continuous feedback and important suggestions for the Bulletproofs protocol for double-blinded commitment schemes and Jens Groth for his initial feedback on the generic ideas evaluated in this paper. Further, I thank Poramin Insom, Reuben Yap, Martun Karapetyan, Levon Petrosyan, and the whole Zcoin team for helpful discussions and support. This protocol has been developed in the scope of Zcoin's next-generation privacy protocol research.

References

1. J. Camenisch, A. Kiayias, M. Yung. On the Portability of Generalized Schnorr Proofs. https://eprint.iacr.org/2009/050.pdf
2. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In EUROCRYPT, vol. 9057 of LNCS. Springer, 2015.
3. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., et al. (eds.) ESORICS. LNCS, vol. 9326, pp. 243265. Springer, Heidelberg (2015).
4. Greg Maxwell. Confidential transactions. https://people.xiph.org/greg/confidential-values.txt, 2016.
5. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Report 2017/1066, 2017. https://eprint.iacr. org/2017/1066
6. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO, volume 576 of Lecture Notes in Computer Science, pages 129 -140,1991.
7. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 327-357. Springer, 2016.
8. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology, 16(3):143-184, 2003.
9. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Advances in Cryptology - EUROCRYPT 2008, pages 379-396, 2008.
10. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In ACM CCS, pages 62-73, 1993.

11. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In IEEE Symposium on Security and Privacy. IEEE, 2014.
12. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In CRYPTO, 2013.
13. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In IEEE Symposium on Security and Privacy, 2013.