Lelantus: Towards Confidentiality and Anonymity of Blockchain Transactions From Standard Assumptions

Aram Jivanyan

Zcoin
aram@zcoin.io
Version: 2019.04-beta-07

Abstract. We propose *Lelantus*[1], a new anonymous payment system which ensures both transaction confidentiality and anonymity with small proof sizes, short verification times and without requiring a trusted setup. Inspired by the Zerocoin protocol, Lelantus extends the original Zerocoin functionality to support confidential transactions while also significantly improving on the protocol performance. Lelantus proof sizes are almost 17 times smaller compared to the original Zerocoin proof sizes. Moreover, we show how to support efficient aggregation of the transaction proofs, so that the proof verification, while asymptotically linear, is very efficient in practice. Lelantus builds on the techniques of Confidential Transactions, Zerocoin and One-out-of-Many proofs and its efficiency is particularly well-suited for enabling private blockchain transactions with minimal trust required while employing well-studied cryptographic assumptions.

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 [6] 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 [15] and Zerocash [13] 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.

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

1.1 Our Contribution

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 [4]. We achieve transaction anonymity with a Zerocoin setup which is implemented through one-out-of-many proofs over generalized Pedersen commitments as is discussed in [4]. 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.

1.2 Related Works and Comparison

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

	Anonymity Set Size	Trusted Setup	Cryptographic Assumptions	Proof Size(KB)	Proof Time(s)	Verification Time(ms)
Monero	10	No	Well-tested	2.1	1	47
Zerocash	2^{32}	Yes	Relatively New	0.3	1-20	8
Zerocoin	10,000	Yes	Well-tested	25	0.2	200
Lelantus 1	2^{14}	No	Well-tested	1.5	1.5	13
Lelantus 2	2^{16}	No	Well-tested	1.5	6	38

Table 1. The security properties and efficiency considerations for different privacy solutions.

Lelantus 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. Lelantus provides strong privacy and competitive performance

while still relying on standard cryptographic assumptions. At the same time, it offers significantly better anonymity than Monero while also being more efficient due to the smaller transaction sizes and shorter verification times.

Organization of the paper: In Section 2 we provide a basic cryptographic background. Next, we discuss the underlying building blocks, data structures and algorithms used to construct our confidential payment system. In section 4 we detail all algorithmic constructions used in the protocol. Section 5 describes efficient batching and precomputation techniques for 1-out-of-N proofs to accelerate the verification of transaction blocks. Section 6 shows how Lelantus can support direct anonymous payments on the blockchain. In Section 7 we discuss performance analysis of the scheme and show benchmarks data.

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 [8] 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 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 [4] provided a Σ-protocol for knowledge of 1-out-of-N commitments $C_0, \ldots, C_{N-1}$ being a commitment to 0, or more precisely a Σ-protocol for the relation

$$R = \{(ck, (C_0; \ldots; C_{N-1}); (l, r) \quad | \quad \forall i : C_i \in C_{ck} \wedge l \in \{0, \ldots, N-1\} \wedge r \in Z_p \wedge C_l = Com_{ck}(0, r))\}$$

This protocol was further optimized in [5] to reduce proof sizes and fasten proof generation. We will leverage the construction provided in [4] to build a Σ-proof for 1-out-of-N double-blinded commitments opening to 0, which can be formalized as follows:

$$\begin{aligned} R = \{&(ck, (C_0; \ldots; C_{N-1}); (l, r_1, r_2) \quad | \\ &\forall i : C_i \in C_{ck} \wedge l \in \{0, \ldots, N-1\} \wedge r \in Z_p \wedge C_l = Com_{ck}(0, r_1, r_2))\} \end{aligned}$$

We require Σ-protocol to be complete, sound and zero-knowledge in the following sense: [4]

- **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 [7].

In the Appendix B 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 [7].

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	
$u = g^{s_0} h^{t_0} \in G$	$\longrightarrow$	
	$\xleftarrow{x \leftarrow \{0,1\}^\lambda}$	
$s_1 = s_0 - x \cdot s \in \mathbb{Z}_p$		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. It can be converted into a non-interactive protocol that is secure and special honest-verifier zero-knowledge in the random oracle model using the Fiat-Shamir heuristic [17].

3 Lelantus Construction

Lelantus 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.

3.1 Data Structures and Algorithms

Lelantus is exploiting the following data structures and algorithms.

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. Lelantus 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 and Intuition

Lelantus can be integrated with any blockchain-based currency, such as Bitcoin. To give a sense on how Lelantus 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.
In [4] 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. [4] leverages the fact that the serial number S can be homomorphically subtracted from all coins from the *CMList* by multiplying them with $Com_{ck}(S;0)^{-1}$ 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

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

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 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. In [6], 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 N_{old} inputs denoted as $C_{i1} = g^{r_{i1}} h^{v_{i1}}, \ldots, C_{iN_{old}} = g^{r_{iN_{old}}} h^{v_{iN_{old}}}$ to output N_{new} new coins $C_{o1} = g^{r_{o1}} h^{v_{o1}}, \ldots, C_{oN_{new}} = g^{r_{oN_{new}}} h^{v_{oN_{new}}}$ and for each such transaction a special proof should be provided claiming that the transaction balance is preserved, i.e.

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

By leveraging the additive homomorphic properties of the Pedersen commitment scheme, it is easy to notice that the balance is preserved if the following equation takes place

$$\frac{C_{i1} \cdot \ldots \cdot C_{iN_{old}}}{C_{o1} \cdot \ldots \cdot C_{oN_{new}}} = g^R$$

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

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

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 $C_{i1}, \ldots, C_{iN_{old}}$ are public, while our goal is to ensure both transaction anonymity and confidentiality.

In our construction, coin commitments are double-blinded Pedersen commitments of the form $C = g^S h_1^V h_2^R$, where V is the coin value, S is the coin's unique serial number which is revealed during the spend, and R is an extra blinding factor, which keeps the coins untraceable even after the coin's serial number and value are published. We enable *JoinSplit* transactions allowing to merge and split coins of arbitrary amounts as well redeem the coin values to convert them to the base currency. Each *JoinSplit* transaction can spend N_{old} inputs to make a public output v_{pub} and simultaneously output N_{new} new coins at the transaction fee f. The transaction balance is preserved if and only if

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

We describe how one can still provide a transaction balance proof without revealing either the origin, i.e., the inputs or the transaction values by exploiting specific design elements of the 1-out-of-N proofs. For all transaction input coins $C_j \in C_{i1}, \ldots C_{iN_{old}}$ the transaction owner exercises the zerocoin-style spend operation as described in Step 1. More precisely, they first reveal the coin's serial number S_{ij} and homomorphically subtract it from all coin commitments $CMList = \big(C_1, \cdots, C_N\big)$. They then leverage their knowledge of the commitment's opening values V_{ij} and R_{ij} to provide a special 1-out-of-N proof establishing the legitimacy of this spend. The 1-out-of-N proofs published to the blockchain hide the origins of their corresponding input coin, but at the same time they contain blinded information about the input values. As we will show in Section 4, the transaction owner can exploit these blinded values to generate a balance proof. Verifiers can compute the published spend proofs and explicit transaction outputs to verify this balance proof.

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 Diffie-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.

4 Algorithm Constructions

4.1 Setup

Our construction works for any additively homomorphic non-interactive commitment scheme over Z_p, where p is a large prime. Example of such commitment schemes include Pedersen commitments

[8]. In our case, the commitment key ck specifies a prime-order group G and three orthogonal group generators g, h_1 and h_2. We also utilize a cryptographically secure hash function $Hash$.

Inputs: Security parameter λ
Outputs: Public parameters $pp = (ck, Hash)$

1. $ck = (G, g, h_1, h_2)$
2. Hash function $Hash : \{0,1\}^* \rightarrow \{0,1\}^\lambda$

4.2 Mint Transactions

The $Mint$ algorithm generates a coin of a given value V and a mint transaction tx_{mint}.

Inputs:

- Public parameters pp
- Coin value $V \in [0, v_{max}]$

Outputs: Coin commitment C and mint transaction tx_{mint}

- Randomly generate the coin spending key q and computes the corresponding public key as $Q = g^q$
- 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 $C = Comm(S; V, R) = g^S h_1^V h_2^R$

The component h_2^R will ensure that it will be computationally unfeasible to identify the commitment even when given both the values S and V. Mint transactions can be associated with one or more transparent inputs from the blockchain base layer, and each base layer input spending should be associated with a proof of ownership for the spent assets. This is usually done by providing a signature Sig_i, which can be verified through the input UTXO's public key. $Mint$ algorithm should also prove that the committed value actually corresponds to the transparent input V. This is done by providing $SchnorrProof$, a generalized Schnorr proof of knowledge of discrete logarithm relation $\frac{C}{h_1^V} = g^S h_2^R$. In the end, the tx_{mint} is comprised of the following components $tx_{mint} = \Big\{SchnorrProof, \{Sig_{i1}, \dots, \}, aux_{data}\Big\}$. The transaction owner saves the coin's private data (q, V, R) while the coin's commitment C is published to the blockchain along with the tx_{mint}.

4.3 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 discussed in [4], 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.

In our case, the *Spend* also redeems the coin value V to a transparent input, an extra value proof should be provided convincing that the hidden values of all spent coins sum up to the redeemed value. *Spend* Transaction generates a zero-knowledge proof of a valid spend comprised of the zero-knowledge 1-out-of-N proof and a special value proof along with other auxiliary transaction data.

Inputs:

- Input coins $C_1, \ldots, C_{old}$
- Private coin data $\Big\{(q_1, V_1, R_1), \ldots, (q_{old}, V_{old}, R_{old})\Big\}$ and their corresponding indexes $l_1, \ldots, l_{old}$ in the $CMlist$.

Outputs: Redeemed value V_{out} and *Spend* transaction - tx_{spend}

- For each input coin $C_1, \ldots, C_{old}$:
  - Prover proves that he knows an index $l \in [0, ..N]$ and the values q, V, R of the coin C_l , so that $S = Hash(g^q)$ and $C_l = g^S h_1^V h_2^R$. This is done via Σ-proofs for a double-blinded commitment opening to 0 via the following steps: The Prover
    1. Reveals the coin address public key $Q = g^q$.
    2. Computes the coin serial number as $S = Hash(Q)$.
    3. Homomorphically substracts the coin serial number from all coin commitments in the set $CMList = (C_0, C_1, .C_{N-1})$ resulting to a new set $CMList' = \{C_0 \cdot Comm(S, 0, 0)^{-1}, \ldots, C_{N-1} \cdot Comm(S, 0, 0)^{-1}\}$
    4. Generates a Σ-proof proving the knowledge of one double-blinded commitment from the set $CMList'$ is opening to 0.
- Provides a correct output value proof $Proof_{value}$ that the transparent input value corresponds to the sum of all spent input coins. This is done with help of generalized Schnorr proofs of knowledge and will be described in details below.
- Signs the transaction with each spending key q_i.

The resulted tx_{spend} is comprised of the following elements

$$tx_{spend} = \Big([\Sigma_1, Q_i], \ldots, [\Sigma_{old}, Q_{old}], V_{out}, Proof_{value}, sign_1, \ldots, sign_{old}\Big)$$

Next we 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. This protocol design elements are instrumental for generating value proof in *Spend* transactions and the balance proof in the $JoinSplit$ transactions.

1-out-of-N proofs for a double-blinded commitment opening to 0. Formally, we give a Σ-protocol for the following relation

$$\begin{aligned} R = \{(ck, (C_0, \ldots, C_{N-1}), (l, v, r) \quad | \quad & \forall i : C_i \in C_{ck} \\ & \wedge\, l \in \{0, \ldots, N-1\} \\ & \wedge\, v, r \in Z_q \wedge C_l = Comm_{ck}(0, v, r))\} \end{aligned}$$

The protocol described below, is the modified version of the Σ-protocol of [5]. 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. $\sigma_{l,i} = \prod_{j=0}^{m-1} \sigma_{l_j, i_j}$ 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 $0 \le i \le N-1$ 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 $j \in [0, \cdots, m-1]$		
$\quad a_{j,0} = -\sum_{i=1}^{n-1} a_{j,i}$		
$B := Com_{ck}(\sigma_{l_0,0}, \ldots, \sigma_{l_{m-1},n-1}; r_B)$		
$A := Com_{ck}(a_{0,0}, \ldots, a_{m-1,n-1}; r_A)$		
$C := Com_{ck}(\{a_{j,i}(1-2\sigma_{l_j,i})\}_{j,i=0}^{m-1,n-1}; r_C)$		
$D := Com_{ck}(-a_{0,0}^2, \ldots, -a_{m-1,n-1}^2; r_D)$		
$For \quad k \in 0, ..., m-1$	$A, B, C, D,$	
$\quad \rho_k, \tau_k, \gamma_k \leftarrow_R \mathbb{Z}_q$	$\{G_K, Q_K\}_{k=0}^{m-1}$	
$\quad G_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot h_2^{-\gamma_k}$	$\longrightarrow$	
$\quad$ computing $p_{i,k}$ as is described above		The values
$\quad Q_k = h_2^{\gamma_k} \cdot Comm(0, \rho_k, \tau_k)$		$A, B, C, D, G_0, Q_0 \ldots, G_{m-1}, Q_{m-1} \in G$
	$\xleftarrow{x \leftarrow \{0,1\}^\lambda}$	$\{f_{j,i}\}_{j,i=0,1}^{m-1,n-1}, z_A, z_C, z_V, z_R \in \mathbb{Z}_q$
$\forall j \in [0, m-1], i \in [1, n-1]$		
$\quad f_{j,i} = \sigma_{l_j i}x + a_{j,i}$		$\forall j : f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i}$
$z_A = r_B \cdot x + r_A$		$B^x A = Com(f_{0,0}, \ldots, \ldots f_{m-1,n-1}; z_A)$
$z_C = r_C \cdot x + r_D$		$C^x D = Com(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C)$
$z_V = V \cdot x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k$	$f_{0,1}, \ldots f_{m-1,n-1}$	
$z_R = R \cdot x^m - \sum_{k=0}^{m-1} \tau_k \cdot x^k$	z_A, z_C, z_V, z_R	$\prod_{i=0}^{N} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1}(G_k \cdot Q_k)^{-x^k} =$
	$\longrightarrow$	$= Comm(0, z_V, z_R)$

Our construction differs from the original protocol described in [5] 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 the pair of blinded 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 elements are blinded via extra random factors $h_2^{\gamma_k}$. The G_k values are correspondingly multiplied with the inverse of $h_2^{\gamma_k}$ to ensure that this random factors will be neutralized in the product $G_k \cdot Q_k$ during the proof verification process.

In the appendix, we will prove the following lemma.
Lemma: The Σ-protocol for knowledge of 1-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.

Generating a proof of correct transparent output value. Transaction spending input coins $C_1, \ldots, C_{old}$ and outputting a transparent net value V_{out} at the transaction cost f should prove that $V_1 + \ldots + V_{old} = V_{out} + f$. Let's see how we can leverage the transcript data of Σ proofs to prove the correctness of this transparent output value.

Note that given the tx_{data}, the verifiers can independently compute the following equation.

$$\mathbf{A} = \frac{\prod_{t=1}^{old}\Big(Comm(0, z_{V_t}, z_{R_t}) \cdot \prod_{k=0}^{m-1} Q_k^{x^k}\Big)}{h_1^{(V_{out}+f)\cdot x^m}} \qquad \text{where} \qquad Q_k = h_2^{\gamma_k} \cdot Comm(0, \rho_k, \tau_k) \qquad (1)$$

Here x is the common verifier challenge used in generating all Σ-proofs for all inputs. x is generated via Fiat-Shamir trick using the initial statements of Σ-proofs of all transaction inputs.

Now let's see that

$$\prod_{t=1}^{old}\Big(Comm(0, z_{V_t}, z_{R_t}) \cdot \prod_{k=0}^{m-1} Q_k^{x^k}\Big) = h_1^{\sum_{t=1}^{old} V_t \cdot x^m} \cdot h_2^{\sum_{t=1}^{old}(R_t \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k)}$$

If the value equation $V_1 + \ldots + V_{old} = V_{out} + f$ holds, then $h_1^{(V_{out}+f)\cdot x^m}$ will be canceled by the value $h_1^{\sum_{t=1}^{old} V_t \cdot x^m}$ in the equation (1) and A will be a valid public key with the witness $a = \sum_{t=1}^{old}(R_t \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k)$.

The prover is the only party knowing a as they possess both the secret values R_t for all input coins and also the blinding factors γ_k^t used for the Σ-proofs generation. Using these observations, the prover can provide a proof of correct output value by signing the transaction with the witness a. All network participants will be able to verify this signature by computing the verification public key A in a non-interactive way from the transaction output value and Σ-proofs statement data.

4.4 JoinSplit Transactions

JoinSplit transactions can simultaneously spend $old \geq 1$ input coins to output $new \geq 0$ fresh output coins and net transparent output value $V_{out} \geq 0$ at the cost of transaction fee f. Each such transaction will be comprised of corresponding spend descriptions, output descriptions and the transaction balance proof.
Inputs:

- Input coins $C_{I_1}, \ldots, C_{I_{old}}$
- Private coin data $\Big\{(q_1, V_1, R_1), \ldots, (q_{old}, V_{old}, R_{old})\Big\}$ and their corresponding indexes $l_1, \ldots, l_{old}$ in the $CMlist$.

Outputs:

– Redeemed value - V_{out}
– Output coins - $C_{O_1}, \ldots, C_{O_{new}}$
– *JoinSplit* transaction - $tx_{joinsplit}$

Let us define the input coins of the transaction as

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

and the output coins as

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

The *JoinSplit* transaction legitimacy proof should convince all network participants of the following.

- All N_{old} spend transactions are valid spends.
- All N_{new} output coins are valid and do not contain any negative values: $\forall j \in 1, \ldots, N_{new} : V_{Oj} > 0$
- No value is created out of thin air and the transaction balance is preserved $V_{I1} + \ldots + V_{Iold} = V_{O1} + \ldots + V_{Onew} + V_{pub}$.

This proof is generated via the following steps:

1. Each input coin is spent via *Spend* transaction described above, which implies revealing the coin's public key, substrating its serial number from all commitments in the list $CMList$ and providing a Σ-proof that one double-blinded commitment of the set of new commitments is opening to 0. Note that *Spend* does not reveal the coin's value explicitly.

2. For each output coin C_{Ol}

   - The prover provides a zero-knowledge range proof BP_O, showing that the coin does not hide a negative value. This is done with the help of Bulletproofs for double-blinded commitments which are described in detail in the Appendix B.

3. The prover provides a zero-knowledge proof that

$$V_{I1} + \ldots + V_{I_{old}} = V_{OUT} + V_{O1} + \ldots + V_{O_{new}} + f$$

4. Prover signs the transactions with the corresponding spending keys $q_1, \ldots, q_{old}$ of all input coins.

The resulted $tx_{joinsplit}$ is comprised of the following elements

$$\begin{aligned} tx_{joinsplit} = \Big(& [\Sigma_1, Q_i], \ldots, [\Sigma_{old}, Q_{old}], V_{out}, Proof_{balance}, \\ & [C_{O1}, RangeProof_{O_1}], \ldots, [C_{O_{new}}, RangeProof_{O_{new}}], \\ & sign_1, \ldots, sign_{old} \Big) \end{aligned}$$

For each N_{old} input coins we will have N_{old} separate Σ-proofs published by the spender, which proof transcripts contain the following data

$(z_{V_1}, \cdots, z_{V_{N_{old}}})$ and $(z_{R_1}, \cdots, z_{R_{N_{old}}})$ where $z_{V_t} = V_t \cdot x^m - \sum_{k=0}^{m-1} \rho_k^t x^k$ and $z_{R_t} = R_t \cdot x^m - \sum_{k=0}^{m-1} \tau_k^t x^k$

$$\{h_2^{\gamma_0^t} \cdot Comm(0, \rho_0^t, \tau_0^t), \cdots, h_2^{\gamma_{m-1}^t} \cdot Comm(0, \rho_{m-1}^t, \tau_{m-1}^t)\} \qquad \text{for } t \in 1, ..., N_{old}$$

Before describing how the prover could generate a transaction balance proof, let's notice that the verifier can leverage the published transaction $tx_{joinsplit}$ data to go over the following steps

1. Take all output coins, the net output value V_{OUT}, the transaction fee f and the common challenge value x used for Σ-proofs constructions and compute the following element

$$\begin{aligned}\mathbf{A} :&= (C_{O1} \cdot \ldots \cdot C_{ON_{new}})^{x^m} \cdot h_1^{(V_{OUT}+f)x^m} = \\ &= g^{(S_{O1}+\ldots+S_{ON_{new}})x^m} h_1^{(V_{OUT}+V_{O1}+\ldots+V_{ON_{new}}+f)x^m} h_2^{(R_{O1}+\ldots+R_{ON_{new}})x^m}\end{aligned}$$

2. Take the elements $z_{V1}, \ldots, z_{V_{N_{old}}}, z_{R1}, \ldots, z_{R_{N_{old}}}$ and $\{Comm(0, \rho_k^t, \tau_k^t)\}_{k=0}^{m-1}$ from the corresponding Σ-proof transcripts and compute the element

$$\begin{aligned}\mathbf{B} :&= Comm(0; z_{V1} + \ldots + z_{VN_{old}}, z_{R1} + \ldots + z_{RN_{old}}) \cdot \prod_{t=1}^{N_{old}} \prod_{k=0}^{m-1} Q_k^{x^k} \\ &= Comm(0; z_{V1} + \ldots + z_{VN_{old}}, z_{R1} + \ldots + z_{RN_{old}}) \cdot \prod_{t=1}^{N_{old}} \Big(\prod_{k=0}^{m-1} (h_2^{\gamma_k^t} \cdot Comm(0; \rho_k^t, \tau_k^t))^{x^k} \Big) \\ &= h_1^{(V_{I1}+\ldots+V_{IN_{old}})x^m} h_2^{(R_{I1}+\ldots+R_{IN_{old}})x^m} \\ &= h_1^{\sum_{t=1}^{old} V_{It} \cdot x^m} h_2^{\sum_{t=1}^{old}(R_{It} \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k)}\end{aligned}$$

If the balance transaction holds, the h_1 exponents in A and B will cancel each other out and we will have

$$\frac{\mathbf{A}}{\mathbf{B}} = g^X h_2^Y \tag{2}$$

where

$$X = (S_{O1} + \ldots + S_{ON_{new}})x^m \text{ and } Y = \sum_{t=1}^{new} R_{Ot} \cdot x^m - \sum_{t=1}^{old} \Big(R_{It} \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k \Big)$$

Now we can observe that for providing a balance proof, it is sufficient for the transaction owner to prove the knowledge of the exponent values X and Y in the equation (2). So the $BalanceProof$ is simply a generalized Schnorr proof of knowledge of the discrete logarithm relation.

In order to confirm the $JoinSplit$ transaction's legitimacy, all network participants should perform the following verification steps.

- Verify the correctness of all 1-out-of-Many proofs $\Sigma_{I_1}, \ldots, \Sigma_{I_{old}}$.
- Check the signatures $sig_{I_1}, \ldots, sig_{I_{old}}$ with help of the input coins public keys $Q_1, \ldots, Q_{old}$.
- Verify all range proofs $BP_{O_1}, \ldots, BP_{O_{new}}$to ensure non of the output coins $C_{O_1}, \ldots, C_{O_{new}}$ contains a negative value.
- Compute the special discrete logarithm relation from the output coin and spent proof transcript data and then check the non-interactive Schnorr proof of knowledge to ensure that the transaction balance is preserved.

5 Batching & Precomputation Techniques for 1-out-of-N Proofs in the Zerocoin Setup

In the blockchain application, the verifier will have to verify multiple 1-out-of-N proofs simultaneously. For example, the blockchain nodes receiving a block of transactions need to verify all transactions and thus the corresponding proofs in parallel. Here we describe an important optimization concerning the simultaneous verification of multiple 1-out-of-N proofs in the Zerocoin setup.
Let us 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 (in our construction, S_t are dynamically computed from the coin's public address Q_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 a single One-out-of-Many Proof boils down to a large multi-exponentiation to check the following equivalency.

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

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

$$f_i^t := \prod_{j=0}^{m-1} f_{j,i_j}^t \qquad D_t := \prod_{k=0}^{m-1} (G_k^t \cdot Q_k^t)^{-x^k} \qquad E_t := Comm(0, z_V^t, z_R^t) \tag{3}$$

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

$$\prod_{i=0}^{N} \Big(C_i^t\Big)^{f_i^t} \cdot D_t \equiv E_t \tag{4}$$

We want to benefit from 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 differ for each transaction.

In order to benefit from batching techniques we can use the following trick. Considering the fact that $C_i^t = \frac{C_i}{g^{(S_t)}}$ and the either the values S_t or their preimages are explicitly revealed during the Spend transaction, we can rewrite the equation (4) as

$$\prod_{i=0}^{N} \Big(C_i^t\Big)^{f_i^t} \cdot D_t = \prod_{i=0}^{N} \Big(\frac{C_i}{g^{S_t}}\Big)^{f_i^t} \cdot D_t \tag{5}$$

and check if the following equivalency holds

$$\prod_{i=0}^{N} C_i^{f_i^t} \equiv \frac{E_t}{D_t} \cdot g^{S_t \cdot (\sum_{i=0}^{N} f_i^t)} \tag{6}$$

The verifier can do this for all proofs, as 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 us to

1. Perform batch verification of the transaction Σ proofs.
2. Speeding up the batch verification by pre-computing the exponent values of C_i (resulting in another 25-60% performance optimization)

Now, in order to perform verification of M spend proofs in batch, the verifier can generate M random values $y_1, \ldots, y_t$ and do the following computations

$$\prod_{t=1}^{M}\Big(\prod_{i=0}^{N} C_i^{f_i^t}\Big)^{y_t} = \prod_{t=1}^{M}\Big(\frac{E_t}{D_t}\cdot g^{s_t\cdot(\sum_{i=0}^{N} f_i^t)}\Big)^{y_t} \tag{7}$$

which in turn is equivalent to verifying that

$$\prod_{i=0}^{N} C_i^{\sum_{t=1}^{M} y_t\cdot f_i^t} = g^{\sum_{t=1}^{M}\Big(y_t\cdot s_t\cdot\sum_{i=0}^{N} f_i^t\Big)}\cdot\prod_{t=1}^{M}\Big(\frac{E_t}{D_t}\Big)^{y_t} \tag{8}$$

This helps to save N exponentiation for each extra proof verification resulting to highly efficient batch verification process.

6 Extending Lelantus for direct anonymous payments

Zerocoin does not provide a mechanism for one user to pay another in "Zerocoins" directly. In order to spend the coin $C = g^S h_1^V h_2^R$, the coin owner should exclusively possess the coin private values q, V, R. Note, that *Spend* transaction requires a valid signature via the spending key q. If during the $Mint$ transaction we could generate the coin spending key q in a way that it becomes accessible only to the intended recipient, then we will have an authentication system authorizing only the intended recipient to spend the coin.

Sender (V, R)		Recipient
		Samples $x \leftarrow_R \mathbb{Z}_p$
Samples $y \leftarrow_R \mathbb{Z}_p$	g^x $\longleftarrow$	Computes public address g^x
Computes $S = Hash(g^{xy})$ $C = g^S h_1^V h_2^R$		For spending the coin C :
C is published to blockchain	y, V, R $\longrightarrow$	1. Reveals the coin public address g^{xy} 2. Generates proof for the spend transfer 3. Signs the transaction with secret key $x \cdot y$

We leverage the fact that coin serial numbers are not generated randomly, but rather from through hashing a specific public key associated with the coin and initially known only to the coin owner. This public key is revealed during the spend transaction and the corresponding private key also referred

as the coin's private address is used to sign the *Spend* transaction. Based on this observation, we can bind the serial number to a public key, which corresponding secret key will be known only to the intended recipient authorizing him to make a valid signature and spend the coin. For $Mint$-ing coins which can be spent only by the targeted recipient and not by the coin creator, the sender and recipient can engage into a Diffie-Hellman-like protocol depicted above. Note, that the coin creator will still notice when this coin associated with the public address g^{xy} is being spent. Although this is a privacy drawback, it can be easily eliminated if we require the recipient to immediately spend and re-mint the received coin.

7 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

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 2. 1-out-of-N Proofs Performance

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 $3m$ exponentiation. The computation of all G_k requires mN exponentiation as the elements $h_2^{-\gamma_k}$ are just the inverses of elements already computed for Q_k. Proofs can be verified using $N+2mn+2m+15$ group exponentiations as follows: $N+2m+2$ exponentiation for the last equation implying big multiexponentation, and $2mn+4$ for the remaining. Our schemes can be instantiated over any group G where the DDH problem is computationally hard. To evaluate the performance of our proofs, we created a reference implementation in C++ using the popular library libsecp256k1 which uses the elliptic curve secp256k1 with 128-bit security and is used in numerous cryptocurrency projects. In the compressed form, secp256k1 points are stored as 33 bytes. In the Table 2 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 is of 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. Table 3, 4, 5 and 6 expose proof verification benchmarks for different values N. 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

Batch Size	Verification Time	Average 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 3. Batch Verification Timing for the Anonymity Set of 16384

Batch Size	Verification Time	Average 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 4. Batch Verification Timing for the Anonymity Set of 32384

Batch Size	Verification Time	Average 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 5. Batch Verification Timing for the Anonymity Set of 65536

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

Table 6. Batch Verification Timing for the Anonymity Set of 262144

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.

8 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.

Acknowledgments.

The author thanks Jens Groth for his continuous support and feedback on the generic ideas evaluated in this paper and Benedict Bünz for his important suggestions for the Bulletproof protocol for double-blinded commitment schemes. 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. Claus-Peter Schnorr. Efficient signature generation by smart cards. Journal of Cryptology, 4(3):161174, 1991.
3. ZCash parameter generation. https://z.cash/technology/paramgen.html, 2016. Accessed: 2017-09-28.
4. 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.
5. 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).
6. Greg Maxwell. Confidential transactions. https://people.xiph.org/greg/confidential-values.txt, 2016.
7. 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
8. 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.
9. 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.
10. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology, 16(3):143-184, 2003.
11. 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.
12. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In ACM CCS, pages 62-73, 1993.
13. 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.
14. 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.
15. 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.
16. Daira Hopwood, Sean Bowe, Tailor Hornby, and Nathan Wilcox. Zcash protocol specification, 2018. URL: https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.
17. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In ACM CCS, pages 62-73, 1993.
18. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulationextractable SNARKs. Cryptology ePrint Archive, Report 2017/540, 2017. http://eprint.iacr.org/2017/540.
19. Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. In Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 17, pages 609642, 2017.

A: Security Proofs of 1-out-of-N Proofs for Double Blinded Commitments

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

$$\prod_{i=0}^{N} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} (G_k \cdot Q_k)^{-x^k} = \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \Big(h_2^{-\gamma_k} \cdot \prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot h_2^{\gamma_k} \cdot Comm(0, \rho_k, \tau_k) \Big)^{-x^k} =$$

$$= \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \Big(\prod_{i=0}^{N-1} C_i^{-p_{i,k} \cdot x^k} \cdot Comm(0, -\rho_k x^k, -\tau_k x^k) \Big)$$

$$= \prod_{i=0}^{N} C_i^{p_i(x)} \cdot \Big(\prod_{i=0}^{N-1} C_i^{-\sum_{k=0}^{m-1} p_{i,k} \cdot x^k} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big)$$

$$= \prod_{i=0}^{N} C_i^{\sigma_{l,i} x^m} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big)$$

$$= C_l^{x^m} \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big)$$

$$= Comm(0, V \cdot x^m, R \cdot x^m) \cdot Comm(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k) \Big)$$

$$= Comm(0, V \cdot x^m - \sum_{k=0}^{m-1} \rho_k x^k, R \cdot x^m - \sum_{k=0}^{m-1} \tau_k x^k) = Comm(0, z_V, z_R)$$

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 $B, C, G_1, \dots G_{m-1}$; $Q_0, Q_1, \dots Q_{m-1} \leftarrow G$; $f_{0,1}, \dots f_{m-1,n-1}, z_A, z_C, z_V, z_R \leftarrow Z_q$. Next it computes $f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i}$; $A = Com_{ck}(f_{0,0}, \dots, f_{m-1,n-1}, z_A) B^{-x}$ and $D = Com_{ck}(f_{i,j}(x - f_{i,j}), z_C) C^{-x}$. The simulator computes 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, $Q_0, Q_1, \dots Q_{m-1}, G_1, \dots G_{m-1}$ 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 $f_{0,1}, \dots f_{m-1,n-1}, z_A, z_C$ 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 $f_{0,1}, \dots f_{m-1,n-1}, z_A, z_C$ and $f'_{0,1}, \dots f'_{m-1,n-1}, z'_A, z'_C$ 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 $\Big((f_{j,i}^{(0)}, z_V^{(0)}, z_R^{(0)}), \dots, (f_{j,i}^{(m)}, z_V^{(m)}, z_R^{(m)}) \Big)$ with respect to $m+1$ different challenges $x^{(0)}, \dots, x^{(m)}$ and the same initial message. Assume that m > 1. As is described in the original paper [9], 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 $0 \leq e \leq m$ 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 $R = \sum_{e=0}^{m} \theta_e z_R^e$.

B: Bulletproofs for Double-Blinded Commitments

Here we describe how Bulletproofs can be generalized to support 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 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:

$h_i' = h_i^{(y^{-i+1})} \in G$ for $j \in [1, ..., n]$

$P = A \cdot S^x \cdot \mathbf{g^{-z}} \cdot (\mathbf{h'})^{\mathbf{zy^n + z^2 2^n}} \in G$

Verifier checks if

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

$P \equiv h_1^{\mu} \mathbf{g^l h'^r}$

$\widehat{t} \equiv \langle \mathbf{l}, \mathbf{r} \rangle \in Z_P$

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 [7]. To use the inner-product argument, observe that verifying $P \equiv h_1^{\mu} g^{\mathbf{l}} \mathbf{h'^r}$ 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 of the provided Bulletproof, 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

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

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

$$\begin{aligned} S &= \left(h_1^{-\mu} \cdot A \cdot \mathbf{g}^{-z \cdot \mathbf{1}^n - \mathbf{l}} \cdot \mathbf{h}'^{-z \cdot \mathbf{y}^n - z^2 \cdot \mathbf{2}^n - \mathbf{r}}\right)^{-x^{-1}} \\ T_1 &= \left(h_1^{-\tau_x^1} \cdot h_2^{-\tau_x^2} \cdot g^{\sigma(y,z) - \widehat{t}} \cdot V^{z^2} \cdot T_2^{x^2}\right)^{-x^{-1}} \end{aligned}$$

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 [11] and [10] 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.