Lelantus: A New Design for Anonymous and Confidential Cryptocurrencies

Aram Jivanyan
Zcoin
www.zcoin.io
Email: aram@zcoin.io

Abstract—**We propose Lelantus, a new privacy payment protocol that ensures confidential and anonymous blockchain transactions with small transaction sizes, short verification times, and without requiring a trusted setup. Based on the modified construction of one-out-of-many proofs, Lelantus enables direct anonymous payments where users can pay each other by hiding their transaction origins and amounts. It efficiently supports large anonymity sets of size hundred thousand and beyond by providing logarithmic proof sizes and fast batch verification of transaction proofs. Lelantus can be integrated also with other major blockchain privacy solutions MimbleWimble and Confidential Transactions, resulting in hybrid designs that ensure transaction anonymity. We analyze and provide formal security proofs for the suggested direct anonymous payment system and for the proposed modifications of one-out-of-many proofs.**

I. 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 which is highly desirable privacy feature for financial transactions. Other cryptographic constructions which do offer anonymity, such as Monero [8], QuisQuis,[16], Zerocash [3] and Zerocoin [30] protocols suffer of 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 scheme does not support direct anonymous payments, it also does not hide the transaction amounts and works only with fixed denominated coins. Monero and QuisQuis both enable direct anonymous payments of arbitrary amounts, but provide relatively weak anonymity guaranties. For example, in Monero, transaction senders specify some number of addresses to mix in to their own transaction, and then use this list of addresses to form a ring signature and hide which specific address was theirs. The anonymity set size in Monero transactions is 10 which from security perspectives is very low and opens doors for powerful tracking analysis attacks as is discussed in [12], [15]. QuisQuis supports anonymity sets of size 16 which still makes it vulnerable to all attacks endemic to the decoy systems with small anonymity sets[12]. Unlike Monero and Quisquis, Zerocash efficiently supports large anonymity sets of size billions, but this hard-to-beat efficiency and advanced privacy features come at the price of reliance on relatively new security assumption and a trusted setup procedures, which necessitate the user's trust in the correctness of this setup.
The goal of this paper is to provide a novel practical transaction scheme which is based only on standard cryptographic assumptions and does not require any trusted setup procedures, at the same time efficiently supports strong anonymity and confidentiality properties for direct blockchain payments.

A. Our Contribution

We start by achieving transaction anonymity with a Zerocoin-like setup which is implemented through one-out-of-many proofs as is discussed in [4]. Zerocoin is strictly limited to work with fixed denominated coins which can be spend anonymously but without any ability of merging, splitting or partially redeeming multiple coins in a confidential way. We extend this fundamental construction in a few significant ways. First, we extend zerocoin coins with secret balances enabling the user to mint coins of arbitrary values and later spent them anonymously into fresh new outputs of arbitrary values. We introduce an innovative zero-knowledge balance proof construction to ensure that the transaction's input and output values sum up. The balance proof generation method leverages specific design properties of the modified one-out-of-many protocol construction discussed in this paper. These properties enable to extract encoded information about the spent inputs values necessary for proving the balance without revealing the inputs origins. Our construction admits an arbitrary number of transaction inputs and outputs without limitations. Next we introduce shielded addresses and enable the user to perform direct anonymous payments where the transaction outputs can be spent only by the intended recipient.
Further, we discuss a highly-efficient batch verification method that enables the network validators verifying hundreds or even thousands of different transactions simultaneously by significantly lowering the average cost of single transaction verification.
The resulting scheme has numerous advantages over the alternative secure payments protocols, namely:

- It does not require any trusted setup processes and the

protocol security is relying only on standard and time-tested cryptographic assumptions.
- Provides strong anonymity of blockchain transactions by efficiently supporting large anonymity sets of size 65536 and higher.
- Transactions are supporting direct anonymous payments and can admit an arbitrary number of input and output coins. Shielded coins can be merged, split or redeemed in an anonymous and confidential way.
- The transaction communication complexity is logarithmic in the anonymity set size. The computational complexity of transaction verification although theoretically is linear, it can be significantly accelerated through batch verification methods and result in a highly-efficient verification process.

We also analyze and provide formal security proofs for our direct anonymous payment system by using a robust security framework introduced by Zerocash[3]. Our paper discusses a novel modification of a one-out-of-many proof system that works with generalized Pedersen commitments and can be of independent interest. We provide complete formal security proofs of this construction's soundness, zero-knowledge and completeness properties.
Lelantus is already approved to be running in production by two major privacy cryptocurrency projects Zcoin and Beam [7], [10].

B. Overview and Intuition

Lelantus can be integrated with any blockchain-based currency, such as Bitcoin. To give a sense of 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. Zerocoin, designed as an extension to Bitcoin and similar cryptocurrencies[30], was one of the first anonymous cryptocurrency proposals to ensure high anonymity for the blockchain transactions. It enables users to transform their base layer coins(e.g. Bitcoin) into shielded coins and later spend the shielded coin without revealing its origin. When spent, a special zero-knowledge proof is generated convincing that the spent coin is one of the previously minted shielded coins which was not already spent before. The set of all shielded coins that the spent coin belongs to is referred to as an anonymity set. Intuitively, the size of the anonymity set defines how strong is the guaranteed transaction anonymity. The bigger is the anonymity set size, the stronger anonymity is archived for each spend coin. The original Zerocoin construction[30] was based on the RSA accumulators and was not efficient from the communication standpoint as each RSA accumulator based zerocoin proof consists of $\sim$25KB. It also required a trust toward the exploited RSA parameters. In [4] authors presented an efficient Zerocoin construction based on an innovative one-out-of-many proof system, which does not require any trusted setup operations, supports significantly smaller proof sizes and efficient computations compared to the original construction. Zerocoin protocol consists of four algorithms (*Setup, Mint, Spend, verify*) which can be implemented with help of one-out-of-many proofs over the homomorphic Pedersen commitments [14] as described below.

1) **Setup:** Generates the public parameters of the underlying commitment scheme by specifying the group G and fixing two generators g and h with no known discrete logarithm relation.
2) **Mint:** For minting a new coin, the user generates a unique coin serial number secret sn, and then commits to sn using the Pedersen commitment scheme and a fresh blinding factor r: The resulted coin $C = \text{Com}(sn, r)$ is published to the blockchain and is added to the list of all previously minted coins $\{C_0, C_1, ...C_{N-1}\}$. The coin serial number sn and the opening value r are used later to spend the coin C.
3) **Spend:** The user parses the set of all previously minted coins $\{C_0, C_1, ...C_{N-1}\}$ and homomorphically substracts the serial number value sn from all these coins. This results in a new set of commitments where one will obviously be opening to 0. Next the user generates a one-out-of-N proof of knowledge of this secret commitment opening to 0 without revealing its index in the referred set. The proof transcript and the coin's serial number sn are published to the blockchain.
4) **Verify:** All network participants can take the revealed serial number sn and check that it does not appear in any previous spend transaction. Next they can homomorphically subtract this value from all other coins of the referred anonymity set and get a new set of commitments. Last, the network participants can check the validity of the provided one-out-of-N proof against this new composed set of commitments.

Step 2: Enabling to mint, merge, split and redeem coins of arbitrary values. As discussed, Zerocoin makes transaction history private, but does not support payments of arbitrary values and also can not enable direct anonymous payments. In [6], a scheme for Confidential Transaction has been proposed to ensure transaction amount confidentiality. With this scheme, the coins are represented through homomorphic cryptographic commitments which encode the coin values v as $C = \text{Com}(v, r)$. Assuming the confidential transaction spends old input coins $\text{Com}(v_1^i, r_1^i), \ldots, \text{Com}(v_{\text{old}}^i, r_{\text{old}}^i)$ and outputs new fresh new coins $\text{Com}(v_1^o, r_1^o), \ldots, \text{Com}(v_{\text{new}}^o, r_{\text{new}}^o)$ it is easy to ensure that the transaction balance equation holds

$$v_1^i + \cdots + v_{old}^i = v_1^0 + \cdots + v_{new}^o$$

by leveraging the additive homomorphic properties of the underlying commitment scheme and proving that

$$C_1^o + \ldots + C_{new}^o - C_1^i - \ldots - C_{old}^i = \text{Com}(0, R)$$

Here R is the aggregate of the blinding factors known only to the transaction owner who can prove its knowledge in a zero-knowledge way. When Pedersen commitment scheme is used, the $\mathrm{Com}(0, R)$ will represent a valid signature verification key with the corresponding signing key R. This enables the transaction owner to prove the knowledge of R by signing the transaction data, which can be later verified with the verification key $\mathrm{Com}(0, R)$ computed from the transaction inputs and outputs. Obviously, the signature verification will pass only if the transaction balance is preserved. This scheme works because the transaction input commitments $C_1^i, \ldots, C_{\text{old}}^i$ are all explicit, while our goal is to ensure both transaction anonymity and confidentiality. To archive that, our first step is to represent coins with generalized Pedersen commitments which commit simultaneously to the coin's serial number and value. Next we support generic $Spend$ transactions which can spend multiple inputs anonymously and output multiple outputs by providing a zero-knowledge balance proof construction, which proves the transaction inputs and outputs sum up without revealing the input coins origins or the coins amount.

Step 3: Enabling direct anonymous payments. We also want to support direct anonymous payments which enable users to transfer value confidentially to the targeted recipients without the further ability for tracing the transferred coins. We introduce shielded addresses, which are generated by the recipient and used by the sender for generating the output coins [34]. Usage of the shielded addresses helps to keep the serial number of the newly created coin private for the sender and enables the recipient to spend the received coin anonymously. We prevent malleability attacks on a spend transaction (e.g., malicious assignment by re-targeting the recipient address of the transaction public output) by generating the coin serial numbers as public keys and require each spend transaction to be signed with the corresponding witness.

Step 4: Performing batch verification of transaction proofs. The communication and computational complexity of transactions are of extreme importance for the practicality and scalability of the payment system. In our system, each spent coin requires a separate one-out-of-many proof to be generated, in which communication complexity is only logarithmic of the anonymity set size N and is highly efficient. The verification complexity of one-out-of-many proof is linear of the anonymity set size N and can take hundreds of milliseconds to verify within a set of a few dozen thousand commitments. In this paper, we will illustrate important batch verification techniques that enable us to verify multiple proofs in batches and lower the average cost of a single proof verification to dozens of milliseconds within significantly large commitment sets. This makes our scheme performance very competitive with other cutting-edge privacy payment approaches. We do not discuss any optimization for the proving complexity of one-out-of-many proofs, which is $O(N\log N)$ but can be significantly lowered by techniques described in [17].

The Table I illustrates how Lelantus compares with other confidential payments protocols such as Monero, QuisQuis, Zerocoin and Zerocash. It becomes evident that the proof sizes and verification times of SNARK-based constructions are hard to beat. This unmatched performance, however, is expensively bought with having to require a trusted setup and reliance on knowledge of exponent cryptographic assumptions. Lelantus provides strong privacy and competitive performance while still relying on standard cryptographic assumptions and without requiring a trusted setup. At the same time, it offers orders of magnitude stronger anonymity properties than Monero or QuisQuis while being also more efficient due to the smaller transaction sizes and shorter verification times.

C. Related Works and Comparison

Privacy remains one of the most important issues for blockchain [18] and there is a significant amount of active research on the development of efficient zero-knowledge proofs. Currently numerous constructions achieve different tradeoffs between transaction proof sizes, proving and verification times under different trust models and cryptographic assumptions. From the verification complexity standpoint the most efficient proof systems to date are zk-SNARKs [28], [29] which require a trusted setup processing. Recently, there have been designed powerful transparent systems such are zk-STARKS [19] and Supersonic[22] which are zero-knowledge proofs for Rank-1 Constraint Satisfaction (R1CS). When applied to the private blockchain transactions use case, their proving time and proof sizes still seems to be beyond the practicality limit for large-scale payment applications[18]. Relatively shorter proofs compared to STARKs are produced by Aurora [20], which uses a transparent setup and is plausibly post-quantum secure. Bulletproofs [13] are another powerful zero-knowledge proof technology based on standard cryptographic assumptions and not requiring any trusted setup procedures. They are ubiquitously used in private digital currency systems[7], [10], [9], [8] for generating efficient range-proofs. Sonic [21] is a zero-knowledge SNARK system which supports universal and continually updateable trusted setup process. Halo [23] is a recent scientific breakthrough which enables recursive proof composition without a trusted setup and based on the discrete log security assumption.

D. Lelantus and MimbleWimble

MimbleWimble is another popular blockchain privacy protocol which powers few cryptocurrency projects including Beam[10] and Grin[9]. The transaction inputs and outputs are introduced through Pedersen commitments and this protocol uses the commitment blinding factors of transaction inputs and outputs as private keys. Sender and receiver must interact to construct a joint signature to authorize a transfer of funds. MimbleWimble enables to aggregate all transactions within the block into one giant transaction resulting to significantly smaller ledger. It also provides cut-through methods, in which all spent outputs cancel against corresponding inputs across the blockchain and helps to erase most of the blockchain

TABLE I
SECURITY PROPERTIES AND EFFICIENCY CONSIDERATIONS FOR DIFFERENT PRIVACY SOLUTIONS. THE LELANTUS PROOF SIZES AND PERFORMANCE NUMBERS ARE MEASURED FOR 2-INPUTS-2-OUTPUT TRANSACTIONS COMPUTED BY BATCH VERIFYING 100 PROOFS.

	Anonymity Set Size	Trusted Setup	Security Assumptions	Proof Size(Kb)	Proving Time(S)	verification Time(ms)
Monero	11	No	Well Studied	2.1	0.9	47
QuisQuis	16	No	Well Studied	13	0.47	71
Zerocash	$4B$	Yes	Relatively New	0.3	1-10	8
Zerocoin	10000	Yes	Well Studied	25	0.2	200
Lelantus$_{1024}$	1024	No	Well Studied	2.7	0.27	7.8*
Lelantus$_{16384}$	16384	No	Well Studied	3.9	2.35	18.2*
Lelantus$_{65536}$	65536	No	Well Studied	5.6	4.8	62*

history and shrink the ledger size further. However the linkability of MimbleWimble transactions remains a major privacy drawback of the protocol. The network observers can easily link transaction inputs and outputs and break the anonymity of users. In order to overcome this limitation, Beam has designed a hybrid scheme of Lelantus and MimbleWimble [11] which provides strong anonymity to MimbleWimble transactions by enabling anonymous spends. In this hybrid approach, users can add their coins into the shielded pool, which later can be transformed to MimbleWimble-transaction compatible coins without revealing their origins. This hybrid scheme has been recently lunched on the Beam' test network and will be soon deployed in production[11].

II. Cryptographic Background

We denote $R = \{x; w \mid L\}$ to be a binary relation for instances x and witnesses w where L_R defines the corresponding language; i.e $L_R = \{x | w : (x, w) \in R\}$. We use $x \leftarrow_R T$ for sampling an element x uniformly at random from a set T.

A. Generalized Pedersen Commitments

Let G be a cyclic group of prime order p where the discrete logarithm problem is hard, and let Z_p be its scalar field. Let g and h be random generators whose discrete logarithm relationship is unknown. A Pedersen commitment scheme [14] enables to commit to $m \in Z_p$ by picking a random blinding factor $r \in Z_p$ and computing $Com(m, r) = g^m h^r$. Pedersen commitment scheme is perfectly hiding and computationally strongly binding under the discrete logarithm assumption as defined below.

Definition 1 (Hiding). A commitment scheme is perfectly hiding if the commitment does not leak any information about the committed value. More formally for all probabilistic polynomial time stateful adversaries $\mathcal{A}$ and the given commitment key $ck = (G, p, g, h)$

$$\Pr[(m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck); b \leftarrow \{0,1\}; \\ c_b = Com(m_b, r_b) : A(c) = b] = \frac{1}{2}$$

Definition 2 (Binding). A commitment scheme is computationally strongly binding if the commitment can only be opened in one way. More formally for all probabilistic polynomial time stateful adversaries $\mathcal{A}$ and the given commitment key $ck = (G, p, g, h)$

$$\Pr[(m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck); (m_0, r_0) \neq (m_1, r_1) : \\ Com(m_0, r_0) = Com(m_1, r_1)] \approx 0$$

The Pedersen commitment scheme can be generalized for multiple messages, i.e. given n messages $m_1, m_2, ..., m_n$ and $n + 1$ independent generator points $g_1, g_2, \dots, g_n$ and h, one can create a commitment of the following form

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

Generalized Pedersen commitment scheme is also computationally strongly binding, perfectly hiding and retains cryptographic homomorphic properties. In our protocol design, we use a private case of generalized Pedersen commitment scheme which is committing to two messages and utilizes three different group generators g, h, f. It commits to the given messages s and t as $Comm_{ck}(s, t, r) = g^s h^t f^r$. For sake of simplicity we refer to this private case of generalized Pedersen commitment scheme as a double-blinded commitment as without loss of generality it can be viewed as a commitment to a message s using two different blinding factors t and r.
Note: We will henceforth define the Pedersen commitment for value m using randomness r as $Com(m, r)$. A double-blinded commitment using the values s, t and r is denoted as $Comm(s, t, r)$.

B. One-out-of-many proofs for a commitment opening to zero

One-out-of-many proof is a special 3-step interactive proof system (Sigma protocol) first introduced by [4] aiming to prove the knowledge of one out of N public commitments $\{C_0, \dots, C_{N-1}\}$ being a commitment to 0. More precisely it is a Σ-protocol for the following relation

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

Important optimizations of one-out-of-many proofs was provided in [5] which reduce the proof sizes and improve the proving complexity of the original proposal. In this paper we introduce a modified version of one-out-of-many proofs for double-blinded commitments, which is a proof of knowledge that the public list of N double-blinded commitments $\{C_0,\ldots,C_{N-1}\}$ includes a commitment to zero $Comm(0,v,r)$. More formally, it can be defined as a proof system for the following relation:

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

We will formally prove that our proposed design for one-out-of-many proofs for double-blinded commitments is perfectly complete, special honest verifier zero-knowledge, and has n-special soundness 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 which also means 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 valid protocol transcript which will be indistinguishable from a valid transcript generated by the owner of the witness.
- **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 provided n different accepting transcripts.

We will use the non-interactive variant of one-out-of-many proofs obtained by Fiat-Shamir heuristic [4] and the proof transcript with respect to the public list of commitments $\{C_0,\ldots,C_{N-1}\}$ is denoted by $\pi_{\text{ooon}}(C_0,\ldots,C_{N-1})$.

C. Bulletproofs

Bulletproofs are a zero-knowledge interactive proof systems for providing short and aggregatable range proofs [13]. Formally, let $v \in Z_p$ and let $C \in G$ be a Pedersen commitment to v using the randomness r. Then the proof system will convince the verifier that the committed value $v \in [0, 2^n - 1]$. In other words, the proof system proves the following relation

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

Bulletproofs are interactive protocols which can be made non-interactive by using the Fiat-Shamir heuristic in the random oracle model. We will denote the transcript of non-interactive range-proof for the commitment C with respect to the generators g and h and the specified range $[0, 2^n - 1]$ by $\pi_{\text{range}}(C; g, h, n)$.

D. Proof of knowledge of discrete logarithm representation

A group element can be expressed as the product of powers of certain generators which is called the DL representation of the element with respect to that generators [24]. A discrete logarithm representation (or DL-REP for short) of $h \in G$ with respect to the tuple of l elements $g_1,\ldots,g_l \in G$ is the tuple $(a_1,\ldots,a_l) \in Z_p^l$ where $h = g_1^{a_1}\cdots g_l^{a_l}$. Note that when

$$\begin{array}{lcl}
\textit{Prover} & & \textit{Verifier} \\
g_1,\ldots,g_k; a_1,\ldots,a_k; y & & g_1,\ldots,g_k; y \\
\text{Computes} & & \\
\quad w_i \leftarrow_R Z_p & z & \\
\quad z = \prod_{i=1}^{l} g_i^{w_i} \in G & \longrightarrow & \\
 & \xleftarrow{x \leftarrow \{0,1\}^{\lambda}} & \text{Accepts if} \\
\quad r_i = w_i - x \cdot a_i \in Z_p & & \\
 & \xrightarrow{r_1\ldots,r_l} & z = y^x \prod_{i=1}^{l} g_i^{r_i}
\end{array}$$

Fig. 1. Proof of Representation

$l = 1$, h is represented as $h = g_1^{a_1}$. In this case finding the a_1 having only g_1 and h is the discrete logarithm problem. Thus the discrete logarithm problem can be regarded as the special case of the DL representation which contains only one term and it is easy to show that finding a DL representation of $h \in G$ with respect to the generators $g_1,\ldots,g_l \in G$ is at least as hard as the DL problem. To show this, without loss of generality we can assume[24] that the generators $g_1,\ldots,g_l$ are computed based on the fixed generator point $g \in G$ and l random exponents $y_1,\ldots,y_l \in Z_p$ as follows $g_1 = g^{y_1},\ldots,g_l = g^{y_l}$. Having an oracle which can construct the DL-REP of h, one can easily break the DL problem. By inputting h into the oracle to get the representation tuple $x_1,\ldots,x_l\ inZ_p$ and using the fact that for any $i \in 1,\ldots,l$ $g_i = g^{y_i}$, one can easily compute the discrete logarithm of h with respect to the generator g

$$h = \prod_{i=1}^{l} g_i^{a_i} = \prod_{i=1}^{l} g^{y_i \cdot a_i} = g^{\sum_{i=1}^{l} y_i \cdot a_i}$$

It is also can be shown that finding two different representations for some $h \in G$ with respect to the same generator points $g_1,\ldots,g_l$ is also at least as hard as the DL problem [24]. Proof of knowledge of representation (also referred as generalized Schnorr proof of knowledge) is a zero-knowledge argument for the following relation

$$R = \{g_1,\ldots,g_k \in G, y;\ a_1,\ldots,a_k \in Z_p\ |\ y = \prod_{i=1}^{k} g_i^{a_i}\}$$

The protocol is depicted in the Figure 1. Verifying the protocol completeness is straightforward. It can be converted into a secure and special honest-verifier zero-knowledge non-interactive protocol in the random oracle model using the Fiat-Shamir heuristic [33]. Hereafter $\pi_{\text{DLREP}}(y; g_1,\ldots,g_k)$ will denote the transcript of non-interactive proof of representation of the value y with respect to given generators $g_1,\ldots,g_l$.

E. One-time strongly-unforgeable digital signatures

We use a digital signature scheme $Sig = (Setup, \mathcal{K}_{sig}, \mathcal{S}_{Sig}, \mathcal{V}_{Sig})$ that works as follows.

- $Setup(1^\lambda) \rightarrow pp_{sig}$. Given a security parameter λ, $Setup$ samples public parameters pp_{sig} for the Signature scheme.
- $\mathcal{K}_{sig}(pp_{sig}) \rightarrow (pk_{sig}, sk_{sig})$ Given public parameters pp_{sig}, $\mathcal{K}$ samples a pair of verification and signing keys.
- $\mathcal{S}_{sig}(sk_{sig}, m) \rightarrow \sigma$. Given a secret signing key sk_{sig} and a message m, $Sign$ signs m to obtain a signature σ.
- $\mathcal{V}_{sig}(pk_{sig}, m, \sigma) \rightarrow b$. Given a public verification key pk_{sig}, message m, and signature σ, $\mathcal{V}_{sig}$ outputs $b = 1$ if the signature is valid for message m or $b = 0$ otherwise.

We require the signature scheme Sig to satisfy the security property of one-time strong unforgeability against chosen-message attacks(SUF-1CMA security) [37].

F. Key-Private Public Key Encryption

We use a public-key encryption scheme defined as a tuple of four algorithms $Enc = (Setup, \mathcal{K}_{enc}, \mathcal{E}_{enc}, \mathcal{D}_{enc})$ that works as follows.

- $Setup(1^\lambda) \rightarrow pp_{enc}$. Given a security parameter λ, $Setup$ samples public parameters pp_{enc} for the public-key encryption scheme.
- $\mathcal{K}_{enc}(pp_{enc}) \rightarrow (pk_{enc}, sk_{enc})$ Given public parameters pp_{enc}, $\mathcal{K}_{enc}$ samples a public key and a secret key for a single user.
- $\mathcal{E}_{enc}(pk_{enc}, m) \rightarrow \sigma$. Given pk_{enc} and a message m, $\mathcal{E}_{enc}$ encrypts m with the public key pk_{enc} to obtain a ciphertext c.
- $\mathcal{D}_{enc}(sk_{enc}, c) \rightarrow m$. Given the secret key sk_{enc} and the ciphertext c encrypted under the corresponding public key pk_{enc}, $\mathcal{D}_{enc}$ outputs the plaintext m if the ciphertext is valid and outputs $null$ if decryption fails.

We require the encryption scheme Enc to be IND-CCA secure and also satisfy to the *key indistinguishability under chosen-ciphertext attack (IK-CCA security)*[27].which, informally requires that ciphertexts cannot be linked to the public key used to encrypt them, or to other ciphertexts encrypted with the same public key.

III. Lelantus Construction

Lelantus is a decentralized privacy payment scheme allowing direct blockchain transactions with hidden origins and amounts. In this section we provide overview of the underlying building blocks, data structures and algorithms used to construct the direct anonymous payment (DAP) system, and also formalize its security properties.

A. Data Structures and Algorithms

Lelantus is exploiting the following data structures and algorithms.

Basecoin Ledger. Lelantus can be integrated with any blockchain-based currency (e.g. Bitcoin) which is referred as the basecoin currency. The basecoin currency ledger L is the only data storage used over the system to record both the basecoin and the confidential *Mint* and *Spend* transactions introduced by Lelantus.

Addresses. Apart of the basecoin ledger addresses, Lelantus exploits special shielded addresses to power direct anonymous payments. Each user can generate arbitrary number of new shielded address pairs $(\text{addr}_{\text{sk}}, \text{addr}_{\text{pk}})$. For each address, the addr_{pk} is the public component used for receiving funds privately, and the addr_{sk} is the private address used only by the address owner to spend the received funds.

Coins. Coin encodes the abstract monetary value which is transferred through the private transactions. Each coin is associated with

- A coin value v which is measured in basecoins and can be any integer from the system specified range $[0, v_{\text{max}})$.
- A coin public address addr_{pk} which indicates the coin transfer destination. The recipient proves his ownership over the received coins through the corresponding secret address addr_{sk}.
- A coin unique serial number sn. The serial number sn is revealed when the coin is spent and it prevents possible double-spending of the coin.
- A coin commitment denoted as C. This is a double-blinded commitment encoding the coin serial number sn and the coin value v and blinded by a randomly generated blinding factor r. The coin commitment is published and stored on the ledger when the coin is created either through the $Mint$ or $Spend$ transaction.

Private Transactions: Lelantus is introducing two new confidential transaction types to the ledger:

- $Mint$ Transactions. A $Mint$ transaction enables to move base layer coins into the shielded layer where they can be further spend anonymously. $Mint$ transaction creates a transaction data tx_{mint} and records it on the ledger. The transaction data contains the new created coin commitment associated with the provided basecoin value and the recipient public address among other cryptographic information.
- $Spend$ Transactions. A $Spend$ transaction enables to merge, split or redeem previously generated coins in an anonymous and confidential way. The transaction creates the transaction data tx_{spend} and records it on the ledger. tx_{spend} contains the new created transaction output coins and all required zero-knowledge cryptographic proofs.

Algorithms: Lelantus is a decentralized anonymous payment(DAP) system defined as a tuple of polynomial-time algorithms: $\prod$=(**Setup, CreateAddress, Mint, Spend, Receive, Verify**).

- **Setup**: This algorithm takes the security parameter λ and outputs all public parameters used by different building blocks of the protocol including the commitment scheme, range proofs, one-out-of-many proofs, public key encryption and digital signature algorithms. The setup process does not require any trusted procedures.

- **CreateAddress**: This algorithm takes as input the public parameters pp and generates a new shielded address pair $(\text{addr}_{\text{sk}}, \text{addr}_{\text{pk}})$.
- **Mint:** This algorithm takes as input the given public value and basecoin UTXOs which should be minted and creates the mint transaction tx_{mint}.
- **Spend:** This algorithm takes the input coins which should be spent, the public recipient addresses and output coin values and generates the spend transaction tx_{spend}.
- **Verify:** This algorithm is used by network validators to check the validity of the tx_{mint} and tx_{spend} transactions.
- **Receive:** This algorithm takes as input a secret address addr_{sk} and scans the ledger to retrieve all unspent coins sent to the corresponding public address.

We will give the detailed description of all algorithms in Section 4.

Anonymity Sets and List of Serial Numbers: For any given moment

- **C-Pool** denotes the list of all coin commitments generated by the $Mint$ and $Spend$ transactions. In practice the set of all coin commitments can be logically split into multiple enumerated anonymity sets of certain fixed length so each such set can be referred unambiguously by the private transactions.
- **S-Pool** denotes the public list of all serial numbers which are revealed when coins are spent.

B. Security

Zerocash [3] established a robust security framework for the DAP scheme security which captures a realistic threat model with powerful adversaries who are permitted to include malicious commitments into the **C-Pool**, control the choice of transaction inputs and obtain the transactions data in advance. We will formally prove our system security within that security model and recall all corresponding definitions and notations here for the sake of paper integrity.

According to [3], a decentralized anonymous payment system security is defined as follows:

Definition 3.1 A DAP scheme Π=(Setup, CreateAddress, Mint, Spend, Receive, Verify) is secure, if it satisfies the ledger indistinguishability, transaction non-malleability and balance properties.

Each security property is formalized as a game between a polynomial-time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, and in each game the behavior of honest parties is simulated via a oracle $\mathcal{O}^{DAP}$. The oracle $\mathcal{O}^{DAP}$ maintains a ledger L and provides an interface for executing **CreateAddress, Mint, Spend** and **Receive** algorithms for honest parties. To simulate behavior from honest parties, $\mathcal{A}$ passes a query to $\mathcal{C}$, which makes sanity checks and then proxies the queries to $\mathcal{O}^{DAP}$. For the $Mint$ or $Spend$ queries $\mathcal{A}$ is allowed to specify the identities of previous transactions, input coins and recipient addresses. The $\mathcal{A}$ learns the resulting transaction but not any of the secrets or trapdoors involved in producing the transaction. The oracle $\mathcal{O}^{DAP}$ also provides an **Insert** query that allows $\mathcal{A}$ to directly insert arbitrary and potentially malicious transactions to the L.

Balance: This property requires that no bounded adversary $\mathcal{A}$ can own more money than what he has minted or received via payments from others. Balance is formalized by the following **BAL** experiment. The $\mathcal{A}$ adaptively interacts with $\mathcal{O}^{\mathcal{DAP}}$ and at the end of the interaction outputs a set of coins S_{coin}. Letting ADDR be set of all addresses of honest users generated by **CreateAddress** queries, $\mathcal{A}$ wins the game if $v_{\text{unspent}} + v_{\text{basecoin}} + v_{\mathcal{A} \to \text{ADDR}} > v_{\text{mint}} + v_{\text{ADDR} \to \mathcal{A}}$ which means the total value he can spend or has spent already is greater than the value he has minted or received. Here

- v_{unspent} is the total value of unspent coins in the S_{coin}.
- v_{basecoin} is the total value of public outputs of *Spend* transactions inserted by $\mathcal{A}$ on the ledger.
- v_{mint} is the total value of $\mathcal{A}'s$ mint transactions.
- $v_{\text{ADDR} \to \mathcal{A}}$ is the total value of payment received by $\mathcal{A}$ from addresses in ADDR
- $v_{\mathcal{A} \to \text{ADDR}}$ is the total value of payments sent by the adversary to the addresses in ADDR.

Definition 3.2 A DAP scheme Π=(Setup, CreateAddress, Mint, Spend, Receive, verify) is **BAL** secure if the adversary $\mathcal{A}$ wins the game **BAL** only with negligible probability.

$$\Pr[\mathbf{BAL}(\Pi, \mathcal{A}, \lambda) = 1] \leq \text{negl}(\lambda)$$

Ledger Indistinguishability This property implies that no bounded adversary $\mathcal{A}$ will be able to extract any other information from the ledger except what is already publicly revealed. Lelantus transaction reveal the public values of new minted coins, the number of $Spend$ transaction inputs and outputs and also the recipient addresses. Ledger-indistinguishability includes the anonymity property defined for Zerocoin in [4] as a special case. It is formalized by the following experiment **L-IND**: First, a challenger samples a random bit b and initializes two DAP scheme oracles $\mathcal{O}_0^{DAP}$ and $\mathcal{O}_1^{DAP}$ maintaining proprietary ledgers L_0 and L_1. Throughout, the challenger allows $\mathcal{A}$ to issue queries to both oracles, thus controlling the behavior of honest parties on L_0 and L_1. At each round of the experiment, the adversary issues queries in pairs Q, Q' which are of the same query type. If the query type is **CreateAddress**, then the same address is generated at both oracles. If it is either **Mint**, **Spend** or **Receive**, then Q is forwarded to L_0 and Q' to L_1. For **Insert** queries, the query Q is forwarded to L_b and Q' is forwarded to L_{1-b}. An important restriction is set that the adversarys queries should maintain the public consistency and the consistency of $\mathcal{A}$'s view of both ledgers. For example, all public values for $Mint$ and $Spend$ queries, also the number of inputs and outputs of $Spend$ queries must be the same. The coin values sent to addresses controlled by the adversary in both queries should match. After each round, the challenger provides the adversary with the view of both ledgers, but in randomized order: $L_{\text{left}} := L_b$ and $L_{\text{right}} := L_{1-b}$. The adversary's goal is to distinguish whether the view he sees corresponds to

$(L_{\text{left}}, L_{\text{right}}) = (L_0, L_1)$ or to $(L_{\text{left}}, L_{\text{right}}) = (L_1, L_0)$. At the conclusion of the experiment, $\mathcal{A}$ outputs a guess b' and wins if $b' = b$. Ledger indistinguishability requires that $\mathcal{A}$ wins **L-IND** with probability at most negligibly greater than 1/2.

Definition 3.3 A DAP scheme Π=(Setup, CreateAddress, Mint, Spend, Receive, Verify) is **L-IND** secure if the adversary $\mathcal{A}$ wins the game **L-IND** only with negligible probability.

$$\Pr[\textbf{L-IND}(\Pi, \mathcal{A}, \lambda) = 1] - \frac{1}{2} \leq \text{negl}(\lambda)$$

Transaction Non-Malleability This property requires that no bounded adversary $\mathcal{A}$ can alter a valid tx_{spend} transaction data. Transaction non-malleability prevents malicious attackers from modifying others transactions by re-addressing the outputs of a *Spend* transaction before the transaction is added to the ledger. Note that the non-malleability of the $Mint$ transactions is assured with help of the basecoin layer signatures put on the UTXO spending transactions. Following to the Zerocash definition, transaction non-malleability is formalized by an experiment **TR-NM**, in which $\mathcal{A}$ adaptively interacts with the oracle $\mathcal{O}^{\text{DAP}}$ and then outputs a spend transaction tx^*. Letting T denote the set of all *Spend* transactions returned by $\mathcal{O}^{\text{DAP}}$, and L denote the final ledger, $\mathcal{O}^{\text{DAP}}$ wins the game if there exists $tx \in T$, such that (i) $tx^* \neq tx$; (ii) tx^* reveals the serial number contained in the tx; and (iii) both tx and tx^* are valid transactions with respect to the ledger L' containing all transactions preceding tx on L. In other words, $\mathcal{A}$ wins the game if tx^* manages to modify some previous *Spend* transaction tx to spend the same coin in a different way. Transaction non-malleability requires that $\mathcal{A}$ wins **TR-NM** with only negligible probability.

Definition 3.4 A DAP scheme Π=(Setup, CreateAddress, Mint, Spend, Receive, verify) is **TR-NM** secure if the adversary $\mathcal{A}$ wins the game **TR-NM** only with negligible probability.

$$\Pr[\textbf{TR-NM}(\Pi, \mathcal{A}, \lambda) = 1] \leq \text{negl}(\lambda)$$

We also require the DAP scheme to be *complete* which implies that any unspent coin on the ledger can be spent. This property means that if the coin commitment C appears on the ledger L but the coin's serial number sn does not appear in the list **S-Pool**, then the coin C can be spent using a valid *Spend* transaction. This security property immediately follows from the completeness properties of the underlying one-out-of-many proof.

We will provide formal proofs for the satisfaction of all three security properties in the Appendix A.

IV. Algorithm Constructions

In this section we provide detailed description of the DAP scheme algorithms.

A. Setup

In our setup the public parameters pp are comprised of the corresponding public parameters of the commitment scheme, range proof, one-out-of-many proof, the key-private public key encryption and digital signatures schemes. All this algorithms operate in the specified prime-order group G with a scalar field Z_p. Setup also samples a cryptographically secure hash function $\mathcal{H} : \{0,1\}^* \to Z_p$.

Setup Algorithm

Inputs: Security parameter λ.
Outputs: Public parameters pp, a cryptographic hash function $\mathcal{H}$.

```
1) Sample a prime order group $G$ with a scalar field $Z_p$.
2) Compute $pp_{\text{ck}} = (g, h, f)$ as a tuple of three independent generator points.
3) Compute $pp_{\text{rp}}$ = $\text{Setup}_{\text{bp}}$.
4) Compute $pp_{\text{ooon}} = \text{Setup}_{\text{ooon}}$.
5) Compute $pp_{\text{enc}} = \text{Setup}_{\text{enc}}$.
6) Compute $pp_{\text{sig}} = \text{Setup}_{\text{sig}}$.
7) Choose the cryptographically secure hash function $\mathcal{H} : \{0,1\}^* \to Z_p$
8) Set $pp = (G, Z_p, pp_{\text{ck}}, pp_{\text{rp}}, pp_{\text{ooon}}, pp_{\text{enc}}, pp_{\text{sig}})$
9) Output $pp$ and $\mathcal{H}$.
```

Note that public parameters of different algorithms may share common fixed generator points.

B. CreateAddress

In our setup we skip the description of the base layer addresses which are used to transparently exchange base coins (e.g. Bitcoin) and only discuss how the shielded layer addresses are generated and used for the confidential payments.

CreateAddress Algorithm

Inputs: Security parameter λ, public parameters pp
Outputs: Address key pair $(\text{addr}_{pk}, \text{addr}_{sk})$

```
1) Generate $s, r, k \leftarrow Z_p$
2) Compute $(k, P) = \mathcal{K}_{enc}(pp_{\text{enc}})$
3) Compute $Q = g^s f^r$
4) Compute a proof of representation of the value $Q$ with respect to the generators $g$ and $f$: $\pi_Q = \pi_{\text{dlrep}}(Q; g, f)$
5) Set $\text{addr}_{\text{pk}} = (P, Q, \pi_Q)$
6) Set $\text{addr}_{\text{sk}} = (k, s, r)$
7) Output $(\text{addr}_{\text{pk}}$ , $\text{addr}_{\text{sk}})$
```

C. Mint

The **Mint** algorithm generates the transaction data tx_{mint} which is used to initiate a $Mint$ transaction.

Mint Algorithm
Inputs: Public parameters pp, coin value $v \in [0, v_{max})$, recipient public address addr_{pk}
Outputs: Mint transaction data tx_{mint}.

1) Parse the addr$_{\text{pk}}$ as (P, Q, π_Q)
2) Generates $x \leftarrow_R Z_p$
3) Computes the coin commitment as $C = Q^x h^v$
4) Computes $D = \mathcal{E}_{enc}(P; x)$
5) Set $* = (v, C, D, \text{addr}_{\text{pk}})$
6) Computes a Schnorr signature $\sigma = \mathcal{S}_{sign}(x; *)$ which can be verified by the public key Q^x.
7) Outputs tx$_{\text{mint}} = (*, \sigma)$

As $Q = g^s f^r$, where s and r are part of the secret address, the output coin commitment will be a double blinded commitment with respect to the generators g, h and f: $C = g^{xs} h^v f^{xr} = Comm(xs, v, xr)$. The blinding component f^{xr} ensures it is computationally unfeasible to identify the commitment given the values g^{xs} and v. The minted value v can be a sum of one or more transparent inputs from the blockchain base layer. Each base layer input spending should be associated with a valid proof of ownership for the spent assets, e.g. with a digital signature σ_i which can be verified through the corresponding UTXO's public key. With current design we support only mint transactions with a single output, but it is possible to support generation of multiple output coins as well. Note that all network participants can check if the output coin C is opening to the public value v by computing the value $C \cdot h^{-v} = Q^x$ and using the result as a verification key to check the provided signature σ.

D. Spend

Spend algorithm generates the tx$_{\text{spend}}$ data which consumes input coins anonymously and generate new fresh output coins. We assume that *Spend* transaction can spend $old \geq 1$ coins $C^i_1, \ldots, C^i_{old}$ and create $new \geq 0$ fresh coins $C^o_1, \ldots, C^o_{new}$. Here $C^i_k = g^{s^i_k} h^{v^i_k} f^{r^i_k}$ and $C^o_l = g^{s^o_l} h^{v^o_l} f^{r^o_l}$. **Spend** also outputs a public net value $v_{out} \geq 0$ and there is a public transaction fee denoted by fee. Before describing how the transaction data is generated, let's note that the transaction legitimacy proof should ensure all network participants that the following holds.

- All *old* spent input coins are valid spends and owned by the sender. Proving this should not reveal any information about the spent coin value or its origin.
- Output commitments do not contain negative values: $\forall j \in 1, \ldots, new : v^o_j > 0$
- The transaction is balanced which means $v^i_1 + \ldots + v^i_{old} = v^o_1 + \ldots + v^o_{new} + v_{out} + fee.$

For each spent input coin a separate Zerocoin like spending process is executed. The user first reveales the coin's serial number sn, next subtracts it from all coin commitments from the corresponding anonymity set **C-Pool** and gets a new set of commitments, and then provides a zero-knowledge proof of a knowledge of one double-blinded commitment in the resulted set of commitments opening to zero.

Our **Spend** algorithm construction leverages modified one-out-of-many proofs for double-blinded commitments which support the anonymous spending and balance proof generation processes. Next we discuss the modified protocol in details.

One-out-of-many proofs for a double-blinded commitment opening to zero. The protocol described below is the modified version of the scheme described in [5]. Assuming that $N = n^m$, the idea behind the 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. Observe that $\sigma_{l,i} = \prod_{j=0}^{m-1} \sigma_{l_j, i_j}$ 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 Figure 2.

This novel construction of one-out-of-many proofs for double-blinded commitments differs from the original protocol described in [5] in a few 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. Then, instead of revealing the values G_k as a product of $\prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot Comm(0, \rho_k, \tau_k)$, this product is split and we explicitly reveal a 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 in turn are further blinded via extra random factors γ_k. The G_k values are respectively multiplied with the inverse of f^{γ_k} to ensure these random factors will be neutralized in the product $G_k \cdot Q_k$ during the proof verification process. In the appendix, we will provide a formal proof for the following lemma.

Lemma 1: *The Σ-protocol for knowledge of one-out-of-many 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.*

Next we provide detail description of the **Spend** algorithm.

Spend Algorithm
Inputs:

- Transaction input coins $\{C^i_1, \ldots, C^i_{old}\}$ and their corresponding witnesses

```
$P(gk, crs, (C_0, \dots, C_{N-1}), l, v, R)$
Compute
  $r_A, r_B, r_C, r_D, a_{j,1}, \dots, a_{j,n-1} \leftarrow_R Z_q$
  for $j \in [0, \cdots, m-1]$
      $a_{j,0} = -\sum_{i=1}^{n-1} a_{j,i}$
  $B := Com_{ck}(\sigma_{l_0,0}, \dots, \sigma_{l_{m-1},n-1}; r_B)$
  $A := Com_{ck}(a_{0,0}, \dots, a_{m-1,n-1}; r_A)$
  $C := Com_{ck}(\{a_{j,i}(1 - 2\sigma_{l_j,i})\}_{j,i=0}^{m-1,n-1}; r_C)$
  $D := Com_{ck}(-a_{0,0}^2, \dots, -a_{m-1,n-1}^2; r_D)$
  $For \quad k \in 0, ..., m-1$
      $\rho_k, \tau_k, \gamma_k \leftarrow_R Z_q$
      $Q_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot h_2^{-\gamma_k}$
        computing $p_{i,k}$ as is described above
      $G_k = h_2^{\gamma_k} \cdot Comm(0, \rho_k, \tau_k)$

                    $A, B, C, D, \{G_K, Q_K\}_{k=0}^{m-1}$  $\longrightarrow$
                    $x \leftarrow \{0,1\}^{\lambda}$  $\longleftarrow$

      $\forall j \in [0, m-1], i \in [1, n-1]$
      $f_{j,i} = \sigma_{l_j i} x + a_{j,i}$
  $z_A = r_B \cdot x + r_A$
  $z_C = r_C \cdot x + r_D$
  $z_v = v \cdot x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k$
  $z_R = R \cdot x^m - \sum_{k=0}^{m-1} \tau_k \cdot x^k$

                    $f_{0,1}, \dots f_{m-1,n-1}$
                    $z_A, z_C, z_v, z_R$  $\longrightarrow$

$v(gk, crs, (C_0, \dots, C_{N-1}))$
Accept if and only if
  The values
    $A, B, C, D, G_0, Q_0 \dots, G_{m-1}, Q_{m-1} \in G$
    $\{f_{j,i}\}_{j,i=0,1}^{m-1,n-1}, z_A, z_C, z_v, z_R \in Z_q$

    $\forall j : f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i}$

    $B^x A = Com(f_{0,0}, \dots, \dots f_{m-1,n-1}; z_A)$
    $C^x D = Com(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C)$

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

Fig. 2. One-out-of-many proof for double-blinded commitments

$\{(l_1, S_1^i, v_1^i, R_1^i), \dots, (l_{old}, S_{old}^i, v_{old}^i, R_{old}^i)\}$. Here each l_k is the k-th input coin's index in the referred anonymity set **C-Pool**.

- The net output value v_{out} and the transaction fee fee.
- The output coin values $v_1^o, \dots, v_{new}^o$ and the corresponding output public addresses $\mathrm{addr}_{\mathrm{pk}_1}, \dots, \mathrm{addr}_{\mathrm{pk}_{new}}$.

Outputs:Spend transaction data $\mathrm{tx}_{\mathrm{spend}}$:

1) Parse $\mathrm{addr}_{\mathrm{pk}_i} = (P_i, Q_i, \pi_{Q_i})$ for each $i \in 1, \dots, new$.
2) Sample $x_1, \dots, x_{new} \leftarrow_R Z_P$.
3) Compute output coins: $C_1^o = Q_1^{x_1} h^{v_1^o}, \dots, C_{new}^o = Q_{new}^{x_{new}} h^{v_{new}^o}$.
4) Compute $D_1 = \mathcal{E}_{enc}(P_1; x_1 || v_1^o)$, $\dots$, $D_{new} = \mathcal{E}_{enc}(P_{new}; x_{new} || v_{new}^o)$. These ciphertexts enable the intended recipients to receive coin secrets directly from the ledger.
5) Compute a separate range proof for each output: $\pi_{\mathrm{range}_1} = \pi_{\mathrm{range}}(C_1^o; Q_1, h), \dots, \pi_{\mathrm{range}_{\mathrm{new}}} = \pi_{\mathrm{range}}(C_{\mathrm{new}}^o; Q_{\mathrm{new}}, h)$
6) For each input coin $C_1^i, \dots, C_{old}^i$:
   a) Publish the coin serial number computed as $\mathrm{sn}_k = g^{S_k^i}$.
   b) Compute $\textbf{C-Pool}_k = (C_0', C_1', \dots, C_{N-1}')$ by homomorphically subtracting the revealed serial number sn_k from all coin commitments in the original anonymity set **C-Pool**: $C_j' = C_j \cdot Comm(\mathrm{sn}_k^{-1}, 0, 0)$.
   c) Generate a non-interactive π_{ooon_k}-proof of knowledge of one double-blinded commitment from the set $\textbf{C-Pool}_k$ is opening to zero. All old non-interactive proofs $\pi_{\mathrm{ooon}_1}, \dots, \pi_{\mathrm{ooon}_{old}}$ are using a common verifier challenge x which is computed through Fiat-Shamir heuristic over all initial statements of old proofs.
7) Compute the transaction balance proof.
   The balance proof represents a non-interactive proof of representation of some value $\frac{A}{B}$ with respect to the generator elements $Q_1, \dots, Q_{\mathrm{new}}$ and f:
   $$\pi_{\mathrm{balance}} = \pi_{\mathrm{dlrep}}\Big(\frac{A}{B}; Q_1, \dots, Q_{new}, h\Big)$$
   The elements A and B are computed over the public transaction data as will describe below.
8) Set
   $$* = \Big\{\{(sn_1, \pi_{\mathrm{ooon}_1}) \dots, (sn_{\mathrm{old}}, \pi_{\mathrm{ooon}_{old}})\},$$
   $$\pi_{\mathrm{balance}}, v_{out}, fee, (\mathrm{addr}_{\mathrm{pk}_1}, C_1^o, D_1, \pi_{\mathrm{range}_1}),$$
   $$\dots, (\mathrm{addr}_{\mathrm{pk}_{\mathrm{new}}}, C_{new}^o, D_{new}, \pi_{\mathrm{range}_{\mathrm{new}}})\Big\}$$
9) For each $k \in 1, \dots, old$ computes $\sigma_k = \mathcal{S}_{sign}(S_k^i; *)$. Note that each σ_k can be verified with the corresponding public key $sn_k = g^{S_k^i}$ which are part of the public transaction data $\mathrm{tx}_{\mathrm{spend}}$.
10) Set $\mathrm{tx}_{\mathrm{spend}} = (*, \sigma_1, \dots, \sigma_{old})$

The values A and B used at Step (7) of the **Spend** algorithm to construct the balance proof are computed as follows: Given the public output coins $C_1^o, \dots, C_{new}^o$, the transparent output value v_{out} and the transaction fee fee, also the verifier challenge variable x used for constructing all old non-interactive π_{ooon}-proofs, the element A is computed as

$$\begin{aligned}\mathbf{A} : &= (C_1^o \cdot \dots \cdot C_{new}^o)^{x^m} h^{(v_{out}+fee)x^m} = \\ &= (Q_1^{x_1} \cdot \dots \cdot Q_{new}^{x_{new}})^{x^m} h^{(v_{out}+v_1^o+\dots+v_{new}^o+fee)x^m}\end{aligned} \quad (1)$$

For each transaction data tx_{spend}, the provided $(\pi_{ooon_1}, \dots, \pi_{ooon_{old}})$ proofs can be parsed to extract the following elements:

$$(z_{v_1}, z_{R_1}, G_0^1, \dots, G_{m-1}^1), \dots, (z_{v_{old}}, z_{R_{old}}, G_0^{old}, \dots, G_{m-1}^{old})$$

According to the Fig. 2, for each $t \in 1, ..., old$

$$z_{v_t} = v_t^i \cdot x^m - \sum_{k=0}^{m-1} \rho_k^t x^k, \quad z_{R_t} = R_t^i \cdot x^m - \sum_{k=0}^{m-1} \tau_k^t x^k$$

and

$$\begin{aligned}G_0^t &= Comm(0, \rho_0^t, \gamma_0^t + \tau_0^t), \cdots, \\ G_{m-1}^t &= Comm(0, \rho_{m-1}^t, \gamma_{m-1}^t + \tau_{m-1}^t)\end{aligned}$$

Given the elements $(z_{v_1}, \dots, z_{v_{old}})$, $(z_{R_1}, \dots, z_{R_{old}})$ and $\{Comm(0, \rho_k^t, \tau_k^t + \gamma_k^t)\}_{k=0}^{m-1}$, **B** is computed as

$$\begin{aligned}\mathbf{B} =& Comm(0; z_{v_1} + \dots + z_{v_{old}}, z_{R_1} + \dots + z_{R_{old}}) \cdot \\ & \prod_{t=1}^{old} \Big(\prod_{k=0}^{m-1} (Comm(0; \rho_k^t, \gamma_k^t + \tau_k^t))^{x^k} \Big) = \\ & h^{(v_1^i + \dots v_{old}^i)x^m} f^{\sum_{t=1}^{old}(R_t^I \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k)}\end{aligned} \quad (2)$$

All network participants can independently compute A and B based only on the public ledger data. Obviously,

$$\frac{\mathbf{A}}{\mathbf{B}} = \frac{h^{(v_{out}+v_1^o+\dots+v_{new}^o+fee)x^m} \cdot Q_1^{(x_1 \cdot x^m)} \cdot \dots \cdot Q_{new}^{(x_{new} \cdot x^m)}}{h^{(v_1^i+\dots+v_{old}^i)x^m} \cdot f^{\sum_{t=1}^{old}\left(R_t^I \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k\right)}} \quad (3)$$

If the transaction balance holds, the output values and the transaction fee sum up with the spent inputs values. In that case the h exponents in the equation (3) will cancel each other out and the value $\frac{A}{B}$ will be represented only with respect to the generators $Q_1, \dots, Q_{new}$ and f. Hence, for providing a balance proof, it is sufficient for the transaction owner to provide a proof of representation of the value $\frac{A}{B}$ with respect to the generators $(Q_1, \dots, Q_{new}, f)$ which is done at step (7) of the **Spend** algorithm.

E. Verify Algorithm

This algorithm enables the network participants to verify legitimacy of any $Mint$ or $Spend$ transaction published on the ledger. It returns a Boolean value indicating the verification success.

Verify Algorithm

Inputs: A transaction data tx.
Outputs: A Boolean value. 1 means successful verification and 0 will indicate failure.

1) If tx = tx_{mint}
   - Parse the transaction data $\text{tx}_{\text{mint}} = (*, \sigma) = (C, v, D, \text{addr}_{\text{pk}} = (Q, P, \pi_Q), \sigma))$
   - Compute $pk_{sig} = C \cdot h^{-v}$
   - Set $b = \mathcal{V}_{sig}(pk_{sig}; *, \sigma)$
   - Output b.

   If the minted coin value matches to the input v, then according to the Step (3) of the **MINT** algorithm, the value pk_{sig} will be equal to Q^x and can be used as the verification public key to check the signature generated with the witness x.

2) If tx = tx_{spend}
   - Parse the transaction data
   $$\begin{aligned}\text{tx}_{\text{spend}} = \Big\{ &\{(sn_1, \pi_{\text{ooon}_1}) \dots, (sn_{\text{old}}, \pi_{\text{ooon}_{old}})\}, \\ &\pi_{\text{balance}}, v_{out}, fee, (\text{addr}_{\text{pk}_1}, C_1^o, D_1, \pi_{\text{range}_1}), \\ &\dots, (\text{addr}_{\text{pk}_{\text{new}}}, C_{new}^o, D_{new}, \pi_{\text{range}_{\text{new}}}), (\sigma_1, \dots, \sigma_{old}) \Big\} \\ &= (*, \sigma_1, \dots, \sigma_{old})\end{aligned}$$
   - For all $k \in \{1, \dots, old\}$
     a) Compute $\text{C-Pool}_k = \{\frac{C_0}{sn_k}, \dots, \frac{C_{N-1}}{sn_k}\}$
     b) verify the π_{ooon_k} proof validity with respect to the list of commitments C-Pool_k. Output 0, if the verification fails.
     c) Use sn_i as the verification public key to check the signature $\mathcal{V}_{sig}(\text{sn}_i, *, \sigma_k)$. Output 0, if the verification fails.
   - For all $j \in \{1, \dots, new\}$
     a) Check if the recipient address $\text{addr}_{\text{pk}_j} = (Q_j, P_J, \pi_{Q_j})$ is well formed by verifying the corresponding proof π_{Q_j}. Output 0, if the verification fails.
     b) Check if the output coin C_j^o contains a non-negative value by verifying the provided range proof π_{range_j} with respect to the generator points Q_j and h. Output 0, if the verification fails.
   - Check if the transaction is balanced as follow:
     a) Compute the values A and B as defined by the Equations 1 and 2.
     b) Check the provided proof of representation $\pi_{balance}$ for the value $\frac{A}{B}$ with respect to the generator points $(Q_1, \dots, Q_{new}, h)$. Output 0, if the verification fails.
   - Output 1.

F. Receive Algorithm

This algorithm enables users to scan the ledger and recover all coin information sent to their public addresses.

Receive Algorithm
Inputs: The blockchain ledger L, a public and private address pair $\text{addr}_{\text{pk}}, \text{addr}_{\text{sk}}$.
Outputs: A set of new received coins $\text{Coins} = \{l_t, S_t, v_t, R_t\}_{t=0...}$.

1) Parse the public address $\text{addr}_{pk} = (P_u, Q_u, \pi_{Q_u})$ and $\text{addr}_{sk} = (k_u, s_u, r_u)$.
2) Parse the anonymity set **C-Pool** $= (C_0, C_1, \dots, C_{N-1})$.
3) Set Coins $= \{\}$.
4) For each tx_{mint} check if its output coin commitment is addressed to Q_u. If so, then
   a) Parse the transaction data $\text{tx}_{\text{mint}} = (*, \sigma) = (C, v, D, P, Q_u, \pi_Q, \sigma))$
   b) Identify the C's index l in the anonymity set **C-Pool**. This can be done by a simple search of C in the **C-Pool**.
   c) Compute $x = \mathcal{D}_{enc}(k_u, D)$
   d) If $C = Q^x h^v = g^{xs_u} f^{xr_u} h^v$ then add the extracted coin data to the set of received coins: Coins $=$ Coins $\cup\ (l, xs_u, v, xr_u)$.
5) For each tx_{spend} on the ledger, check if it contains an output coin commitment addressed to Q_u. If it does, then
   a) Extract the identified coin data (P, Q_u, π_Q, C^0, D) from the tx_{spend}.
   b) Identify the coin index l in the anonymity set **C-Pool**.
   c) Compute $\mathcal{D}_{end}(k, D)$ and parse the result as (x, v) to extract the encrypted values v and x.
   d) If $C = Q^x h^v = g^{xs_u} f^{xr_u} h^v$ then add the extracted coin data to the set of received coins: Coins $=$ Coins $\cup\ (l, S, v, R)$ to the list of received coins where $S = xs_u$ and $R = xr_u$.
6) Output the list of all received coins Coins.

The user can later spend any received coin using the corresponding private coin data (l, S, v, R).

V. Batch verification of one-out-of-many proofs

In the blockchain application, the verifier will have to verify multiple one-out-of-many 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 to the simultaneous verification of multiple one-out-of-many proofs in the Lelantus setup.

Let us assume the verifier has to verify multiple *Spend* transactions which jointly spend M different coins from the **C-Pool**: $(C_0, C_1, .C_{N-1})$. According to the **Spend** algorithm each transaction $\text{tx}^t_{\text{spend}}$ contains the coin's serial number sn_t and an associated proof π^t_{ooon}. The verifier have to first compute new set of commitments $\{C^t_i = \frac{C_i}{sn_t}\}_{i=0}^{N-1}$ and then check the validity of the associated proof π^t_{ooon} with respect to the resulted set $(C^t_0, C^t_1, \dots, C^t_{N-1})$. The verification of a single one-out-of-many proof π_{ooon} boils down to a large multi-exponentiation of the following form.

$$\prod_{i=0}^{N-1} C_i^{t\, \prod_{j=0}^{m-1} f^t_{j,i_j}} \cdot \prod_{k=0}^{m-1} (G^t_k \cdot Q^t_k)^{-x^k} \equiv Comm(0, z^t_v, z^t_R)$$

which requires N exponentiation operations with transaction specific base points $(C^t_0, C^t_1, \dots, C^t_{N-1})$. Taking into account the fact that for the t-th transaction $C^t_i = \frac{C_i}{\text{sn}_t}$ and the serial number $\text{sn}_t = g^{S_t}$ is part of the tx_{spend} transaction public transcript, we can rewrite the equation above as

$$\prod_{i=0}^{N-1} C_i^{f^t_i} \equiv \frac{E_t}{D_t} \cdot \text{sn}_t^{\sum_{i=0}^{N-1} f^t_i}$$

where

$$f^t_i := \prod_{j=0}^{m-1} f^t_{j,i_j}; D_t := \prod_{k=0}^{m-1} (G^t_k \cdot Q^t_k)^{-x^k}; E_t := Comm(0, z^t_v, z^t_R)$$

So for all transactions referring to the same initial anonymity set **C-Pool**, the verification equation can be turned into a multi-exponentiation operation with common based points, which in turn will yield to huge computational benefits, as:

- The verifier can perform batch verification of the one-out-of-many proofs.
- The batch verification can be further accelerated by pre-computing the exponent values of all base points C_i (resulting in another 25-60% performance optimization)

In order to perform verification of M spend proofs in batch, the verifier can generate M random values $y_1, \dots, y_M$ and compute the following product

$$\prod_{t=1}^{M}\Big(\prod_{i=0}^{N} C_i^{f^t_i}\Big)^{y_t} = \prod_{t=1}^{M}\Big(\frac{E_t}{D_t} \cdot \text{sn}_t^{\sum_{i=0}^{N} f^t_i}\Big)^{y_t}$$

which in turn is equivalent to verifying that

$$\prod_{i=0}^{N-1} C_i^{\sum_{t=1}^{M} y_t \cdot f^t_i} = \prod_{t=1}^{M}\Big(\frac{E_t}{D_t} \cdot \text{sn}_t^{\sum_{i=0}^{N} f^t_i}\Big)^{y_t}$$

This helps to save N exponentiation for each extra one-out-of-many proof verification resulting to highly efficient batch verification process. Without going into the full details it is worth mentioning that other parts of the transaction validity proofs also could be verified in batches. The transaction balance proofs represented through proofs of discrete logarithm representation, range proofs and also the remaining checks of the one-out-of-many proofs are all relying on multi-exponentiation operations. They all can be concatenated together and checked simultaneously through a larger multi-exponentiation operation in a more efficient way.

VI. 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.

TABLE II
BATCH VERIFICATION PERFORMANCE FOR ONE-OUT-OF-MANY PROOFS FOR VARIOUS ANONYMITY SET SIZES

$\|A\|$	Batch Size	Proof Size	Proving Time(ms)	verification Time(ms)
16384	1	1412	1199	234
	5			291
	10			461
	50			649
	100			1070
65536	1	2456	2378	882
	5			975
	10			1104
	50			2006
	100			2993
	1000			22366
100000	1	2044	4416	1366
	5			1504
	10			1683
	50			3129
	100			4947
	1000			37798
262144	1	2016	14098	3542
	5			3989
	10			4531
	50			8970
	100			13979
	1000			109964

The proof can be computed using $mN+2mn+2m+6$ group exponentiation, as the computation of the values A, B, C and D from the Fig. requires $2mn+4$ exponentiation operations. 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 exponentiation as follows: $N+2m+2$ exponentiation for the last equation implying big multi-exponentiation, and $2mn+4$ for the remaining computations. 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 Table II we bring the proof size and performance parameters for different anonymity set sizes, as the overall proof and verification time for spend transactions will be dominated by the one-out-of-many proof components. All experiments were performed on an Intel I7-4870HQ system with a 2.50 GHz processor. The Table II illustrates how the average cost of a single one-out-of-many proof verification can be as low as 10ms in the anonymity set of size 16384 and as low as 37m in the anonymity set of size 100000.

VII. 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.

Acknowledgment

The author thanks Jens Groth for his continuous support and feedback and Sarang Noether for important feedback, analysis and evaluation of different ideas regarding to the direct anonymous payments. The author also thanks to Poramin Insom, Reuben Yap, Levon Petrosyan and the whole Zcoin team for important discussions and support. This protocol has been developed in the scope of Zcoin's (www.zcoin.io) 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] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza. *Zerocash: Decentralized Anonymous Payments from Bitcoin*. Proceedings of the IEEE Symposium on Security Privacy (Oakland) 2014, 459-474, IEEE, 2014.

[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*. ESORICS. LNCS, vol. 9326, pp. 243265. Springer, Heidelberg (2015).

[6] Greg Maxwell. *Confidential transactions* https://people.xiph.org/greg/confidential-values.txt, 2016.

[7] Zcoin: Zcoin's upcoming privacy protocols: https://zcoin.io/lelantus-zcoin/

[8] Monero: A Reasonably Private Digital Currency. https://web.getmonero.org/

[9] Grin: The private and lightweight mimblewimble blockchain. https://grin-tech.org/

[10] MimbleWimble-based Privacy Coin. https://beam.mw/

[11] Lelantus-MW: The symbiosis: https://docs.beam.mw/2019-11-22-Vladislav-Gelfer-at-grincon1.pdf

[12] Ian Miers. *Satoshi Has No Clothes: Failures in On-Chain Privacy*. https://slideslive.com/38911785/satoshi-has-no-clothes-failures-in-onchain-privacy.

[13] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, and G. Maxwell. *Bulletproofs: Short proofs for confidential transactions and more*. Cryptology ePrint Archive, Report 2017/1066, 2017. https://eprint.iacr. org/2017/1066

[14] 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.

[15] Analysis of Monero transaction inputs and outputs. https://github.com/noncesense-research-lab/monero_transaction_io

[16] P. Fauzi, S. Meiklejohn, R. Mercer and C. Orlandi. *Quisquis: A new design for anonymous cryptocurrencies*. Cryptology ePrint Archive, Report 2018/990, 2018. https://eprint.iacr.org/2018/990.

[17] A. Jivanyan and T. Mamikonyan. *Hierarchical One-out-of-Many Proofs With Applications to Blockchain Privacy and Ring Signatures*. Cryptology ePrint Archive: Report 2020/430, 2020. https://eprint.iacr.org/2020/430

[18] Vitalik Buterin. *Hard Problems in Cryptocurrency: Five Years Later*. https://vitalik.ca/general/2019/11/22/progress.html

[19] E. Ben-Sasson, I. Ben-Tov, Y. Horesh and M. Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. https://eprint.iacr.org/2018/046.pdf, 2018.

[20] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, N. Ward. *Aurora: Transparent Succinct Arguments for R1CS*. https://eprint.iacr.org/2018/828.pdf

[21] M. Maller, S. Bowe, M. Kohlweiss, S. Meiklejohn. *Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings*. https://eprint.iacr.org/2019/099.pdf

[22] B. Bnz, B. Fisch and A. Szepieniec. *Transparent snarks from dark compilers*. https://eprint.iacr.org/2019/1229, 2019.

[23] S. Bowe, J. Grigg, and D. Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. 2019. URL: https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.
[24] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, Cambridge, Massachusetts, August 2000. ISBN 0-262-02491-8.
[25] Yehuda Lindell. *Parallel coin-tossing and constant-round secure two-party computation*. Journal of Cryptology, 16(3):143-184, 2003.
[26] Mihir Bellare and Phillip Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols*. In ACM CCS, pages 62-73, 1993.
[27] M. Bellare, A. Boldyreva, A. Desai and D. Pointcheval. *Key-privacy in public key encryption*. In Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 01, pages 566582.
[28] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer and M. Virza.*SNARKs for C: verifying program executions succinctly and in zero knowledge*. In CRYPTO, 2013.
[29] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Ocumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive: Report 2019/953.
[30] I. Miers, C. Garman, M. Green and A. D. Rubin. *Zerocoin: Anonymous distributed e-cash from bitcoin*. In IEEE Symposium on Security and Privacy, 2013.
[31] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. *Zcash protocol specification*. URL: https://github.com/zcash/zips/blob/master/protocol/protocol.pdf, 2018.
J. Camenisch, M. Stadler. *Proof Systems for General Statements about Discrete Logarithms*. Report No. 260, March 1997, Dept. of Computer Science, ETH Zurich.Electronically available from: ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/
[32] Ivan Damgard. *Efficient concurrent zero-knowledge in the auxiliary string model*. In EUROCRYPT, volume 1807 of Lecture Notes in Computer Science, pages 418430, 2000.
[33] Mihir Bellare and Phillip Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols*. In ACM CCS, pages 62-73, 1993.
[34] Aram Jivanyan and Sarang Noether. *Enabling Untraceable Anonymous Payments in the Lelantus Protocol*. https://lelantus.io/enabling-untraceable-anonymous-payments.pdf
[35] Jens Groth and Mary Maller. *Snarky signatures: Minimal signatures of knowledge from simulation extractable SNARKs*. Cryptology ePrint Archive, Report 2017/540, 2017. http://eprint.iacr.org/2017/540.
[36] A. Chiesa, M. Green, J. Liu, P. Miao, I. Miers, and P. 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.
[37] https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/

APPENDIX A
DAP SCHEME SECURITY

A DAP scheme $\prod$ = **(Setup, CreateAddress, Mint, Spend, verify, Receive)** is secure if it satisfies the ledger-indistinguishability, transaction non-malleability, and balance properties (defined in Section 3). Here we sketch the formal proofs of our scheme's security properties. We do not formally prove the completeness of our scheme here as it is easy to check that the completeness of our scheme follows from the completeness of the one-out-of-many proof construction and the fact that coin serial numbers are unique.

A. *Proof of Transaction Non-Malleability*

Let $\mathcal{T}$ be the set of all tx_{spend} transactions generated by the $\mathcal{O}^{DAP}$ in response to the adversaries **Spend** queries. Note that $\mathcal{A}$ does own any of the secrets or trapdoors involved in producing these transactions or the relevant secret keys.The adversary $\mathcal{A}$ wins the **TR-NM** experiment described in Section 3, whenever it outputs a valid transaction tx^* such that for some $tx \in \mathcal{T}$ the following holds:

1) $\text{tx}^* \neq \text{tx} verify(pp, \text{tx}^*, L) = 1$ where L' is the part of the ledger preceding tx.
2) A serial number revealed in tx^* is also revealed in tx.

Define $\epsilon = \Pr[\textbf{TR-NM}(\Pi, \mathcal{A}, \lambda) = 1]$. Our goal is to prove that the ϵ is negligible in λ.
Without loss of generality we can assume that both transactions tx^* and tx spend a single input coin. Let's assume the tx^* spends the coin C^* sent to the public address $\text{addr}_{\text{pk}^*}$. Note that the corresponding secret address is a tuple of three scalars $\text{addr}_{\text{sk}} = (k_i, s_i, r_i)$. Let's define

$$\text{tx} = \Big\{(sn, \pi_{\text{ooon}}), (\text{addr}_{\text{pk}}, C^o, D, \pi_{\text{range}}) \\ \pi_{\text{balance}}, v_{out}, fee, \sigma\Big\} = (memo, \sigma)$$

and

$$\text{tx}^* = \Big\{(sn, \pi_{\text{ooon}^*}), (\text{addr}_{\text{pk}^*}, C^{o*}, D^*, \pi^*_{\text{range}}) \\ \pi^*_{\text{balance}}, v^*_{out}, fee^*, \sigma^*\Big\} = (memo^*, \sigma^*)$$

where $memo$ is the transaction data without the signature. Assume by way of contradiction that ϵ is not negligible. According to our construction, the coin spend through tx was computed as $C = h^v Q^x = g^{sx} h^v f^{rx}$, where Q is part of the corresponding public address $\text{addr}_{\text{pk}} = (P = g^k, Q = g^s f^r)$. The serial number sn of the output coin C^o revealed during the tx is of the form $sn = g^S$ which is a valid public key with a witness $S = sx$ and serves as a verification key for the signature σ. S is the product of two secret values s and x, where s is a part of the secret address addr_{sk}. Recall that $\mathcal{A}$ does own any secret value involved in producing either the addr_{sk} or C. Hence the following two disjoint events may occur.

- The adversary knows the serial number witness $S = sx$. As s is part of the recipient's secret address and x is a blinding value both being chosen randomly, S will be indistinguishable from a random number. The possibility of guessing the correct value of S without knowing both s and x is negligible. Note that the public address component Q is a commitment to s via Pedersen commitment scheme, and a ciphertext $D = \mathcal{E}_{enc}(x, P)$ encrypts x with the public key P, which appears on the ledger. Both Q and D appear on the ledger, but as long as the discrete logarithm problem is hard and the public-key encryption scheme is IND-CCA secure, the attacker can extract or guess the correct value of s and x only with a negligible probability.
- The adversary does not know the witness S. As the attacker $\mathcal{A}$ wins, we a have $\text{tx}^* \neq \text{tx}'$ which means $(memo', \sigma') \neq (memo^*, \sigma^*)$. Next, as both transactions are valid we have $\mathcal{V}_{sig}(sn, memo^*, \sigma^*) = 1$ and $\mathcal{V}_{sig}(sn, memo, \sigma) = 1$. This means the adversary could generate a valid signature for the given message under the verification key sn without possessing the signature

key. If the probability of this event is non-negligible, then it will be possible to win the SUF-1CMA game against the signature scheme Sig with non-negligible probability.

B. ***Balance Proof***

Next we will show that our DAP scheme satisfies to the **BAL** security property according to the definition 3.2 Assuming $\epsilon = \Pr[\mathbf{BAL}(\Pi, \mathcal{A}, \lambda) = 1]$, we will prove that ϵ is negligible in λ. In order to prove the balance security, first the **BAL** experiment is modified in a specific way that does not affect the $\mathcal{A}$'s view: For each spend transaction

$$\text{tx}_{\text{spend}} = \Big\{\{(sn_1, \pi_{\text{ooon}_1}) \ldots, (sn_{\text{old}}, \pi_{\text{ooon}_{old}})\}, \pi_{\text{balance}}, v_{out}, fee, (\text{addr}_{\text{pk}_1}, C_1^o, D_1, \pi_{\text{range}_1}), \ldots, (\text{addr}_{\text{pk}_{\text{new}}}, C_{new}^o, D_{new}, \pi_{\text{range}_{\text{new}}}), \sigma_1, \ldots, \sigma_{old}\Big\}$$

on the ledger L, the challenger $\mathcal{C}$ computes the transaction witness $a = (l_1, \ldots, l_{old}; (S_1^i, v_1^i, R_1^i), \ldots, (S_{old}^i, v_{old}^i, R_{old}^i); (S_1^o, v_1^o, R_1^o), \ldots, (S_{new}^o, v_{new}^o, R_{new}^o)$. Assuming a witness data is computed for all $Spend$ transactions inserted either by **Spend** or **Insert** queries, the challenger maintains an augmented ledger $(L, \vec{a})$ where a_i is the witness data for the i-th $Spend$ transaction. The resulted augmented ledger $(L, \vec{a})$ is balanced if the following holds.

1) Each $(\text{tx}_{\text{spend}}, a)$ in $(L, \vec{a})$ contains openings of distinct coin commitments and each commitment is an output of valid $Mint$ or $Spend$ transaction which precedes tx_{spend} on the ledger. This requirement implies that all transactions spend only valid coins and no coin is spent more than once within the same transaction.
2) No two inputs $(\text{tx}_{\text{spend}}, a)$ and $(\text{tx}'_{\text{spend}}, a')$ in $(L, \vec{a})$ contain openings of the same coin commitment. This implies no coin is spent through two different transactions. Together with the first requirement this implies that each coin is spent only once.
3) For each $(\text{tx}_{\text{spend}}, a)$ in $(L, \vec{a})$ which spends old input coins, and for each $k \in 1, \ldots, old$ the following conditions hold:
   - If the i-th input of the tx_{spend} transaction C_k^i is the output of some mint transaction tx_{mint} on L, then the public value v in tx_{mint} is equal to v_k^i.
   - If C_k^i is an output of some $Spend$ transaction $\text{tx}^*_{\text{spend}}$ on L, then the witness a^* contains an opening of that output coin commitment C^{o*} to a value v^* that is equal to v_k^i.

   This implies that no assets have been created out of thin air while spending a coin.
4) Each $(\text{tx}_{\text{spend}}, a)$ in $(L, \vec{a})$ contains openings of $C_1^i, \ldots, C_{old}^i$ and $C_1^o, \ldots, C_{new}^o$ to values $v_1^i, \ldots, v_{old}^i$ and $v_1^o, \ldots, v_{new}^o$ so that the balance condition holds: $v_1^i + \ldots + v_{old}^i = v_1^o + \ldots + v_{new}^o + v_{out} + fee$. This means the transaction balance is preserved by all $Spend$ transactions and no assets have been created out of thin air during new coins generation process.
5) For each $(\text{tx}_{\text{spend}}, a)$ in $(L, \vec{a})$ where the tx_{spend} was inserted by $\mathcal{A}$ through an **Insert** query, it holds that, for each $k \in 1, \ldots, old$, if the C_k^i is the output of an earlier $Mint$ or $Spend$ transaction tx', then the public address addr_{pk} associated with the coin C_k^i in the transaction tx' is not contained in ADDR, which is the set of addresses belonging to honest users and returned by a **CreateAddress** queries.This means that the adversary can not spent a coin created by honest parties.

If these five conditions jointly hold than obviously the $\mathcal{A}$ did not spend or control more money than it was previously minted or paid to one of his addresses: $v_{\text{mint}} + v_{\text{ADDR} \to \mathcal{A}} \geq v_{\text{Unspent}} + v_{\text{output}} + v_{\mathcal{A} \to \text{ADDR}}$. Therefore, in order to prove that the DAP scheme is **BAL** secure it suffices to prove that the augmented ledger is balanced with all but negligible probability.

By way of contradiction let's assume the the adversary $\mathcal{A}$ produces a non-balanced augmented ledger $(L, \vec{a})$ with non-negligible probability which means that one or more of the five conditions described above have been violated with non-negligible probability. We show how this leads to a contradiction.

$\mathcal{A}$ **violates Condition 1:** Suppose that $\Pr[\mathcal{A}$ wins but violates the Condition 1$]$ is non-negligible. Each tx_{spend} generated by honest parties satisfies this condition by default thus the violation implies there exist a pair $(\text{tx}_{\text{spend}}, a)$ in $(L, \vec{a})$ where the tx_{spend} was inserted by the $\mathcal{A}$. Assuming tx_{spend} spends old coins then the following holds: (i) $\exists j, k \in \{1, \ldots, old\}$ s.t. $C_j^i = C_k^i$ or (ii) there $\exists k \in \{1, \ldots, old\}$ such that the coin C_k^i has no corresponding output coin commitment in any $Mint$ or $Spend$ transaction tx created before tx_{spend}. Either scenario leads to contradiction as shown below.

- Let's denote $C_j^i = g^{S_j^i} h^{v_j^i} f^{R_j^i}$ and $C_k^i = g^{S_k^i} h^{v_k^i} f^{R_k^i}$. As the tx_{spend} is a valid transaction, all serial numbers revealed by the transaction are distinct so $S_j \neq S_k$. If the condition (i) holds, this means the witness a contains two different openings $(S_j^i, v_j^i, R_j^i) \neq (S_k^i, v_k^i, R_k^i)$ for the same commitment $C_j^i = C_k^i$. This violates the binding property of the commitment scheme.
- Assuming the transaction tx_{spend} spends an input C_k^i which does not appear on the ledger L as an output of any preceding $Mint$ or $Spend$ transaction. The witness a contains an opening (S_k, v_k, R_k) of the commitment C_k^i and its secret index l_k, which identifies the spent coin commitment's position in the set of all previously output coin commitments. As the tx_{spend} is a valid transaction, it reveals a unique serial number sn_k and an one-out-of-many proof π_{ooon_k} which is valid with respect to the anonymity set **C-Pool**. If the coin C_k^i does not appear on the **C-Pool**, this violates the soundness property of the underlying one-out-of-many proof construction.

$\mathcal{A}$ **violates Condition 2:** Suppose that $\Pr[\mathcal{A}$ wins but violates the Condition 2$]$ is non-negligible.

This means the ledger L contains two valid transactions tx_{spend} and $\text{tx}'_{\text{spend}}$ which spend the same coin commitment C but reveal distinct serial numbers $sn = g^S$ and $sn' = g^{S'}$. Similar to the argument above, this means the $\vec{a}$ contains two different openings $(S, v, R) \neq (S', v', R')$ of the same coin commitment which violates the binding property of the commitment scheme.

$\mathcal{A}$ **violates Condition 3:** Suppose that $\Pr[\mathcal{A}$ wins but violates the Condition 3$]$ is non-negligible. In this scenario the ledger L contains a transaction tx_{spend} in which an input coin commitment C opens to a value v but it violates one of the (i) or (ii) requirements of the **Condition 3** with non-negligible probability.

- Assuming the requirement (i) is violated, there is a tx_{mint} transaction, which outputs the the coin commitment C opening to v, but its public input $v_{in} \neq v$. As the tx_{mint} is valid, it contains a proof of knowledge of the exponent value x of the element $\frac{C}{h^{v_{in}}} = Q^x$ with respect to the base point Q. The public address also contains a proof of representation π_Q which proves that Q is represented only with respect to the generators g and f and does not contain any h component. In case C encodes a value v not equal to v_{in}, this breaks the soundness property of the Schnorr's proof of knowledge.
- Assuming the requirement (ii) is violated, there is a $\text{tx}'_{\text{spend}}$ which outputs the coin commitment C with different coin value $v' \neq v$. In this situation the augmented ledger will contain two different openings of the same commitment $(S, v, R) \neq (S', v', R')$ for the same commitment C. Obviously this violates the binding property of the commitment scheme

.

$\mathcal{A}$ **violates Condition 4:** Suppose that $\Pr[\mathcal{A}$ wins but violates the Condition 4$]$ is non-negligible. This means the ledger L contains a tx_{spend} transaction where $v_1^i + \ldots + v_{old}^i \neq v_1^o + \ldots + v_{new}^o + v_{out} + fee$. As the transaction is considered valid, the value $\frac{A}{B}$ computed over the revealed transaction data according to the **Spend** algorithm, will contain a non-zero exponent with respect to the generator h, at the same time the balance proof $\pi_{\text{balance}} = \pi_{\text{DLREP}}(\frac{A}{B}; Q_1, \ldots, Q_{\text{new}}, f)$ will prove the knowledge of $\frac{A}{B}$'s representation with respect to the generators $Q_1, \ldots, Q_{\text{new}}$ and f. Note that each Q_i in turn is provably represented with respect to the generators g and f and does not contain any h component. This violates the soundness property of the underlying Schnorr proof of representation and leads to contradiction.

$\mathcal{A}$ **violates Condition 5** Suppose that $\Pr[\mathcal{A}$ wins but violates the Condition 5$]$ is non-negligible. The violation of the Condition 5 means there exists a tx_{spend} inserted by $\mathcal{A}$ which spends a coin C controlled by an honest party. Without loss of generality we can assume the tx_{spend} spends a single input to a single output and is of the form

$$\text{tx}_{\text{spend}} = \Big\{(sn, \pi_{\text{ooon}}), (\text{addr}^o_{\text{pk}}, C^o, D, \pi_{\text{range}})$$

$$\pi_{\text{balance}}, v_{out}, fee, \sigma\Big\} = (memo^*, \sigma^*)$$

Let's assume that $\text{addr}_{\text{pk}} = (P, Q, \pi_Q)$ is the mentioned public address controlled by an honest party which was used to generate the spent coin C. The corresponding secret address is $\text{addr}_{\text{sk}} = (k, s, r)$ where $Q = g^s f^r$, $P = g^k$. Assuming the coin C was computed as $C = Q^x h^v = g^{xs} h^v f^{xr}$. Based on the soundness property of the one-out-of-many proof, the adversary can generate a valid spend proof only after possessing the corresponding witness data (xs, v, xr). As we have shown in the transaction non-malleability proof, the adversary can get the witness only by breaking the security of the underlying public key encryption and commitment schemes. As long as the discrete logarithm problem is hard, the adversary has nothing but a negligible chance to do this.

C. *Proof of Ledger Indistinguishability*

We will show that our DAP scheme satisfies to the **L-IND** security property according to the Definition 3.3. Assuming $\epsilon = 2 \cdot \Pr[\textbf{L-IND}(\prod, \mathcal{A}, \lambda) = 1] - 1$, we will prove that ϵ is negligible in λ. We define the L-IND experiment below.

Given the DAP scheme $\prod$, the L-IND experiment is an interaction between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ which terminates with a binary output by $\mathcal{C}$. As discussed in the Section 3.B, at the beginning of the experiment, the $\mathcal{C}$ samples $pp \leftarrow Setup(1^\lambda)$ and sends to $\mathcal{A}$; next it samples a random bit $b \in \{0, 1\}$ and initializes two separate DAP oracles $\mathcal{O}_0^{DAP}$ and $\mathcal{O}_1^{DAP}$ each with own separate ledger and internal state. At each consecutive step of the experiment

1) $\mathcal{C}$ provides $\mathcal{A}$ two ledgers $(L_{left} = L_b, L_{rigth} = L_{1-b})$ where L_b and L_{1-b} are the current ledgers of the oracles $\mathcal{O}_b^{DAP}$ and $\mathcal{O}_{1-b}^{DAP}$ respectively.
2) $\mathcal{A}$ sends to $\mathcal{C}$ two queries Q, Q' of the same type (i.e. one of **CreateAddress, Mint, Spend, Receive, Insert**).
    - If the query type is **Insert** or **Mint**, $\mathcal{C}$ forwards Q to L_b and Q' to L_{1-b} enabling the $\mathcal{A}$ to insert own transactions or mint new coins directly in L_{left} and L_{right}.
    - For all queries of type **CreateAddress, Spend** or **Receive**, the $\mathcal{C}$ first checks if two queries Q and Q' are publicly consistent and then forwards Q to $\mathcal{O}_0^{DAP}$ and Q' to $\mathcal{O}_1^{DAP}$. It gets the two oracle answers a_0 and a_1.
3) $\mathcal{C}$ replies to $\mathcal{A}$ with a_b, a_{1-b}.

As the adversary does not know the bit b and the mapping between (L_{left}, L_{right}) and (L_0, L_1), he can not learn weather he elicit the behavior of honest parties on (L_0, L_1) or on (L_1, L_0). At the end of the experiment, $\mathcal{A}$ sends $\mathcal{C}$ a guess $b' \in \{0, 1\}$. $\mathcal{C}$ outputs 1 if $b = b'$, and 0 otherwise. In our experiment we assume no public address is used more than once for either $Mint$ or $Spend$ transactions. We require the queries Q and Q' be publicly consistent as follows: If the query type of Q and Q' is **Receive**,they are publicly consistent by default. If the query type of Q and Q' is **CreateAddress**, both oracles generate the same address. If the query type of Q and Q' is **Mint**, the minted values of both queries should be equal. If the query type of Q and Q' is **Spend** then

- Both Q and Q' should be well-formed and valid which means the referenced input coin commitments must appear in the corresponding ledger's **C-Pool** and be unspent. The transaction balance equation must hold.
- The number of spent coins and output coins must equal. The Recipient address list should be the same for both queries. The public values and the transaction strings in Q and Q' must be equal.
- If the i-th spent input in Q references a coin commitment in L_0 posted by the $\mathcal{A}$ through an **INSERT** query, then the corresponding index in Q' must also reference a coin commitment in L_1 posted by $\mathcal{A}$ through an **INSERT** query and the coin values of these two coins must be equal as well(and vice versa for Q').
- For the j-th output coin in Q, if the corresponding recipient address is not in **ADDR** (i.e., belongs to $\mathcal{A}$), then v_j^o in both Q and Q' must be equal and vice versa for Q'.

In order to prove that $\mathcal{A}$'s advantage in the L-IND experiment is negligible, we first consider a simulation experiment $\mathcal{D}_{sim}$, in which $\mathcal{A}$ interacts with the $\mathcal{C}$ as in the L-IND experiment with the following modifications.

The simulation experiment $\mathcal{D}_{sim}$: Recalling the special honest-verifier zero-knowledge nature of one-out-of-many proofs, Bulletproofs and proofs of discrete logarithm representation we can build a honest verifier zero-knowledge simulator which, given a challenge $x \in \{0,1\}^{\lambda}$, can simulate the validity proof of **Spend** transactions by simulating all its different building blocks.

The simulation. The simulation $\mathcal{D}^{\text{sim}}$ works as follows: As in the original experiment, the $\mathcal{C}$ samples the system parameters $pp = Setup(1^{\lambda})$ and a random bit b, next initializes two separate DAP oracles $\mathcal{O}_0^{\text{DAP}}$ and $\mathcal{O}_1^{\text{DAP}}$. Afterwards $\mathcal{D}^{\text{sim}}$ proceeds in steps. At each step it provides to $\mathcal{A}$ two ledgers $L_{\text{left}} := L_b, L_{\text{right}} := L_{1-b}$ after which $\mathcal{A}$ sends two publicly consistent queries (Q, Q') of the same type. Recall that the queries Q and Q' are publicly consistent with respect to public information and the information related to the addresses controlled by $\mathcal{A}$. Depending on the Q's type, the challenger acts as follows:

- Answering **CreateAddress, Mint, Receive, Insert** queries: In these cases the answer to each query proceeds as in the original L-IND experiment.
- Answering **Spend** queries: In this case Q and Q' have the form (**Spend**, $C_1^i, \text{addr}_{\text{pk},1}^i, \dots, C_{\text{old}}^i, \text{addr}_{\text{pk,old}}^i, v_1^o, \dots, v_{new}^o, \text{addr}_{\text{pk},1}^o, \text{addr}_{\text{pk,new}}^o, v_{\text{out}}$). The challenger generate a simulated tx_{spend} as follows:
  1) For $j \in 1, \dots, \text{old}$
     - Samples a random $\text{sn}_j = g^{s_j}$.
     - Simulates the proof $\pi_{\text{ooon}_j}^{Sim} \leftarrow \text{Sim}(\text{OOON}, \text{sn}_j)$.
     - If $\text{addr}_{\text{pk}_j}^o$ is not generated by one of the previous **CreateAddress** queries and is obviously controlled by the adversary, the simulator calculates the values C_j^o and D_j^o as in the **Spend** algorithm.
     - If $\text{addr}_{\text{pk}_j}^o$ belongs to a honest party and is generated by a previous **CreateAddress**, $\mathcal{C}$
       * Samples a random coin commitment $C_j^o = g^{r_1}h^{r_2}f^{r_3}$
       * Generates new public key encryption key pair $(k', P') \leftarrow \mathcal{K}_{\text{enc}}(pp_{enc})$ and computes $D_j^o = \mathcal{E}_{enc}(P', r)$ for a random r of an appropriate length.
  2) Simulate the balance proof by using a zero-knowledge simulation for the Schnorr proof of representation: $\pi_{\text{balance}}^{Sim} \leftarrow \text{Sim}(\text{DLREP}, \{C_1^o, \dots, C_{\text{new}}^0, \pi_{\text{ooon}_1}^{Sim}, \dots, \pi_{\text{ooon}_{\text{old}}}^{Sim}\})$

C does the same for the second query Q'. We define $\text{Adv}^{\mathcal{D}}$ the advantage of $\mathcal{A}$'s to win the L-IND game in the experiment $\mathcal{D}$.

As can be seen from the simulation described above, all answers sent to $\mathcal{A}$ in $\mathcal{D}^{sim}$ are computed independently of the bit b which makes $\text{Adv}^{\mathcal{D}_{sim}} = 0$. We will prove that $\mathcal{A}$'s advantage in the real L-IND experiment $\mathcal{D}_{real}$ is at most negligibly different than $\mathcal{A}$'s advantage in $\mathcal{D}_{sim}$. In order to prove that let's describe two intermediary experiments, in each of which $\mathcal{C}$ conducts a specific modification of the $\mathcal{D}_{real}$ experiment with $\mathcal{A}$.

Experiment $\mathcal{D}_1$: This experiment modifies $\mathcal{D}_{real}$ by simulating all one-out-of-many proofs, the bulletproofs and the balance proof without using any witnesses. As all these protocols are perfect zero-knowledge, the simulated proofs are indistinguishable from the real proofs generated in $\mathcal{D}_{real}$. Hence the advantage $\text{Adv}^{\mathcal{D}_1} - \text{Adv}^{\mathcal{D}_{real}} = 0$.

Experiment $\mathcal{D}_2$: This experiment modifies $\mathcal{D}_1$ by replacing all ciphertexts corresponding to addresses of honest recipients, in the $\mathcal{D}_1$ experiment with encryption of random strings of appropriate lengths. Assuming the underlying encryption scheme is IND-CCA and IK-CCA secure, we can get that the adversaries advantage in the $\mathcal{D}_2$ experiment is negligibly different from its advantage in the $\mathcal{D}_1$ experiment. The proof logic is identical to the proof of Lemma D.1 in [3]. As $|\text{Adv}^{\mathcal{D}_2} - \text{Adv}^{\mathcal{D}_1}| \leq \text{negl}(1^{\lambda})$ and $\text{Adv}^{\mathcal{D}_1} - \text{Adv}^{\mathcal{D}_{real}} = 0$, we can conclude that $|\text{Adv}^{\mathcal{D}_2} - \text{Adv}^{\mathcal{D}_{real}}| \leq \text{negl}(1^{\lambda})$.

Experiment $\mathcal{D}_{sim}$: The $\mathcal{D}_{sim}$ experiment is already defined above and it differs from $\mathcal{D}_2$ by the fact all output coin commitments corresponding to the public addresses of honest parties are replaced by commitments to random values. We do not give the full argument here, but based on the perfectly hiding property of the commitment scheme, the commitment to random values $C = g^{r_1}h^{r_2}f^{r_3}$ is indistinguishable from a commitment to the given output value v^o computed as $C = Q^x h^{v^o}$. Hence this replacement gives the adversary an

extra zero advantage and $|\mathrm{Adv}^{\mathcal{D}_{sim}} - \mathrm{Adv}^{\mathcal{D}_2}| = 0$.
This finalizes the proof by showing that $|\mathrm{Adv}^{\mathcal{D}_{sim}} - \mathrm{Adv}^{\mathcal{D}_{real}}| \leq \mathrm{negl}(1^\lambda)$.

Appendix B
Security Proofs of one-out-of-many proofs for double-blinded commitments

In this section we formally prove the following lemma.

Lemma 1: *The Σ-protocol for knowledge of one-out-of-many double-blinded commitments opening to zero 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.*

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 can be proven as follow:

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

Next we 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 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, $Q_0, Q_1, \ldots Q_{m-1}, G_1, \ldots G_{m-1}$ 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 $f'_{0,1}, \cdots f'_{m-1,n-1}, z'_A, z'_C$ to one challenge would break the binding property of $B^x A$ and $C^x D$ so the simulation is perfect. It is easy to convert an SHVZK argument into a full zero-knowledge argument secure against arbitrary verifiers in the common reference string model using standard techniques discussed in [32].

Last we prove that the protocol is $(m+1)$-special sound where $N = n^m$. Suppose an adversary can produce $(m+1)$ different accepting responses $\Big((f_{j,i}^{(0)}, z_v^{(0)}, z_R^{(0)}), \ldots, (f_{j,i}^{(m)}, z_v^{(m)}, z_R^{(m)})\Big)$ 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 [?], 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 $R = \sum_{e=0}^{m} \theta_e z_R^e$.