Lelantus: A New Design for Anonymous and Confidential Cryptocurrencies

Aram Jivanyan

Zcoin
aram@zcoin.io
https://zcoin.io

Abstract. For cryptocurrency payments to be truly private, transactions have to have two properties: confidentiality, i.e., hiding the transferred amounts, and anonymity, i.e. hiding the identities of the sender and/or receiver in a transaction.
In this paper, we propose Lelantus, a new decentralized anonymous payment (DAP) protocol that ensures confidential and anonymous blockchain transactions with small transaction sizes, short verification times, and without requiring a trusted setup. It efficiently supports large anonymity sets of size hundred thousand and beyond by providing logarithmic proof sizes and efficient sub-linear verification time of the transactions. We implement Lelantus to measure its performance and show that it is very efficient to support scalable privacy cryptocurrencies.
We also formally prove the security of the proposed protocol characterized by three security properties referred to as *ledger indistinguishability, transaction non-malleability*, and *balance*.
Lelantus design concepts can be used in combination with the MimbleWimble and Confidential Transactions protocols, two other popular blockchain privacy schemes for confidential transactions. A hybrid scheme of Lelantus-MimbleWimble has been developed and implemented into a fully-fledged privacy cryptocurrency in order to provide both confidentiality and strong anonymity of blockchain payments.
As part of our protocol, we also introduce an extension of one-out-of-many proofs for generalized Pedersen commitments and provide formal security proofs for the proposed design, which may be of own interest.

1 Introduction

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. 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. Then we develop a proprietary zero-knowledge balance proof mechanism to ensure that the transaction's input and output values sum up. The transaction balance proof 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 a 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 support direct anonymous payments and 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 is sub-linear due to efficient batch verification methods.

We also analyze and provide formal security proofs for our decentralized 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.
Two major privacy cryptocurrency projects Zcoin[33] and Beam[36] have announced transition to Lelantus and Lelantus-MW.

1.1 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[27], 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 spent coin. The original Zerocoin construction[27] was based on the RSA accumulators and was not efficient from the communication standpoint as each proof consisted of $\sim$25KB. It also required a trust toward the exploited RSA parameters. In [4] authors presented a novel one-out-of-many proof protocol and also proposed an efficient Zerocoin construction 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*) and can be implemented with help of one-out-of-many proofs over the homomorphic Pedersen commitments [8] 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 $C = \mathrm{Com}(sn, r)$ is published to the blockchain and is added to the pool 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 serial number from all coins in the pool and and then 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. For ensuring both transaction confidentiality and anonymity, our first step is to represent coins with generalized Pedersen commitments which commit simultaneously to the coin's serial number and its value. Next we support generic *Spend* transactions which can spend multiple inputs anonymously and output multiple outputs by providing a zero-knowledge balance proof that the transaction inputs and outputs sum up without revealing the input coins origins or the coins amount.

Step 3: Enabling direct anonymous payments. We extend the protocol to support direct anonymous payments which enable users to transfer a value confidentially to the targeted recipients without any future 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. 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 but the verification complexity is linear. It may take hundreds of milliseconds to verify a single spend proof 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. The Table 1 illustrates how Lelantus compares with other confidential payments protocols such as Monero, QuisQuis, Zerocoin and Zerocash. As will be discussed in the Section 5, the reported performance can still be significantly improved, but even the current implementation shows that for a typical 2-input, 2-output transaction our scheme

Table 1. 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	2^{32}	Yes	Relatively New	1	1-10	8
Zerocoin	10000	Yes	Well Studied	25	0.2	200
Lelantus_{1024}	1024	No	Well Studied	2.7	0.27	6.8*
Lelantus_{16384}	16384	No	Well Studied	3.9	2.35	10.2*
Lelantus_{65536}	65536	No	Well Studied	5.6	4.8	52*

provides much stronger anonymity than Monero or QuisQuis at the comparable computational and communication cost. The proof sizes and verification times of SNARK-based constructions are hard to beat but 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.

1.2 Related Works and Comparison

There are many cryptographic constructions which do offer blockchain payment privacy in the UTXO model, such as Zerocash [3], Monero [34], QuisQuis,[10] and Zerocoin [27]. Zerocoin is one of the pioneer technologies providing strong anonymity for Bitcoin transactions but this scheme was computationally not efficient and supported only limited functionality without hiding the transaction amounts and working only with fixed denominated coins. Monero and QuisQuis both enable direct anonymous payments of arbitrary amounts, but they provide relatively weak anonymity guaranties. The anonymity set size in Monero and QuisQuis transactions are 11 and 16 accordingly which is small enough to make different tracking analysis attacks possible[40, 9]. Recent work [11–13] discusses efficient ring signature schemes which aim to improve both the anonymity and complexity of Monero transactions. Unlike Monero or Quisquis, Zerocash supports large anonymity set in a very efficient way, but this hard-to-beat efficiency and advanced privacy features come at the price of reliance on relatively new cryptographic security assumption and a trusted setup procedure, which necessitate the user's trust in the correctness of this setup.
There is a significant amount of active research on the development of efficient zero-knowledge proof systems. 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 [25, 26] which require a trusted setup processing, although the recent work in this direction can support universal and continually updateable trusted setup processes[18]. There have been also designed powerful transparent zero-knowledge proof systems such are Bulletproofs [7], zk-STARKS [16], Aurora [17], Supersonic[19] and Halo[20]. Bulletproofs are ubiquitously used in private digital currency systems[33, 36, 35, 34] for generating efficient range-proofs, but the computational or communication

complexity of these transparent schemes still seems to be beyond the practicality limit when applied to the large-scale confidential payment applications.

1.3 Lelantus and MimbleWimble

MimbleWimble is another popular blockchain privacy protocol implemented by few privacy cryptocurrency projects including Beam[36] and Grin[35]. In MimbleWimble design, 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 history and shrink the ledger size. However this design enables network observers to easily link the transaction inputs and outputs and break the anonymity of users. This remains a major privacy drawback and in order to overcome this limitation, Beam has designed a hybrid scheme of Lelantus and MimbleWimble [37] which provides strong anonymity to MimbleWimble transactions by enabling transactions with hidden origins. In this hybrid approach, users can create coins as generalized Pedersen commitments which hide both the coin value and a unique serial number. These coins are added into the shielded pool and can be spent anonymously using one-out-of-many proofs. The Spend process of the shielded coin first reveals the coin's serial number and outputs a fresh coin represented through a regular Pedersen commitment. The provided zero-knowledge proof shows that the output coin encodes the same value as the spent shielded coin without revealing its origin. This output coin can later participate into a MimbleWimble-style transaction which spends regular coins and outputs either regular or shielded coins by proving that the transaction is still balanced. This hybrid scheme will be deployed by Beam[37]. Provided hybrid design simultaneously ensures strong anonymity and MimbleWimble balance proofs, but still requires interaction between the sender and recipient for finalizing the transaction. Beam also created a Mimblewimble Confidential Lelantus Asset scheme [38] similar to the Confidential Assets proposal by Blockstream [39]. For more details on these constructions we refer to the original papers [37, 38].

2 Cryptographic Background

We denote $R = \{x; w \mid L_R\}$ to be a binary relation for instances x and witnesses w where L_R defines the corresponding language; i.e $L_R = \{x|w : (x, w) \in R\}$. A zero-knowledge proof for the relation R means that the prover knows secret witness w, so that $(x, w) \in L_R$ where x is a public value. We use $x \leftarrow_R T$ for sampling an element x uniformly at random from a set T. 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.

Generalized Pedersen Commitments: Let g and h be random generators of G whose discrete logarithm relationship is unknown. A Pedersen commitment scheme [8] 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 which means it does not leak any information about the committed value. It is also computationally strongly binding under the discrete logarithm assumption which means the commitment can only be opened in one way. For more formal definitions of these security properties and their security proofs we refer to [8]. 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 $g_1, g_2, \ldots, g_n$ 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 utilizes three different group generators g, h, f to commit to the given messages s and t by using the blinding factor r 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 double-blinded commitment.
Note: We will henceforth denote 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 $Comm(s, t, r)$.

One-out-of-many proofs for a commitment opening to zero: One-out-of-many proof is a three step interactive proof of knowledge that one out of N public commitments $\{C_0, \ldots, C_{N-1}\}$ is opening to 0. It has been first introduced in [4] and more efficient construction has been proposed in [5] which significantly reduces the proof sizes and improves the proving complexity. In this paper we introduce a modified version of one-out-of-many proofs for double-blinded commitments described by three algorithms $(\text{Setup}_{\text{ooon}}, \text{Prove}_{\text{ooon}}, \text{Ver}_{\text{ooon}})$ which is a proof of knowledge that the public list of N double-blinded commitments $\{C_0, \ldots, C_{N-1}\}$ includes some $Comm(0, v, r)$ which is a commitment to 0. In the Appendix A, we provide the full description of our scheme and formally prove its security in the same framework used in [4]. In Lelantus design we will use the non-interactive variant of one-out-of-many proofs obtained through Fiat-Shamir heuristic [4] and the proof transcript of non-interactive one-out-of-many proof with respect to the public list of commitments $\{C_0, \ldots, C_{N-1}\}$ is denoted by $\pi_{\text{ooon}}(C_0, \ldots, C_{N-1})$.

Bulletproofs: Bulletproofs are a zero-knowledge interactive proof systems defined by three algorithms $(\text{Setup}_{\text{bp}}, \text{Prove}_{\text{bp}}, \text{Ver}_{\text{bp}})$ for providing short and aggregatable range proofs [7]. 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 $v \in [0, 2^n - 1]$. We will denote the transcript of non-interactive range-proofs for the commitment C with respect to the generators g and h and the specified range $[0, 2^n - 1]$ by $\pi_{\text{range}}(C; g, h, n)$.

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 [21]. A discrete logarithm representation (or DL-REP for short) of $h \in G$ with respect to the generators $g_1, \ldots, g_l \in G$ 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 $l = 1$, 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[21]. 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 [21].An interactive proof of knowledge of representation (also referred as generalized Schnorr proof of knowledge) is a zero-knowledge argument for the relation $R = \{g_1, \ldots, g_k \in G, y;\ a_1, \ldots, a_k \in Z_p \mid y = \prod_{i=1}^{k} g_i^{a_i}\}$. It satisfies to the completeness, special honest verifier zero-knowledge and special soundness properties as is discussed in [21, 4]. We will denote the transcript of non-interactive proof of representation of the value y with respect to given generators $g_1, \ldots, g_l$ $\pi_{\text{dlrep}}(y; g_1, \ldots, g_k)$.

One-time strongly-unforgeable digital signatures: We use a digital signature scheme $Sig = (Setup, \mathcal{K}_{sig}, \mathcal{S}_{Sig}, \mathcal{V}_{Sig})$ described as follows.

- $\text{Setup}_{\text{sig}}(1^\lambda) \rightarrow pp_{sig}$. Given a security parameter λ, *Setup* samples public parameters pp_{sig}.
- $\mathcal{K}_{sig}(pp_{sig}) \rightarrow (pk_{sig}, sk_{sig})$ Given pp_{sig}, $\mathcal{K}_{sig}$ samples a pair of verification and signing keys.
- $\mathcal{S}_{sig}(sk_{sig}, m) \rightarrow \sigma$. Given the key sk_{sig}, $\mathcal{S}_{sig}$ signs the message m to obtain a signature σ.
- $\mathcal{V}_{sig}(pk_{sig}, m, \sigma) \rightarrow b$. Given the 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) [41].

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

- $Setup_{\text{enc}}(1^\lambda) \rightarrow pp_{enc}$. Given a security parameter λ, *Setup* samples public parameters pp_{enc}..
- $\mathcal{K}_{enc}(pp_{enc}) \rightarrow (pk_{enc}, sk_{enc})$ Given pp_{enc}, $\mathcal{K}_{enc}$ returns a pair of public and secret key.
- $\mathcal{E}_{enc}(pk_{enc}, m) \rightarrow \sigma$. 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 m if decryption succeeds or *null* otherwise.

We require the encryption scheme *Enc* to be both IND-CCA and IK-CCA secure[24].

3 Lelantus Construction

Lelantus is a decentralized anonymous payment system 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 in the proposed design, and also discuss its security properties.

3.1 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 $[0, v_{\max})$.
- 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 and 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.

3.2 Security

For our design we use multiple cryptographic protocols including the proprietary one-out-of-many proof scheme for double-blinded commitments and the transaction balance proof protocol. Following

to the security framework used to prove the security of the original one-out-of-many proof construction[4], we show that our proposed design for one-out-of-many proofs has perfect completeness, perfect special-honest zero-knowledge and $(m+1)$-soundness in the sense defined and proven in the Appendix A. For proving the knowledge soundness property of the transaction balance proof we use the computational witness-extended emulation model as defined in [22] and used for example in [7]. Informally witness-extended emulation implies that whenever an adversary produces an accepting proof transcript, then there exists an emulator producing an identically distributed proof with the same probability of acceptance, but also the corresponding witness. We build a witness-extend emulator for the transaction balance proof in the Appendix B.
In the Appendix C we discuss the security properties of our proposed DAP scheme with respect to the security framework introduced by Zerocash [3]. This framework captures a realistic threat model with powerful adversaries who are permitted to include malicious commitments into the commitment pool, control the choice of transaction inputs or obtain the transactions data in advance. According to [3], a decentralized anonymous payment system $\prod$=(Setup, CreateAddress, Mint, Spend, Receive, Verify) is secure, if it is **Complete** and satisfies the **Ledger-Indistinguishability**, **Transaction Non-Malleability** and **Balance** properties. We will give precise definitions of these security properties and provide formal proof sketches for all three security properties in the Appendix C.

4 Algorithm Constructions

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

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 $\mathcal{H} : \{0,1\}^* \rightarrow Z_p$.

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

1. Choose a cryptographically secure hash function $\mathcal{H} : \{0,1\}^* \rightarrow Z_p$ and 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 generators of G.
3. Compute the setup parameters of all cryptographic primitives used as design components: $pp_{\text{rp}} = \text{Setup}_{\text{bp}}$, $pp_{\text{ooon}} = \text{Setup}_{\text{ooon}}$, $pp_{\text{enc}} = \text{Setup}_{\text{enc}}$, $pp_{\text{sig}} = \text{Setup}_{\text{sig}}$.
4. Output $pp = (G, Z_p, pp_{\text{ck}}, pp_{\text{rp}}, pp_{\text{ooon}}, pp_{\text{enc}}, pp_{\text{sig}})$ and $\mathcal{H}$.

Note that one-out-of-many proofs are working with commitments generated with respect to pp_{ck}. Also g is used as the fixed generator point in the pp_{sig} and pp_{enc}.

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

Mint: This 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_{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 $\text{tx}_{\text{mint}} = (*, \sigma)$

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 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.

Spend: **Spend** algorithm generates the tx_{spend} data for the *Spend* transaction, which spends $old \geq 1$ coins $C^i_1, \ldots, C^i_{old}$ anonymously and creates $new \geq 0$ fresh output 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 is associated with a public transaction fee fee. The transaction legitimacy proof should ensure all network parties 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.
- The transaction is balanced which means $v^i_1 + \ldots + v^i_{old} = v^o_1 + \ldots + v^o_{new} + v_{out} + fee$ and $\forall j \in 1, \ldots, new : v^o_j \geq 0$.

Our **Spend** algorithm leverages the modified one-out-of-many proofs for double-blinded commitments to generate the anonymous spending and balance proofs. The one-out-of-many proof construction for double-blinded commitments is discussed in full details in the Appendix A. For each spent coin the user first reveals the coin's serial number sn, next computes a novel set of commitments by subtracting sn from all coin commitments in corresponding anonymity set **C-Pool**. Then the user generates a zero-knowledge proof of a knowledge of one double-blinded commitment

in this novel set of commitments opening to zero. Below we describe the **Spend** algorithm in details.

Spend Algorithm

Inputs: Transaction input coins $\{C_1^i, \ldots, C_{old}^i\}$ and their corresponding witnesses $\{(l_1, S_1^i, v_1^i, R_1^i), \ldots, (l_{old}, S_{old}^i, v_{old}^i, R_{old}^i)\}$ (here each l_k is the k-th input coin's index in the anonymity set **C-Pool**), the net output value v_{out} and the transaction fee fee, the output coin values $v_1^o, \ldots, v_{new}^o$ and their corresponding output public addresses $\text{addr}_{\text{pk}_1}, \ldots, \text{addr}_{\text{pk}_{new}}$.

Outputs:Spend transaction data tx_{spend}:

1. Parse $\text{addr}_{\text{pk}_i} = (P_i, Q_i, \pi_{Q_i})$ for each $i \in 1, \ldots, new$.
2. Sample $x_1, \ldots, x_{new} \leftarrow_R Z_p$ and compute the output coins: $C_1^o = Q_1^{x_1} h^{v_1^o}, \ldots, C_{new}^o = Q_{new}^{x_{new}} h^{v_{new}^o}$.
3. Compute $D_1 = \mathcal{E}_{enc}(P_1; x_1 || v_1^o)$, $\ldots$, $D_{new} = \mathcal{E}_{enc}(P_{new}; x_{new} || v_{new}^o)$. These ciphertexts enable the intended recipients to receive coin secrets directly from the ledger.
4. Compute a separate range proof for each output: $\pi_{\text{range}_1} = \pi_{\text{range}}(C_1^o; Q_1, h), \ldots, \pi_{\text{range}_{\text{new}}} = \pi_{\text{range}}(C_{\text{new}}^o; Q_{\text{new}}, h)$
5. For each input coin $C_1^i, \ldots, C_{old}^i$:
   (a) Publish the coin serial number computed as $\text{sn}_k = g^{S_k^i}$.
   (b) Compute $\textbf{C-Pool}_k = (C'_0, C'_1, \ldots, C'_{N-1})$ by homomorphically subtracting the revealed serial number sn_k from all coin commitments in the original anonymity set **C-Pool**: $C'_j = C_j \cdot Comm(\text{sn}_k^{-1}, 0, 0) \quad \forall j \in \{0, \ldots, N-1\}$.
   (c) Generate 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 non-interactive proofs $\pi_{\text{ooon}_1}, \ldots, \pi_{\text{ooon}_{old}}$ are using a common verifier challenge x which is computed through Fiat-Shamir heuristic by hashing all initial statements of *old* proofs and the output coin commitments, range proofs and ciphertexts computed at steps 2, 3, and 4 of **Spend** algorithm.
6. Compute the transaction balance proof $\pi_{\text{balance}} = \pi_{\text{dlrep}}\left(\frac{A}{B}; Q_1, \ldots, Q_{new}, h\right)$ as a non-interactive proof of representation of a specific value $\frac{A}{B}$ with respect to the generators $Q_1, \ldots, Q_{\text{new}}$ and f. The elements A and B are computed over public transaction data will be detailed below.
7. Set

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

8. For each $k \in 1, \ldots, old$ computes $\sigma_k = \mathcal{S}_{sign}(S_k^i; *)$. Note that each signature σ_k can be verified with the corresponding public key $sn_k = g^{S_k^i}$ which is part of the public transaction data tx_{spend}.
9. Set $\text{tx}_{\text{spend}} = (*, \sigma_1, \ldots, \sigma_{old})$

The values **A** and **B** used at Step 6 of the **Spend** algorithm to construct the balance proof are computed as follows: Given the public output coins $C_1^o, \ldots, C_{new}^o$, the transparent output value v_{out} and the transaction fee fee, and also the verifier challenge variable x used for constructing all *old* non-interactive π_{ooon}-proofs, the element A is computed as

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

For each transaction data tx_{spend}, the provided $(\pi_{ooon_1}, \ldots, \pi_{ooon_{old}})$ proofs can be parsed to extract special elements $(z_{v_1}, z_{R_1}, G_0^1, \ldots, G_{m-1}^1), \ldots, (z_{v_{old}}, z_{R_{old}}, G_0^{old}, \ldots, G_{m-1}^{old})$ where, according to the Fig. 1 of the Appendix A, each $z_{v_t} = v_t^i \cdot x^m - \sum_{k=0}^{m-1} \rho_k^t x^k$ and $z_{R_t} = R_t^i \cdot x^m - \sum_{k=0}^{m-1} \tau_k^t x^k$ for each

$t \in \{1, ..., old\}$ and $G_k^t = Comm(0, \rho_k^t, \gamma_k^t + \tau_k^t)$ for all $t, k \in \{1, ..., old\}, \{0, ..., m-1\}$. $\mathbf{B}$ is computed as follows:

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

All network participants can independently compute $\mathbf{A}$ and $\mathbf{B}$ based on the public ledger data. Here,

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

If the transaction balance holds, the output values and the transaction fee sum up with the spent inputs values. In this 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 $Q_1, \ldots, Q_{new}$ 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 $(Q_1, \ldots, Q_{new}, f)$ which is done at step (6) of the **Spend** algorithm.

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 $\text{tx} = \text{tx}_{\text{mint}}$ then
   - Parse the transaction payload $\text{tx}_{\text{mint}} = (*, \sigma) = (C, v, D, \text{addr}_{\text{pk}} = (Q, P, \pi_Q), \sigma))$
   - Compute $pk_{sig} = C \cdot h^{-v}$ and output $b = \mathcal{V}_{sig}(pk_{sig}; *, \sigma)$

   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 a verification public key to check the signature generated with the witness x.

2. If $\text{tx} = \text{tx}_{\text{spend}}$ then parse the transaction payload

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

   - For all $k \in \{1, \ldots, old\}$
     (a) Compute C-Pool$_k = \{\frac{C_0}{sn_k}, \ldots, \frac{C_{N-1}}{sn_k}\}$ and verify the π_{ooon_k} proof validity with respect to the list of commitments C-Pool$_k$. Output 0, if the verification fails.
     (b) Use sn_i as the verification public key to check the signature $\mathcal{V}_{sig}(\text{sn}_i, *, \sigma_k)$. Output 0 and halt, if the verification fails.

- For all $j \in \{1, \ldots, new\}$
  (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 Schnorr 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 and halt, 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 $(Q_1, \ldots, Q_{new}, h)$. Output 0, if the verification fails.
- Output 1.

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
$\text{Coins} = \{l_t, S_t, v_t, R_t\}_{t=0\ldots}$.

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** $= (C_0, C_1, \ldots, C_{N-1})$ and set $\text{Coins} = \{\}$.
3. For each tx_{mint} on the ledger 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) Decrypt $x = \mathcal{D}_{enc}(k_u, D)$ and check if $C = Q^x h^v = g^{xs_u} f^{xr_u} h^v$. If so, add the extracted coin data to the set of received coins: $\text{Coins} = \text{Coins} \cup (l, xs_u, v, xr_u)$.
4. 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, 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: $\text{Coins} = \text{Coins} \cup (l, S, v, R)$ to the list of received coins where $S = xs_u$ and $R = xr_u$.
5. 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).

5 Batch Verification and Performance

The *Spend* transaction validation process is dominated by the verification of one-out-of-many proofs, which complexity is linear of the referred anonymity set size. This is serious computational drawback for large-scale use cases when thousands of anonymous blockchain transactions should be verified with respect to large anonymity sets of size hundred thousand or beyond. In this section we describe an important batch verification mechanism for one-out-of-many proofs in the Lelantus setup which results to an efficient sub-linear verification complexity for each proof.
Assume that M different spent proofs should be verified with respect to the same anonymity set $(C_0, C_1, \ldots . C_{N-1})$. According to the **Spend** algorithm, each spent coin reveals its serial number

sn_t and a corresponding proof π^t_{oooon}. For each proof π^t_{oooon} the verifier computes the reference commitments set $\{C^t_i = \frac{C_i}{sn_t}\}_{i=0}^{N-1}$ and then checks the validity of the associated one-out-of-many proof with respect to this set $(C^t_0, C^t_1, \ldots, C^t_{N-1})$. As can be seen from the Fig 1 in the Appendix A, the verification of a single one-out-of-many proof π_{oooon} boils down to a large multi-exponentiation operation over the entire anonymity set.

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

This operation requires N group exponentiation with respect to the base points $(C^t_0, C^t_1, \ldots, C^t_{N-1})$ which differ for each proof. Taking into account the fact that each $C^t_i = \frac{C_i}{\text{sn}_t}$ and the reference serial number $\text{sn}_t = g^{S_t}$ is part of the public proof transcript tx_{spend}, we can rewrite the above equation as

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

Here we denote $f^t_i := \prod_{j=0}^{m-1} f^t_{j,i_j}; D_t := \prod_{k=0}^{m-1}(G^t_k \cdot Q^t_k)^{-x^k}; E_t := Comm(0, z^t_v, z^t_R)$ for brevity. This observation allows the verification of multiple spent proofs to be computed through a single multi-exponentiation operation with common based points and results in huge computational gain. For verifying M different spend proofs simultaneously, the verifier generates M random values $y_1, \ldots, y_M$ and computes the following product

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

which requires only $O(N)$ group exponentiation operations.

Table 2. Performance characteristics of one-out-of-many proofs with respect to various anonymity sets

$\|A\|$	Batch Size	Proof Size	Proving Time	Verif. Time	Avg. Time(ms)
16384	1	1412	1199	234	
	5			291	58
	10			461	46
	50			649	13
	100			1070	10
65536	1	2456	2378	882	
	5			975	195
	10			1104	110
	50			2006	40
	100			2993	29
	1000			22366	22

$\|A\|$	Batch Size	Proof Size	Proving Time	Verif Time	Avg. Time(ms)
262144	1	2016	14098	3542	
	5			3989	800
	10			4531	450
	50			8970	180
	100			13979	139
100000	1	2044	4416	1366	
	5			1504	300
	10			1683	168
	50			3129	62
	100			4947	45
	1000			37798	34

We have created a reference C++ implementation of Lelantus over the elliptic curve group secp256k1 using the popular library libsecp256k1. Table 2 reports the proof sizes and performance parameters

for one-out-of-many proofs with respect to different anonymity set sizes. All our experiments were performed on an Intel I7-4870HQ system with a 2.50 GHz processor. Table 2 illustrates how the average cost of a single spend verification can be as low as 30ms with respect to a large anonymity set of size 100.000 which provides significantly stronger anonymity than Monero at the comparable performance. In practice, both Zcoin and Beam are using anonymity sets of size 65536.
It is worth noting that there is still a large room for further optimizations. Particularly, one important aspect is the computation of the coefficients $f_i^t := \prod_{j=0}^{m-1} f_{j,i_j}^t$ which requires m scalar multiplications. For large batches, the additional verification cost of each proof in a batch is dominated by the effort of computing these coefficients. Using a Gray coded numbers instead of n-ary coded as discussed in the Appendix A allows to compute these coefficients significantly faster. Since Gray code guarantees only one position will change between each integer representation in the sequence, it enables each coefficient to be computed with a single multiplication and a single division on the previous coefficient instead of requiring m multiplications. As shown in [42], this optimizations results in 60% improvement in the verification performance making the average verification time for a single proof over an anonymity set of size 65525 less than 10ms.
Except of the one-out-of-many proofs, other parts of the transaction validity proofs also could take advantage of batch verification. Discrete logarithm representation proofs and range proofs all are relying on multi-exponentiation operations. These proofs can be concatenated together and checked simultaneously through a larger multi-exponentiation operation in a more efficient way.
Our protocol works over any homomorphic group and presumably a RUST implementation over the Ristretto points may result in better performance [43]. It is worth mentioning that the proving complexity of one-out-of-many proofs also can be significantly lowered by techniques described in [14].

6 Conclusion

In this paper we have presented a new decentralized anonymous payment system which provides confidentiality and high anonymity in a scalable way. Our proposal relies only on standard cryptographic assumptions, does not require any trusted setup and is relatively easy to both implement and test. We have also provided complete formal security proofs for the proposed DAP design including formal proofs for its separate cryptographic building blocks, which can be of their own interest.

Acknowledgments

The author thanks Jens Groth for his initial support and Sarang Noether for his continuous feedback and analysis and also for the evaluation of different ideas regarding to the direct anonymous transfer methods. 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):161–174,1991.

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. 243–265. Springer, Heidelberg (2015).
6. Greg Maxwell. *Confidential transactions* https://people.xiph.org/greg/confidential-values.txt, 2016.
7. 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
8. Torben P. Pedersen. *Non-interactive and information-theoretic secure verifiable secret sharing.* In CRYPTO, volume 576 of Lecture Notes in Computer Science, pages 129 -140,1991.
9. Analysis of Monero transaction inputs and outputs. https://github.com/noncesense-research-lab/monero_transaction_io
10. 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.
11. . W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S.A. Krishnan Thyagarajan, and J. Wang. *Omniring: Scaling Private Payments Without Trusted Setup.* In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19).
12. T. Yuen, Shi-feng Sun, J. K. Liu, Man Ho Au,M. F. Esgin, Qingzhao Zhang, and Dawu Gu. 2019. *RingCT 3.0 for block chain confidential transaction: shorter size and stronger security.* https://eprint.iacr.org/2019/508.
13. Pedro Moreno-Sanchez, Arthur Blue, Duc Le, Sarang Noether, Brandon Goodell, Aniket Kate. *DLSAG: Non-interactive Refund Transactions for Interoperable Payment Channels in Monero.* In Proceedings of the 2020 Financial Cryptography Conference.
14. 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
15. Vitalik Buterin. *Hard Problems in Cryptocurrency: Five Years Later.* https://vitalik.ca/general/2019/11/22/progress.html
16. 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.
17. 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
18. 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
19. B. Bünz, B. Fisch and A. Szepieniec. *Transparent snarks from dark compilers.* https://eprint.iacr.org/2019/1229, 2019.
20. 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.
21. Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates.* MIT Press, Cambridge, Massachusetts, August 2000. ISBN 0-262-02491-8.
22. Yehuda Lindell. *Parallel coin-tossing and constant-round secure two-party computation.* Journal of Cryptology, 16(3):143-184, 2003.
23. Mihir Bellare and Phillip Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols.* In ACM CCS, pages 62-73, 1993.
24. 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 566–582.
25. 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.

26. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge.* Cryptology ePrint Archive: Report 2019/953.
27. 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.
28. 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/
29. Ivan Damgard. *Efficient concurrent zero-knowledge in the auxiliary string model.* In EUROCRYPT, volume 1807 of Lecture Notes in Computer Science, pages 418–430, 2000.
30. Mihir Bellare and Phillip Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols.* In ACM CCS, pages 62-73, 1993.
31. 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.
32. 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 609–642, 2017.
33. Zcoin: Zcoin's upcoming privacy protocols: https://zcoin.io/lelantus-zcoin/
34. Monero: A Reasonably Private Digital Currency. https://web.getmonero.org/
35. Grin: The private and lightweight mimblewimble blockchain. https://grin-tech.org/
36. MimbleWimble-based Privacy Coin. https://beam.mw/
37. Lelantus-MW: https://docs.beam.mw/Lelantus-MW.pdf
38. V. Gelfer. *Mimblewimble Confidential Lelantus Assets* https://github.com/BeamMW/beam/wiki/MW-CLA
39. A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille: *Confidential Assets*, https://blockstream.com/bitcoin17-final41.pdf
40. I. Miers: *Satoshi Has No Clothes: Failures in On-Chain Privacy.*
https://slideslive.com/38911785/satoshi-has-no-clothes-failures-in-onchain-privacy.
41. https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/
42. Computation of set filter coefficients with Gray coded numbers. https://github.com/cargodog/one-of-many-proofs/
43. Fast, safe, pure-rust elliptic curve cryptography.https://github.com/dalek-cryptography.
44. Ian Miers. *Blockchain Privacy: Equal Parts Theory and Practice (especially Flashlight Attack part)* https://www.zfnd.org/blog/blockchain-privacy/flashlight

Appendix A One-out-of-many proofs for double-blinded commitments

In this section we provide the detailed description of one-out-of-many proofs for double-blinded commitments and also give formal security proofs for the proposed design. The protocol is defined by three algorithms $(\text{Setup}_{\text{ooon}}, \text{Prove}_{\text{ooon}}, \text{Ver}_{\text{ooon}})$ and is a zero-knowledge proof of knowledge for the following relation:

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

For proving the security of our scheme we show that the proposed design is perfectly complete, special honest verifier zero-knowledge, and has n-special soundness as is discussed in [4]. Informally, these security properties are defined as follows:

- **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.

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

Fig. 1. One-out-of-many proof for double-blinded commitments

Our construction builds upon the one-out-of-many proof construction introduced 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 full protocol is described in details in Figure 1.
Technically, 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 the proof transcript reveals two different values z_v and z_R for the two random values used in the commitment. This construction could be similarly extended for Generalized Pedersen commitments of any size. The next major difference is instead of outputting the product $\prod_{i=0}^{N-1} C_i^{p_{i,k}} \cdot Comm(0, \rho_k, \tau_k)$ as a single element G_k, we split this product and output a pair of two values $G_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}}$ and $Q_k = Comm(0, \rho_k, \tau_k)$ indeed. Note that the outputted elements G_k and Q_k in turn are blinded via extra random factors f^{γ_k} and $f^{-\gamma_k}$ which will be neutralized in the product $G_k \cdot Q_k$ during the proof verification process. Revealing G_k and Q_k separately instead of outputting their product increases the proof size, but is vital for generating the transaction balance proof. In other use cases of one-out-of-many proofs for generalized Pedersen Commitment proofs the proof transcript could safely output the product of G_k and Q_k as a single element.
Next, we formally proof 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.*

Perfect Completeness:. By the perfect completeness of the sigma protocol of Fig. 1 we have that the verifier always accepts the valid proofs. To see that the protocol is perfectly 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 correctness of the last verification equation can be proven as follow:

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

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

$$\prod_{i=0}^{N} C_i^{p_i(x)} \prod_{i=0}^{N-1} C_i^{-\sum_{k=0}^{m-1} p_{i,k} \cdot x^k} \cdot Comm0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k \Big) = \prod_{i=0}^{N} C_i^{\sigma_{l,i} x^m} \cdot Comm\Big(0, -\sum_{k=0}^{m-1} \rho_k x^k, -\sum_{k=0}^{m-1} \tau_k x^k\Big) =$$

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

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

Perfect Special-Honest Verifier Zero-Knowledge: 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:

$$B, C, G_1, \dots G_{m-1}; Q_0, Q_1, \dots Q_{m-1} \leftarrow_R G$$

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

. Next it computes

$$f_{j,0} = x - \sum_{i=1}^{n-1} f_{j,i} \quad \forall j \in \{0, \dots, m-1\}$$

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

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

Soundness: Assuming that $N = n^m$, we will show our sigma protocol is $(m+1)$-special sound. Suppose an adversary can produce $(m+1)$ different accepting responses

$$\Big((f_{j,i}^{(0)}, z_v^{(0)}, z_R^{(0)}), \dots, (f_{j,i}^{(m)}, z_v^{(m)}, z_R^{(m)})\Big)$$

with respect to the same initial message and $m+1$ different challenges $x_{(0)}, \dots, x_{(m)}$ where $m > 1$. As described in the original paper [5], 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 $\sum_{i=0}^{n-1} \sigma_{l_j,i} = 1$ with just two different responses. This opening will define the index $l = \sum_{j=0}^{n-1} l_j n^j$, where l_j is the index of the only 1 in the sequence $\sigma_{l_j,0}, \dots, \sigma_{l_j,n-1}$. Following the proof, all answers satisfy $f_{j,i}^{(e)} = \sigma_{l_j,i} x_{(e)} + a_{j,i}$ for $0 \le e \le m$ with overwhelming probability due to the binding property of the commitment scheme.
Next, given 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

$$c_l^{x_{(e)}^m} \cdot \prod_{k=0}^{m-1} \big(G_k \cdot Q_k\big)^{-x_{(e)}^k} = Comm(0, z_v^e, z_R^e)$$

The equation holds for all $x_e \in x_{(0)}, \dots, x_{(m)}$ where the values G_k are derived from the initial statement. Consider the Vandermonde matrix with the e-th row specified as $\Big(1, x_{(e)}, \dots, (x_{(e)})^m\Big)$. 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 $(0, \ldots, 0, 1)$. This will enable to deduce $c_l = \prod_{e=0}^{m} \Big(c_l^{x^m_{(e)}} \cdot \prod_{k=1}^{m-1} (G_k \cdot Q_k)^{x^k_{(e)}} \Big)^{\theta_e} = Comm\Big(0, \sum_{e=0}^{m} \theta_e z_v^e, \sum_{e=0}^{m} \theta_e z_R^e\Big)$, which provides an opening of double-blinded commitment c_l to the 0 using the blinding factors $v = \sum_{e=0}^{m} \theta_e z_v^e$ and $R = \sum_{e=0}^{m} \theta_e z_R^e$.

Appendix B Security Properties of the Transaction Balance Proof

The transaction balance proof shows that for each transaction the amount of all output coins sum up with the input coin values. We want this proof to be **Complete** in a sense that if the prover knows the witness for the balance statement and follows to the proof generation steps correctly, then he always succeeds to convince the verifier. We also require the transaction balance proof to be **Special honest verifier zero-knowledge**: so it does not reveal anything about the prover's witness including the input and output coin values or other secrets which may reveal the input coin origins. It is relatively straightforward to show that the proposed transaction balance proof is complete and special honest verifier zero-knowledge and we leave the formal proofs for the the full paper. In this section we provide a sketch that the the proposed proof is also **Special sound**, which implies that unless the prover does not know the transaction balance proof witness, he should not be able to convince the verifier. We use the witness-extended emulation to define the soundness property of the transaction balance proof as is defined in [22] and used for example in [7].
Informally witness-extended emulation implies that whenever an adversary produces an proof which satisfies the verifier with some probability, then there exists an emulator producing an identically distributed proof with the same probability of acceptance, but also the corresponding witness. The emulator is permitted to rewind the interaction between the prover and verifier to any move, and resume with the same internal state for the prover, but with fresh randomness for the verifier. We leverage the soundness properties of the one-out-of-many proofs, Schnorr's proof of representation and bulletproofs to show how the emulator can extract the balance proof witness and generate an associated valid proof.

Note that according **Spend** algorithm description and the equation (3), the balance proof witness is comprised of the following elements:

$$\{v_t^i, R_t^i, \{\rho_k^t, \tau_k^t, \gamma_k^t\}_{k=0}^{m-1}, (x_1, \ldots, x_{new}), (v_1^o, \ldots, v_{new}^o)\}$$

where v_t^i and R_t^i are the t-th input coin amount and blinding factor accordingly for all $t \in 1, ..., old$, $\{\rho_k^t, \tau_k^t, \gamma_k^t\}_{k=0}^{m-1}$ are the random values used to generate the one-out-of-many proof for the t-th spent coin, v_l^o and x_l are the l-th output coin value and blinding factor. For each $t \in 1, ..., old$ the spend proof outputs the values

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

and

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

Next we briefly describe how to build an emulator which extracts all witness elements. We leave the full argument description to the full paper.

1. **Step 1: Extracting the values** $v_t^i, R_t^i, \{\rho_k^t, \tau_k^t\}_{k=0}^{m-1}$: We use the $(m+1)$ special-soundness property of one-out-of-many proofs to extract all secret values used to generate the elements z_{v_t} and z_{R_t}. For the given transaction, when the prover submits the initial statements of all one-out-of-many proofs for each input coin, the emulator rewinds the prover with $(m+1)$ different randomnesses $x_{(0)}, x_{(1)}, \ldots, x_{(m)}$ and gets $(m+1)$ different one-out-of-many proof responses for each initial statement. At the end of this loop, for any spent coin the emulator can construct the following system of linear equations:

$$\begin{aligned} z_{v_t}^0 &= v_t^i \cdot x_{(0)}^m - \sum_{k=0}^{m-1} \rho_k^t x_{(0)}^k \\ z_{v_t}^1 &= v_t^i \cdot x_{(1)}^m - \sum_{k=0}^{m-1} \rho_k^t x_{(1)}^k \\ &\vdots \\ z_{v_t}^m &= v_t^i \cdot x_{(m)}^m - \sum_{k=0}^{m-1} \rho_k^t x_{(m)}^k \end{aligned}$$

   As all $x_{(i)}$ are unique, the system of linear equations will have an invertible matrix of coefficients. Therefor the emulator can solve the system and find all $(m+1)$ unknowns $v_t^i, \rho_0^t, \rho_1^t, \ldots, \rho_{m-1}^t$. Similarly the emulator can extract the unknowns $R_t^i, \tau_0^t, \tau_1^t, \ldots, \tau_{m-1}^t$ from the values $z_{R_t}^0, z_{R_t}^1, \ldots, z_{R_t}^m$.

2. **Step 2: Extracting the values** $\gamma_0^t, \gamma_1^t, \ldots \gamma_{m-1}^t$. Using the 3-special-soundness property of the Shnorr's proofs of representation, the emulator can extract the witness of the Schnorr's proofs provided for the balance proof equation by rewinding the prover with 3 fresh random values:

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

   The witness of the Schnorr's proof of representation is comprised of the values $(x_1 \cdot x^m), \ldots, (x_{new} \cdot x^m)$ and $T = \sum_{t=1}^{old}\left(R_t^i \cdot x^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x^k\right)$ and as the challenge variable x is fixed, the extraction of $x_1 \cdot x^m$, $x_2 \cdot x^m$ and $x_{new} \cdot x^m$ will immediately reveal the values $x_1, x_2, \ldots, x_{new}$.
   In order to extract all $l = t \cdot m$ values γ_k^t, the emulator restarts rewinding the prover $l = t \cdot m$ times with different challenge variables $x_{(1)}, x_{(2)}, \ldots, x_{(t \cdot m)}$ and generates the Shnorr's proof of representation for all different computations $\frac{A}{B}$. Next, also running the soundness proof of Schnorr's proof of representation the emulator will be able to eventually extract $t \cdot m$ different values

$$T_i = \sum_{t=1}^{old}\left(R_t^I \cdot x_{(i)}^m + \sum_{k=0}^{m-1} \gamma_k^t \cdot x_{(i)^k}\right)$$

   for each $i \in \{1, \ldots, tm\}$ and obtain a system of linear equations

$$T_1 - \sum_{t=1}^{old}\left(R_t \cdot x_{(1)}^m\right) = \sum_{t=1}^{old}\sum_{k=0}^{m-1} \gamma_k^t \cdot x_{(i)}^k$$

$$T_2 - \sum_{t=1}^{old}\Big(R_t \cdot x_{(2)}^m\Big) = \sum_{t=1}^{old}\sum_{k=0}^{m-1}\gamma_k^t \cdot x_{(2)}^k$$

$$\vdots$$

$$T_l - \sum_{t=1}^{old}\Big(R_t \cdot x_{(l)}^m\Big) = \sum_{t=1}^{old}\sum_{k=0}^{m-1}\gamma_k^t \cdot x_{(l)}^k$$

where we assume the value R_T is extracted during the previous round of emulation. As all $x_{(i)}$ are different, the coefficient matrix will be invertible with an overwhelming probability and the emulator can solve the system to extract the unknowns $\{\gamma_k^t\}$ for all $t \in 1, \ldots,$ old and $k \in 0, \ldots, m-1$.

At the next step we can extend the emulation process by using the witness-extended emulation for bulletproofs to also extract the output coin values. Importantly, as the solution of the linear equation system of size m or $t \cdot m$ can be found in a polynomial time, and also the emulator for the bulletproofs will work in the expected polynomial time, than our emulator will work in the expected polynomial time.
Soundness of the balance proof implies that for any accepting proof all input values including the elements G_k used by the prover are formed correctly and the prover possesses the corresponding witness. This soundness property of the transaction balance proof is used to prove the **Balance** property of the DAP scheme as defined in the Appendix C.

Appendix C DAP scheme security

In this section we formally discuss the security of our decentralized anonymous payment scheme within the DAP scheme security framework introduced by Zerocash[3] and recall all corresponding definitions and notations below for the sake of paper integrity.

Definition C.1 A DAP scheme Π=(Setup, CreateAddress, Mint, Spend, Receive, Verify) is secure, if it is **Complete** and satisfies the **Ledger Indistinguishability**, **Transaction Non-Malleability** and **Balance** properties.

Completeness implies that any unspent coin on the ledger can be spent by the respective owner which 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. It is easy to check that for our scheme this property immediately follows from the completeness property of the one-out-of-many proof construction and the fact that all coin's serial numbers are unique.

The next security properties are 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 $v_{\text{unspent}} + v_{\text{basecoin}} + v_{\mathcal{A}\rightarrow\text{ADDR}} > v_{\text{mint}} + v_{\text{ADDR}\rightarrow\mathcal{A}}$ which means the total value he can spend or has spent already is greater than the value he has minted or received, where

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

Definition C.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 transactions reveal the public values of new minted coins, the number of *Spend* transaction inputs and outputs and also the recipient addresses. Note that the Ledger-indistinguishability property includes the anonymity property defined for Zerocoin in the [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 adversary's 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 C.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.

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

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 property 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 same serial 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 C.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.

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

C.1 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 above, whenever it outputs a valid transaction tx^* such that for some $tx \in \mathcal{T}$ the following holds:

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

Let's 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.

C.2 Proof of Balance

In this section we show that our DAP scheme satisfies to the **BAL** security property according to the definition C.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

$$\begin{aligned}\text{tx}_{\text{spend}} = \Big\{ & \{(sn_1, \pi_{\text{ooon}_1}) \ldots, (sn_{\text{old}}, \pi_{\text{ooon}_{old}})\}, \\ & \pi_{\text{balance}}, v_{out}, fee, (\text{addr}_{\text{pk}_1}, C^o_1, D_1, \pi_{\text{range}_1}), \\ & \ldots, (\text{addr}_{\text{pk}_{\text{new}}}, C^o_{new}, D_{new}, \pi_{\text{range}_{\text{new}}}), \sigma_1, \ldots, \sigma_{old} \Big\}\end{aligned}$$

on the ledger L, the challenger $\mathcal{C}$ computes the transaction witness $a = \Big(l_1, \ldots, l_{old}; (S^i_1, v^i_1, R^i_1), \ldots, (S^i_{old}, v^i_{old}, R^i_{old}); (S^o_1, v^o_1, R^o_1), \ldots, (S^o_{new}, v^o_{new}, R^o_{new})\Big)$. 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^i_k is the output of some mint transaction tx_{mint} on L, then the public value v in tx_{mint} is equal to v^i_k.
   - If C^i_k 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^i_k.

   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 $v^i_1 + \ldots + v^i_{old} \neq v^o_1 + \ldots + v^o_{new} + v_{out} + fee$. As the transaction is considered valid and an accepting balance proof is provided, this violates the soundness property of the transaction balance proof proved in the Appendix B and leads to a 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.3 Proof of Ledger Indistinguishability

We will show that our DAP scheme satisfies to the **L-IND** security property according to the Definition C.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 $\mathcal{O}^{DAP}_0$ and $\mathcal{O}^{DAP}_1$ 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

$$(\textbf{Spend}, C^i_1, \text{addr}^i_{\text{pk},1}, \ldots, C^i_{\text{old}}, \text{addr}^i_{\text{pk,old}}, v^o_1, \ldots, v^o_{new}, \text{addr}^o_{\text{pk},1}, \text{addr}^o_{\text{pk,new}}, v_{\text{out}})$$

. The challenger generate a simulated tx_{spend} as follows:

1. For $j \in 1, \ldots, \text{old}$
   - Samples a random $\text{sn}_j = g^{s_j}$.
   - Simulates the proof $\pi^{Sim}_{\text{ooon}_j} \leftarrow \text{Sim}(\text{OOON}, \text{sn}_j)$.
   - If $\text{addr}^o_{\text{pk}_j}$ is not generated by one of the previous **CreateAddress** queries and is obviously controlled by the adversary, the simulator calculates the values C^o_j and D^o_j as in the **Spend** algorithm.
   - If $\text{addr}^o_{\text{pk}_j}$ belongs to a honest party and is generated by a previous **CreateAddress**, $\mathcal{C}$
     * Samples a random coin commitment $C^o_j = g^{r_1} h^{r_2} f^{r_3}$
     * Generates new public key encryption key pair $(k', P') \leftarrow \mathcal{K}_{\text{enc}}(pp_{enc})$ and computes $D^o_j = \mathcal{E}_{enc}(P', r)$ 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: $\pi^{Sim}_{\text{balance}} \leftarrow \text{Sim}(\text{DLREP}, \{C^o_1, \ldots, C^0_{\text{new}}, \pi^{Sim}_{\text{ooon}_1}, \ldots, \pi^{Sim}_{\text{ooon}_{\text{old}}}\})$

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 $|\text{Adv}^{\mathcal{D}_{sim}} - \text{Adv}^{\mathcal{D}_2}| = 0$.

This finalizes the proof by showing that $|\text{Adv}^{\mathcal{D}_{sim}} - \text{Adv}^{\mathcal{D}_{real}}| \leq \text{negl}(1^\lambda)$.