Lelantus-CLA

Pyrros Chaidos[1] and Vladislav Gelfer[2]

[1]National & Kapodistrian University of Athens, Greece
pchaidos@di.uoa.gr

[2] BEAM Foundation
valdo@beam-mw.com

Abstract

This article presents Lelantus-CLA, an adaptation of Lelantus [Jiv19] for use with the Mimblewimble protocol and confidential assets. Whereas Mimblewimble achieves a limited amount of privacy by merging transactions that occur in the same block, Lelantus uses a logarithmic-size proof of membership to effectively enable merging across different blocks. At a high level, this allows value to be added to a common pool and then spent privately, but only once. We explain how to adapt this construction to Mimblewimble, while at the same time simplifying the protocol where possible.

Confidential assets is a mechanism that allows multiple currencies to co-exist in the same ledger and (optionally) enables transactions to be conducted without disclosing the currency.

Finally, we also describe how we can use Bulletproof "coloring" to enable offline payments, thus addressing one of the original shortcomings of Mimblewimble.

1 Introduction

In this article we destibe Lelantus-CLA, an adaptation of Lelantus [Jiv19] for use with the Mimblewimble protocol and confidential assets.

Mimblewimble achieves a limited amount of privacy by merging transactions that occur in the same block, effectively mixing together all transactions that were submitted to the same block. This has significant privacy improvements when compared with e.g. Bitcoin in which transactions are individually traceable and mixing (or tumbling) requires the use of a separate service. However, the degree of privacy that is offered is limited, and is not under the control of the user. While large blocks offer better mixing, the user has little way of ensuring that.

Lelantus uses a logarithmic-size proof of membership to effectively enable merging across different blocks. At a high level, this may be thought of as a large common pool of value to which individuals can make deposits and withdrawals in private. The key feature is of course that any particular deposit can only be withdrawn once. We explain how to adapt this construction to Mimblewimble, while at the same time simplifying the protocol where possible. We term this construction Lelantus-MW.

In Lelantus-CLA, we also add support for confidential assets [PBF+18, ZYD+19] in an efficient manner by allowing users to hide the asset types used inside transactions whilst enforcing that all types are valid.

At the same time, we also describe how we can use Bulletproof "coloring" to enable offline payments, thus addressing one of the original shortcomings of Mimblewimble. In our context coloring refers to hiding values inside the randomness used in the construction of zero knowledge proofs. This enables those with knowledge of the appropriate seed to detect and extract these values. With this functionality in place, senders can send funds without needing to interact with recipients, as the recipients are able to extract the necessary information on their own.

1.1 Outline

In Section 2 we give the necessary cryptographic definitions for the core concepts used in this work. Next, in Section 3 we describe the constructions which use when designing and implementing Lelantus-MW and Lelantus-CLA. Our choices are well-studied constructions, requiring limited assumptions.

Before delving into Mimblewimble-MW (Section 5), Section 4 revisits Mimblewimble to establish our baseline and better highlight the improvements in our construction.

In Section 6 we describe support for multiple assets, and introduce Mimblewimble-CLA which leverages confidential assets [PBF+18, ZYD+19] so that the asset types used in a transaction may be hidden.

Finally, in 8, we describe the technical details required to conduct transactions in a meaningful way (i.e ensure privacy and integrity are upheld). This includes the concept of bulletproof coloring, as well as the notions of tickets and addresses.

2 Preliminaries

2.1 The Discrete Log Setting

Let GGen be an algorithm that on input 1^λ returns $gk = (\mathbb{G}, p, g)$ such that $\mathbb{G}$ is the description of a finite cyclic group of prime order p, where $|p| = \lambda$, and g is a generator of $\mathbb{G}$.

Definition 1 (Discrete Logarithm Assumption)**.** *The discrete logarithm assumption holds relative to* GGen *if for all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[(\mathbb{G}, p, g) \leftarrow \mathrm{GGen}(1^\lambda); h \leftarrow \mathbb{G}; a \leftarrow \mathcal{A}(\mathbb{G}, p, g, h) : g^a = h\right] \approx 0$$

2.2 Commitment Schemes

Commitment schemes allows one to commit to a value v without revealing it. We use $\mathrm{Com}_{ck}(v; r)$ to refer to a commitment to a value v using randomness r under a previously determined key ck. A commitment can later be opened to reveal the value contained. We require that commitments are hiding i.e it must be difficult to correctly guess the value contained in a commitment given the key and the commitment itself. We also require that commitments are binding i.e. it must be difficult for a malicious opener to create a commitment and open it to two different values.

Definition 2 (Perfectly hiding)**.** *We say a non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is perfectly hiding if a commitment does not reveal the committed value. For all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{CGen}(gk); (m_0, m_1) \leftarrow \mathcal{A}(ck); \\ b \leftarrow \{0,1\}; c \leftarrow \mathrm{Com}(m_b) \end{array} : \mathcal{A}(c) = b\right] = \frac{1}{2}$$

where $\mathcal{A}$ *outputs* $m_0, m_1 \in \mathbb{Z}_p$.

Definition 3 (Binding)**.** *A non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is computationally binding if a commitment can only be opened to one value. For all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{CGen}(gk); \\ (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck) \end{array} : \begin{array}{c} \mathrm{Com}(m_0; r_0) = \mathrm{Com}(m_1; r_1) \\ \text{and } m_0 \neq m_1 \end{array}\right] \approx 0$$

where $\mathcal{A}$ *outputs* $m_0, m_1, r_0, r_1 \in \mathbb{Z}_p$.

2.2.1 Homomorphic Commitments

An important requirement for our application is that they are additively homomorphic, that is the sum of two commitments is a commitment to the sum of the values contained within. We require that the property holds for values and randomisers alike.

$$\mathrm{Com}(v; r) + \mathrm{Com}(v'; r') = \mathrm{Com}(v + v'; r + r')$$

2.3 Zero Knowledge Arguments of Knowledge

Let R be a polynomial time decidable binary relation, i.e., a relation that defines a language in NP. We call w a witness for a statement u if $(u, w) \in R$. We define the language

$$L_R = \{x \mid \exists w : (u, w) \in R\}$$

as the set of statements x that have a witness w in the relation R.

A zero knowledge protocol for an NP-relation R enables a prover to demonstrate to a verifier that a statement u satisfies $u \in L_R$, i.e. that there exists a witness w such that $(u, w) \in R$ without disclosing anything else, in particular not disclosing the value of w that the prover has in mind.

More formally, a protocol for a relation R w.r.t. a setup $gk \leftarrow \mathrm{GGen}(1^\lambda)$ and a common reference string crs is a tuple $(\mathrm{CRSGen}, \mathcal{P}, \mathcal{V})$. If a common reference string, crs is used, we assume that the generator, $\mathrm{CRSGen}(gk)$ generates it before the execution of the protocol. As we are interested in statements about commitments, we additionally consider the commitment key ck, part of the crs. Both gk and crs are provided as implicit inputs to all algorithms. We consider that the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, are both probabilistic polynomial time interactive algorithms. The transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$.

We require that a protocol woks correctly when used by honest participants (*completeness*), that it prevents provers from convincing verifiers about a statementa unless they know a witness (*knowledge soundness*), and that it does not leak information about w (*zero-knowledge*).

Definition 4 (Public coin)**.** *A protocol* $(\mathcal{P}, \mathcal{V})$ *is called* public coin *if the verifier chooses his messages uniformly at random and independently of the messages sent by the prover, i.e., the challenges correspond to the verifier's randomness* ρ*.*

Definition 5 (Perfect completeness)**.** $(\mathcal{P}, \mathcal{V})$ *has perfect completeness if for all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[(u, w) \leftarrow \mathcal{A}(1^\lambda) : (u, w) \notin R \text{ or } \langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1\right] = 1$$

Definition 6 (n-Special Soundness)**.** *We say that* $(\mathcal{P}, \mathcal{V})$ *is* n*-special sound [GK15] if there exists an efficient extractor* χ *such that for any statement* u*, given* n *accepting transcripts* $\{(a, x_i, z_i)\}_{i=1}^n$ *where the challenges* x_i *are distinct,* $\chi(u, a, x_i, z_i)$ *outputs* w *s.t.* $(u, w) \in R$*.*

For more complex protocols, involving a large number of rounds n-Special Soundness can be hard to obtain so we make use of the related notion of witness-extended emulation introduced by Lindell [Lin03] as it is used in [GI08, BBB$^+$18]. We refer the reader to these texts for details.

A protocol is zero-knowledge if it does not leak information about the witness beyond what can be inferred from the truth of the statement. Special honest verifier zero-knowledge further specifies that if the verifier's challenges are known in advance, then it is possible to simulate the entire argument without knowing the witness.

Definition 7 (Special Honest Verifier Zero-Knowledge (SHVZK))**.** *A public coin argument* $(\mathcal{P}, \mathcal{V})$ *is called a* special honest verifier zero knowledge (SHVZK) *argument for* R *if there exists a probabilistic polynomial time simulator* $\mathcal{S}$ *such that for all interactive non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[(u, w, \rho) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle \mathcal{P}(u, w), \mathcal{V}(u; \rho) \rangle : (u, w) \in R \text{ and } \mathcal{A}(tr) = 1\right]$$

$$\approx \Pr\left[(u, w, \rho) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \mathcal{S}(u, \rho) : (u, w) \in R \text{ and } \mathcal{A}(tr) = 1\right]$$

where ρ *is the public coin randomness used by the verifier.*

If this holds also for unbounded adversaries $\mathcal{A}$, we say $(\mathcal{P}, \mathcal{V})$ is statistically special honest verifier zero-knowledge. If the two distributions are equal, then we say $(\mathcal{P}, \mathcal{V})$ is perfect special honest verifier zero-knowledge.

3 Constructions

3.1 Generalised Pedersen Commitments

Pedersen commitments [Ped91] are a widely used homomorphic commitment scheme. Under the discrete logarithm assumption, Pedersen Commitments are perfectly hiding and computationally binding. In this paper we will also use a variant of the standard scheme that makes use of two randomisers. We will use r, s to refer to the randomisers and $\text{Com}(v, r, s)$ to refer to the commitments. The key generation and commitment functions of the pedersen scheme are as follows:

$$\begin{aligned}&\text{CGen}(gk) : h, k \leftarrow \mathbb{G}; \text{Return } ck := (g, h, k)\\&\text{Com}_{ck}(v, r, s) : \text{Return } c = g^r h^v k^s\\&\text{Com}_{ck}(v, r) : \text{Return } c = g^r h^v\end{aligned}$$

We point out that $\text{Com}_{ck}(v, r)$ is functionally identical to $\text{Com}_{ck}(v, r, 0)$ due to the structure of the commitment scheme. The (generalised) Pedersen Commitments are a core component of our construction, so ck is used in defining the protocol relations. It is simple to extend Pedersen commitments to longer bases, by appending elements to ck. For example let $\boldsymbol{b} := \{b_1, \dots b_n\} \in \mathbb{G}^n$, the we can use:

$$\text{Com}_{ck,\boldsymbol{b}}(v, r, s, v_1, \dots, v_n) : \text{Return } c = g^r h^v k^s {b_1}^{v_1} \dots {b_n}^{v_n}$$

In section 6, we will be making use of commitments a single "asset tags" b_i contained in $\boldsymbol{b}$. Effectively, we will use b_i in the place of h contained in ck. To simplify notation, we will expand the vector $\boldsymbol{b}$ by considering that $b_0 = h$. More concretely we write that:

$$\text{Com}_{ck,\boldsymbol{b},i}(v, r, s) : \text{Return } c = g^r {b_i}^v k^s$$

We note that $\text{Com}_{ck,b_i}(v, r, s) = \text{Com}_{ck,\boldsymbol{b}}(0, r, s, v_1, \dots, v_n)$ when $v_i = v$ and $v_j = 0$ for $j \neq i$, and $\text{Com}_{ck,h}(v, r, s) = \text{Com}_{ck}(v, r, s)$ by identity.

3.1.1 Representing Supply

Before we continue, we will add a simple but useful shorthand to represent value "in the open" without any serial number or randomness. The Sup terms are deterministic, so we can use them in balance equations to represent value entering the system e.g. via minting or an initial allocation.

$$\begin{aligned}&\mathsf{Sup}(v) : \text{Return } \text{Com}_{ck}(v, 0)\\&\mathsf{Sup}_i(v) : \text{Return } \text{Com}_{ck,\boldsymbol{b},i}(v, 0, 0)\\&\mathsf{Sup}(v, v_1, \dots, v_n) : \text{Return } \text{Com}_{ck,\boldsymbol{b}}(v, 0, 0, v_1, \dots, v_n)\end{aligned}$$

3.2 Range Proofs

The additive homomorphic property of Pedersen Commitments is a first step towards ensuring that sums of values "add up". However, since the values themselves remain hidden, we need to enforce that users are unable to enter negative values; otherwise one might pay \$1 for a \$5 coffee and receive -\$4 as change. Such a transaction would in principle balance, but in practice the intended recipient of a negative note would simply ignore it[1]. We thus need to disallow commitments to negative values. We do this by requiring that committed values lie in a certain range and use a zero knowledge protocol to enforce that without having to reveal the values.

Bulletproofs Bulletproofs are a general purpose zero knowledge proving system that can be tailored to perform efficient range proofs. Given a bound $v_{\max}$, the original construction from [BCC+15] can be used directly to prove knowledge of a witness (r, v) for the relation:

$$R_\rho = \{(C, b), (r, v) : C = g^r b^v \wedge (0 \leq v < v_{\max})\}$$

Bulletproofs require a CRS structured as a Pedersen commitment key:

$$\begin{aligned} \mathrm{CRSGen}_\rho(gk) : & g_1, \ldots, g_n, h_1, \ldots, h_n \leftarrow \mathbb{G}, \text{ for } n = \log_2(v_{max}) \\ & \text{Return } crs := (\boldsymbol{g}, \boldsymbol{h}) \end{aligned}$$

In section 5, we slightly modify the protocol to add a second commitment to the statement:

$$R'_\rho = \{(C, b, \hat{C}), (r, v) : C = g^r b^v \wedge (0 \leq v < v_{max})\}$$

This is immaterial for the interactive version, but when applying the Fiat-Shamir transformation, this ensures that a proof can only be verified in relation to $\hat{C}$ as well as C.

Schnor Signatures In our applications we will often need to prove that the value of certain commitments is exactly zero. While it would be possible to do that with a range proof with a very restricted range, it is more efficient to do so directly. We use the term “signatures” as opposed to proofs of zero value due to convention. We use Schnor signatures to prove knowledge of a witness r for the relation:

$$R_\sigma = \{C, r : C = g^r\}$$

We will later use a generalised version for the relation, which allows for an arbitrary base b:

$$R_{\sigma b} = \{(C, b), (r, w) : C = g^r b^w\}$$

This allows us to define relations R_{nv} and R_{ns} that enable us to prove knowledge of a commitment's opening such that it holds zero value or a zero serial number. We will also make use of it directly in section 6 to handle arbitrary asset tags.

$$R_{nv} = \{(C, k), (r, s) : C = g^r k^s\}$$

We can also use the same protocol with a different base to show that a commitment has a zero in the serial component.

$$R_{ns} = \{(C, h), (r, v) : C = g^r h^v\}$$

[1]Strictly speaking, a small negative value would be equivalent to a large positive value since arithmetic happens modulo q.

3.3 Logarithmic Membership Proofs

The protocol of Groth and Kohlweiss [GK15], later optimized in [BCC$^+$15] allows one to prove they know the opening of a single commitment in a set of many without specifying which one. The size of the proofs is logarithmic in the size of the set which enables us to be efficient. Assuming a maximum set size of $m_{\max}$ the protocol requires a common reference string as follows:

The protocol requires[2] a CRS with structure similar to a Pedersen commitment key:

$$\text{CRSGen}_m(gk) : g_1, \ldots, g_n \leftarrow \mathbb{G}, \text{ for } n = \log_2(m_{max}) \\ \text{Return } csr' := \boldsymbol{g}$$

The protocol enables us to reveal a serial number s and prove that a commitment C contains that serial number, and is contained in set $\mathbf{C}$, as in the following relation:

$$R_m = \{(\mathbf{C}, s), r : C = g^r k^s \wedge (C \in \mathbf{C})\}$$

3.4 The Fiat-Shamir Heuristic

The protocols we describe above operate in multiple rounds which is not practical for our application. The Fiat-Shamir transformation [FS87] produces a non-interactive version of a Σ-protocol by substituting the verifier's challenge with the output of a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ on the protocol's statement and current transcript. The transformation can be proven secure in the random oracle model [BR93]. The transformation crucially relies in the public coin and HVZK property of the protocols.

4 Mimblewimble

In this section we briefly describe the original version of Mimblewimble as adapted from [Jed16]. For simplicity, we omit the discussion of fees, minting and block structure.

4.1 Transactions

Intuitively, a transaction in Mimblewimble consists of gathering an amount of value stored in existing notes and re-distributing it in a set of new notes, so that the sums of value in the old (input) notes and the new (output) notes are equal. This is augmented by proofs certifying all the new notes are within the allowed range of values, and by a number of special zero-value notes (kernels) that exist to maintain privacy.

Formally, a transaction T consists of three lists: $\mathsf{T.I}, \mathsf{T.O}, \mathsf{T.K}$. The list $\mathsf{T.I}$ consists of commitments $C_{I1}, C_{I2}, \ldots$. The list $\mathsf{T.O}$ consists of tuples $(C_{O1}, \pi_{O1}), (C_{O2}, \pi_{O2})$, such that C_{Oj} is a commitment and π_{Oj} is a range proof on statement C_{Oj}, h. The

[2]In practice, as the structure is identical to the bulletproof crs we are able to reuse after ensuring the length is set to the maximum of $\log_2(m_{max}), \log_2(v_{max})$.

list of kernels $\mathsf{T.K}$ consists of tuples $(C_{K1}, \sigma_{K1}), (C_{K2}, \sigma_{K2})$, such that C_{Kl} is a commitment and σ_{Kl} is a signature demonstrating C_{Kl} is zero-valued[3].

Definition 8. *A transaction* $\mathsf{T} = (\mathsf{T.I}, \mathsf{T.O}, \mathsf{T.K})$ *is valid if:*

- *The list of commitments* $\mathsf{T.I}$ *is a subset of the current* UTXO *set.*
- *For each commitment* C_{Oj}*, the corresponding proof* π_{Oj} *is valid for statement* C_{Oj}, h*.*
- *For each commitment* C_{Kl} *the corresponding signature* σ_{Kl} *is valid.*
- *The the sum of inputs minus the sum of outputs is equal to the sum of zero-valued commitments:* $\sum_i C_{Ii} - \sum_j C_{Oj} = \sum_l C_{Kl}$*.*

4.2 Merging Transactions

The way transactions are structured is highly conducive to merging: given two transactions $\mathsf{T}_1 = (\mathsf{T}_1.\mathsf{I}, \mathsf{T}_1.\mathsf{O}, \mathsf{T}_1.\mathsf{K})$, $\mathsf{T}_2 = (\mathsf{T}_2.\mathsf{I}, \mathsf{T}_2.\mathsf{O}, \mathsf{T}_2.\mathsf{K})$ such that $\mathsf{T}_1.\mathsf{I}$ has no common elements with $\mathsf{T}_2.\mathsf{I}$ and $\mathsf{T}_1.\mathsf{O}$ has no common elements with $\mathsf{T}_2.\mathsf{O}$, we can merge[4] them as: $\mathsf{T_M} = (\mathsf{T}_1.\mathsf{I} \cup \mathsf{T}_2.\mathsf{I}, \mathsf{T}_1.\mathsf{O} \cup \mathsf{T}_2.\mathsf{O}, \mathsf{T}_1.\mathsf{K} \cup \mathsf{T}_2.\mathsf{K})$. Furthermore, if a commitment appears in both the input and output lists, it can be deleted (cut) from both without affecting the sum. With this in mind we arrive at:

$$\begin{aligned}
\mathsf{T_M.I} &:= (\mathsf{T}_1.\mathsf{I} \cup \mathsf{T}_2.\mathsf{I}) \setminus (\mathsf{T}_1.\mathsf{O} \cup \mathsf{T}_2.\mathsf{O}) \\
\mathsf{T_M.O} &:= (\mathsf{T}_1.\mathsf{O} \cup \mathsf{T}_2.\mathsf{O}) \setminus (\mathsf{T}_1.\mathsf{I} \cup \mathsf{T}_2.\mathsf{I}) \\
\mathsf{T_M.K} &:= \mathsf{T}_1.\mathsf{K} \cup \mathsf{T}_2.\mathsf{K}
\end{aligned}$$

We consider the ledger L as a special transaction with an empty input set. The output set of the ledger is termed the UTXO set.

$$\mathsf{UTXO} := \mathsf{L.O}$$

The ledger is updated by merging valid transactions onto it. The issue of deciding which transactions are to be merged and in which order is outside the scope of this text. When a transaction is accepted by the network, its outputs are added to the UTXO set, and its inputs deleted from it.

In practice, this can be used to merge all submitted transactions into one transaction per block, and subsequently merge all block transactions into a single transaction. This practice of merging and cutting can decrease the storage requirements while still allowing one to fully verify transactions. It also provides a small level of confidentiality.

4.3 Mimblewimble Anonymity

Due to merging, Mimblewimble blocks contain a single transaction, produced as a merging of all user-submitted transactions during that time. This can obfuscate the transaction graph in the view of adversaries with access only to the block-level, but

[3]In addition, to having 0 value, we are also demonstrating it has 0 serial number, which will be relevant later.

[4]We note here that we allow repetitions in the kernel lists.

individual transactions will still be distinguishable with regard to adversaries with access to the network layer: even with the values hidden, “who paid whom” will be in the clear.

5 Lelantus-MW

While the standard Mimblewimble protocol offers some privacy by obfuscating the transaction graph in the block level, it has two shortcomings: first, small blocks offer less obfuscation and second, the layer of obfuscation collapses if the adversary is able to closely monitor the network. Lelantus offers a design that enables *shielded* transactions, i.e. transactions that store or withdraw value from a universal set without revealing which element they are withdrawing from. This raises the issue of double-spending: if spent elements are unknown they cannot be removed from the set, and thus users might attempt to spend them again. This is solved by adding an extra “serial number” field to the commitments of shielded values. When a shielded value is being spent the serial number is revealed and checked against a list of previously-revealed serial numbers. This prevents double spending (due to the binding property of the commitment scheme), without compromising privacy as serial numbers are not revealed when shielded values are created.

Given the above discussion, we extend our definition of transactions in two ways:

- We allow shielded inputs. Whereas a transparent input “justifies” its existence by appearing in the UTXO set, a shielded one is supported by the combination of a membership proof and a single-use serial number.
- We allow shielded outputs. Whereas transparent outputs simply enter the UTXO set, shielded ones are combined with an additional masking factor and added to the STXO set.

Shielded outputs are in part similar to transparent ones: they have a commitment containing their value and corresponding range proof. To disallow double-spending, they also need to have a serial number, which is hidden in a second commitment, to be combined with the first one when added to the STXO set,. Obviously, the second commitment must carry zero value. Because we wish to exclude serial number components from entering the balance sum, we need to tie the two commitments together in some fashion, which we accomplish by adding the second one to the statement of the range proof.

A shielded output is a tuple $(C_{SO}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO})$ such that:

- C_{SO} is a commitment (i.e a group element)
- π_{SO} is a proof for the relation R'_ρ with statement $C_{SO}, h, \hat{C}_{SO}$ (i.e a range proof counter-signing $\hat{C}_{SO}$).
- $\hat{C}_{SO}$ is a generalised commitment (i.e a group element)
- σ_{SO} is a proof for relation R_{nv} for statement $\hat{C}_{SOj}, k$ (i.e a proof that $\hat{C}_{SO}$ has zero value).

The tuple $\hat{C}_{SO}, \sigma_{SO}$ is called a *ticket* because it can be pre-generated ahead of time and also because it contains the serial number. To prevent (potentially accidental) misuse leading to loss of funds we require that tickets are unique. Concretely, we check that $\hat{C}_{SO}$ has not been used previously.

Shielded inputs are different. Like transparent ones, they contain a commitment carrying value. However, that commitment does not appear (directly) in the STXO set as that would provide no privacy. The input also provides a group element which is interpreted as a hash pre-image of the serial number, accompanied with a signature proving knowledge of the discrete log. The last part of the input is a membership proof of the combined commitment and serial number in STXO.

A shielded input is a tuple $(C_{SI}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI})$ such that:

- C_{SI} is a commitment (i.e a group element)
- σ_{SI} is a proof for relation R_{ns} for statement C_{SI}, h (i.e a proof that C_{SI} has zero serial number).
- Q_{SI} is a group element.
- σ'_{SI} is a proof for relation R_σ for statement Q_{SI} (i.e a proof of knowledge for the log of Q_{SI}).
- π_{SI} is a proof for the relation R_m on statement $(\boldsymbol{C}, s)$ where $\boldsymbol{C} = \frac{\mathsf{STXO}}{C_{SI}}$ snd $s = H(Q_{SI})$).

With the above changes in mind we are able to adapt Definition 8 as follows:

Definition 9. *A transaction* $\mathsf{T} = (\mathsf{T.I}, \mathsf{T.SI}, \mathsf{T.O}, \mathsf{T.SO}, \mathsf{T.K})$ *is valid if:*

- *The list of commitments* $\mathsf{T.I}$ *is a subset of the current* UTXO *set.*
- *For each commitment* $C_O \in \mathsf{T.O}$*, the corresponding proof* π_O *is valid.*
- *For each commitment* C_{Kl} *the corresponding signature* σ_{Kl} *is valid.*
- *For each shielded input,* $(C_{SI}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI})$ *the corresponding proofs,* $\sigma_{SI}, \sigma'_{SI}, \pi_{SI}$*, are valid, and* $H(Q_{SI}) \notin \mathsf{USN}$.
- *For each shielded output,* $(C_{SO}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO})$ *the corresponding proofs* π_{SO}, σ_{SO} *are valid.*
- *The the sum of inputs minus the sum of outputs is equal to the sum of zero-valued commitments:* $\sum_i C_{Ii} + \sum_{i'} C_{SIi'} - \sum_j C_{Oj} - \sum_{j'} C_{SOj'} = \sum_l C_{Kl}$.

5.1 The Shielded Set

As with standard Mimblewimble, the non-shielded outputs of the ledger form the UTXO set. We define the shielded (STXO) set as the list of combined shielded output commitments in the ledger:

$$\mathsf{STXO} := \{C_{SOi} \cdot \hat{C}_{SOi} | (C_{SO}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO}) \in \mathsf{L.SO}\}$$

Items cannot be deleted from the shielded set. In order to disallow double-spending, we keep a list of used serial numbers:

$$\mathsf{USN} := \{H(Q) | (C_{SI}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI}) \in \mathsf{L.SI}\}$$

A note on the Membership Proof Even though the list of shielded outputs consists of generalised commitments, we observe that given a serial number s and a commitment $C = \mathrm{Com}(v; r)$, we can divide the list by C. Assuming one of the original list elements has serial number s and value v with some randomness r', the corresponding quotient will be $\frac{\mathrm{Com}(v,r',s)}{\mathrm{Com}(v,r)} = \mathrm{Com}(0, r' - r, s) = \mathrm{Com}(0, r'', s)$. This allows us to directly use the membership proof of [GK15, BCC+15] without the need to adapt them for the generalised commitments.

6 Multiple Assets

In the previous sections we described Lelantus-MW as a single asset system: value in that asset is represented within a commitmend C in the exponent of h in $ck = (g, h, k)$. It is however possible to generalize this notion to a set of n additional types of assets represented by a vector of n random group elements $\boldsymbol{b} = \{b_1 \dots b_n\} \in \mathbb{G}^n$. For ease of exposition, we will expand the vector $\boldsymbol{b}$ by considering that $b_0 = h$.

We can thus use $\mathrm{Com}_{ck,\boldsymbol{b},i}(v, r, s)$ when committing to values in asset i. For $i = 0$, this is equivalent to using $\mathrm{Com}_{ck}(v, r, s)$.

The descriptions and definitions of section 5 can be made to support these additional currencies with few changes. First off, a commitment carrying value is accompanied with an asset index i, indicating it is related to a single asset b_i for $0 \leq i <= n$ (this is implicitly true in our existing description of both Mimblewimble as well as Lelantus-MW). Second, we adapt the range proofs for both shielded and unshielded outputs to use b_i in the place of h in the statement (in the case where $i = 0$, this collapses to the previous definition). Third, we adapt the serial number check for shielded inputs, i.e σ_{SI} for relation R_{ns} to use statement (C_{SI}, b_i) instead of (C_{SI}, h). Again, for $i = 0$ this is identical to the initial definition.

For $\boldsymbol{b}$ sampled via a random oracle, the binding property of Pedersen commitments ensures that there exists no efficient adversary who is able to interpret a commitment with non-zero value over b_i as one over b_j for $i \neq j$. While this extension is concise and maintains separation amongst different assets, it offers no improvement in privacy compared to the alternative of $n + 1$ single-asset systems operating independently. It does however enable higher-level protocols that can facilitate exchange of values between assets in a more efficient way than what would be possible for separate systems.

6.1 Confidential Assets

The concept of confidential assets [PBF+18, ZYD+19] is a natural addition to multiple asset support: instead of stating the asset(s) of our commitments before proving their validity, we would prefer to prove their validity without stating the asset. The solution given in [PBF+18]and adapted in [ZYD+19] is to use a temporary asset tag t in place of b_i and demonstrate in zero knowledge that t is "equivalent" to some $b_i \in \boldsymbol{b}$ without revealing i. This is performed using the homomorphic property of Pedersen so that the value and serial number of the commitment are unchanged.

Asset blinding Let $b_i \in \boldsymbol{b}$. The user randomly selects $a \leftarrow \mathbb{Z}_q$ and sets $t \leftarrow b_i \cdot g^a$. We refer to t as a "blinded" asset.

Commitment equivalence Let $C = \mathrm{Com}_{ck,\boldsymbol{b},i}(v,r,s) = b_i^v g^r k^s$, and let $t = b_i \cdot g^a$ as above. Then, substituting $b_i = t \cdot g^{-a}$ we have $C = t^v g^{r-av} k^s$. That is, we can use C as a commitment $C = \mathrm{Com}_{ck',\boldsymbol{b},i}(v, r - va, s)$, where $ck' = (g,t,k)$. This implies that commitments under b_i can be used unchanged with blinded values of b_i with only local calculations from the user.

Asset surjection proofs Commitment equivalence hinges on t being correctly formed. For example it is simple to show that if t includes a k component, the serial of C with respect to to t would be different than the one with respect to the original base b_i. We thus need the user to demonstrate they know i, r such that $t = b_i \cdot g^a$, or equivalently that they know i, r so that $b_i/t = g^a$ i.e they know the logarithm over g of b_i/t for some i. The user can thus use the protocol for R_m on statement $\left(\frac{\boldsymbol{b}}{t}, 0\right)$ – the 0 derives from the fact that neither $\boldsymbol{b}$ or t involve serial number components over k.

6.2 Lelantus-CLA

We now expand our definitions of outputs, inputs and valid transactions to account for confidential assets:

A shielded CA output is a tuple $(C_{SO}, t, \pi_{ASP}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO})$ such that:

- C_{SO} is a commitment (i.e a group element)
- t is a blinded asset (i.e a group element)
- π_{ASP} is a proof for the relation R_m with statement$\left(\frac{\boldsymbol{b}}{t}, 0\right)$ (i.e an asset surjection proof for t).
- π_{SO} is a proof for the relation R'_ρ with statement $C_{SO}, t, \hat{C}_{SO}$ (i.e a range proof counter-signing $\hat{C}_{SO}$).
- $\hat{C}_{SO}$ is a generalised commitment (i.e a group element)
- σ_{SO} is a proof for relation R_{nv} for statement $\hat{C}_{SOj}, k$ (i.e a proof that $\hat{C}_{SO}$ has zero value).

A shielded CA input is a tuple $(C_{SI}, t, \pi_{ASP}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI})$ such that:

- C_{SI} is a commitment (i.e a group element).
- t is a blinded asset (i.e a group element).
- π_{ASP} is a proof for the relation R_m with statement$\left(\frac{\boldsymbol{b}}{t}, 0\right)$ (i.e an asset surjection proof for t).
- σ_{SI} is a proof for relation R_{ns} for statement C_{SI}, t (i.e a proof that C_{SI} has zero serial number).
- Q_{SI} is a group element.
- σ'_{SI} is a proof for relation R_σ for statement Q_{SI} (i.e a proof of knowledge for the log of Q_{SI}).
- π_{SI} is a proof for the relation R_m on statement $(\boldsymbol{C}, s)$ where $\boldsymbol{C} = \frac{\mathsf{STXO}}{C_{SI}}$ snd $s = H(Q_{SI})$).

Supply In the operation of the system, value may be created according to some higher level protocol which specifies asset creation and minting. We leave the details of this protocol undefined, but assume that supply is represented via a tuple $\mathsf{T.S} = (\boldsymbol{v}, V_{\mathsf{Sup}})$, so that the amount of value created over each asset is represented by a vector $\boldsymbol{v}$ that can be calculated by all users based on some information V_{Sup}. With the above changes in mind we are able to adapt Definition 9 as follows:

Definition 10. *A CA transaction* $\mathsf{T} = (\mathsf{T.I}, \mathsf{T.SI}, \mathsf{T.O}, \mathsf{T.SO}, \mathsf{T.K}, \mathsf{T.S})$ *is valid if:*

- *The list of commitments* $\mathsf{T.I}$ *is a subset of the current* UTXO *set.*
- *For each commitment* $C_O \in \mathsf{T.O}$*, the corresponding proof* π_O *is valid.*
- *For each commitment* C_{Kl} *the corresponding signature* σ_{Kl} *is valid.*
- *For each shielded input,* $(C_{SI}, t, \pi_{ASP}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI})$ *the corresponding proofs,* $\pi_{ASP}, \sigma_{SI}, \sigma'_{SI}, \pi_{SI}$*, are valid, and* $H(Q_{SI}) \notin \mathsf{USN}$*.*
- *For each shielded output,* $(C_{SO}, t, \pi_{ASP}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO})$ *the corresponding proofs* $\pi_{ASP}, \pi_{SO}, \sigma_{SO}$ *are valid.*
- *The tuple* $\mathsf{T.S} = (\boldsymbol{v}, V_{\mathsf{Sup}})$ *is consistent with the higher level asset creation protocol.*
- *The the sum of inputs plus suply minus the sum of outputs is equal to the sum of zero-valued commitments:* $\sum_i C_{Ii} + \mathsf{Sup}(\boldsymbol{v}) + \sum_{i'} C_{SIi'} - \sum_j C_{Oj} - \sum_{j'} C_{SOj'} = \sum_l C_{Kl}$*.*

This implies that supply is simple to merge supply across transactions: for two transactions $\mathsf{T}_1, \mathsf{T}_2$ each with supply $\mathsf{T_i.S} = (\boldsymbol{v}_i, V_{\mathsf{Sup},i})$ the merged supply is $\mathsf{T_M.S} = (\boldsymbol{v}_1 + \boldsymbol{v}_2, V_{\mathsf{Sup},1} \cup V_{\mathsf{Sup},2})$.

7 Security Properties

We are interested in two security properties: first, we require that over each asset type value cannot be created or destroyed except as allowed by the protocol. We term this "Supply Persistence" and define it bellow. Intuitively, this means that the balance of unshielded outputs minus unshielded inputs, plus that of shielded shielded outputs minus that of shielded inputs equals the supply value in $\mathsf{T.S}$. This implies that assets do not interact inside transactions.

Definition 11 (Supply Persistence)**.** *For any PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *can output a list of transactions* $\mathsf{T^i} = (\mathsf{T^i.I}, \mathsf{T^i.SI}, \mathsf{T^i.O}, \mathsf{T^i.SO}, \mathsf{T^i.K}, \mathsf{T^i.S})$ *for* $i = 0 \ldots k$ *and an opening* $\boldsymbol{v^*}, r^*$ *such that:*

- *The list of commitments in* $\mathsf{T^i.I}$ *is a subset of the* UTXO *set formed by the previous transactions* $\mathsf{T^0} \ldots \mathsf{T^{i-1}}$*, or empty for* $\mathsf{T^0}$*.*
- *For each commitment* $C_O \in \mathsf{T^i.O}$*, the corresponding proof* π_O *is valid.*
- *For each commitment* $C_{Kl} \in \mathsf{T^i.K}$ *the corresponding signature* σ_{Kl} *is valid.*
- *For each shielded input,* $(C_{SI}, t, \pi_{ASP}, \sigma_{SI}, Q_{SI}, \sigma'_{SI}, \pi_{SI}) \in \mathsf{T^i.SI}$ *the corresponding proofs,* $\pi_{ASP}, \sigma_{SI}, \sigma'_{SI}, \pi_{SI}$*, are valid with regards to the shielded set formed by the previous transactions* $\mathsf{T^0} \ldots \mathsf{T^{i-1}}$*, or empty for* $\mathsf{T^0}$*, and additionally* $H(Q_{SI}) \notin \mathsf{USN}$*.*

- *For each shielded output,* $(C_{SO}, t, \pi_{ASP}, \pi_{SO}, \hat{C}_{SO}, \sigma_{SO}) \in \mathsf{T^i}.\mathsf{SO}$ *the corresponding proofs* $\pi_{ASP}, \pi_{SO}, \sigma_{SO}$ *are valid.*
- *For each transaction, the sum of inputs plus suply minus the sum of outputs is equal to the sum of zero-valued commitments:* $\sum_i C_{Ii} + \mathsf{Sup}(\boldsymbol{v}) + \sum_{i'} C_{SIi'} - \sum_j C_{Oj} - \sum_{j'} C_{SOj'} = \sum_l C_{Kl}$.
- *After transaction* T^k*,* UTXO *consists of a single element* C^**, such that* $C^* = \mathrm{Com}_{ck,\boldsymbol{b}}(v_0^*, r^*, 0, v_1^*, \ldots, v_n^*)$.
- *For at least one component* l*,* $v_l^* > \mathsf{TSup}_l$*, where* $\mathsf{TSup} = \sum_{i=0}^{k} \boldsymbol{v}_i$.

is neglibigle

Proof (Sketch). We will show that if a PPT adversary $\mathcal{A}$ can ouptut $\mathsf{T^i}$ for $i = 0 \ldots k$ and an opening $\boldsymbol{v^*}, r$ as specified, then we can recover with high probability a non-trivial discrete log relation between the elements of ck and $\boldsymbol{b}$, which in turn reduces to the discrete logarithm problem. Towards that, we rewind the adversary to extract the asset types, values, serials and randomness used for all shielded or unshielded outputs. For unshielded inputs we can determine the corresponding output directly. For shielded ones, we first extract the position of the commitment using the proof of membership. Finally, we extract the randomisers from kernels. If any of our extractions fail, we give up, but that only happens with negligible probability. If the extraction indicated that a shielded output has been doublespent, the openings directly give us a discrete log relation.

Assuming every extraction is successful, we have extracted a decomposition of each commitment into a vector of known exponents. This allows us to recursively construct the value of C^* independently of the adversarial opening $\boldsymbol{v^*}, r^*$ starting with the inputs of $\mathsf{T^k}$. If the calculated value disagrees with the adversarial opening, we directly arrive at the required discrete log relation, as we have two openings for C^*. If the openings match, we iterate backwards over $\mathsf{T^i}$ looking for a transaction where for at least one asset type l the value of outputs exceeds that of inputs plus supply.

Unshielded outputs cannot be doublespent as they are deleted once they are spent, and we have determined during out extraction that no shielded outputs have been doublespent either. Thus, there must be at least one transaction $\mathsf{T^i}$ with mismatch of input, output and supply values, as otherwise we would have $v_l^* \leq \mathsf{TSup}_l$ across all assets l. □

Furthermore, we require that users cannot "steal" value from other users. Informally, this requires that an adversary who has been assigned and sent a total of value of $\boldsymbol{v} = (v_0, \ldots, v_n)$ across all asset types, is unable to produce an opening to an unspent output of value $v_j + \delta$ for any j and $\delta > 0$. Informally, this relies on the zero knowledge property of our proof systems, so that the adversary gains no information from transactions not addressed or initiated by him. Knowledge soundness prevents adversaries from misrepresenting received transactions. A third requirement is a way for users to transaction outputs in a way that allows the receiver to spend them, but the sender cannot. We describe this in the next section. We defer a complete proof in the vein of [FOS19] as a formal treatment of the property has additional technical requirements wrt the simulation extractability of bulletproofs.

8 Transacting using Lelantus-CLA

At a high level, sending funds consists of creating an output that the receiver can subsequently open. The receiver can then use the commitment as an input to one of their transactions. The data structures and proofs used for shielded transactions imply that “owners” of value can spend it as they wish, with strong privacy guarantees. However, the “owner” is defined as the person who knows the relevant witnesses to construct a proof. This raises two issues:

- How is it possible for the sender to create commitments they cannot open?
- How will the intended recipient be able to spend them?

The first issue is easy to handle, by means of a Diffie-Hellman like protocol, as with standard mimblewimble. The second issue is the main focus of this chapter. While a trivial solution would be to have the sender simply send the necessary data out of band (e.g. via email), it is obviously preferable if this can happen in-band and at the same time without introducing any additional overheads.

We rely on the concept of “Coloring”, which is originally related to protocols used to ascribe and recognise additional properties to otherwise ordinary bitcoins without breaking compatibility with existing software. In our case, coloring has to do with establishing a covert channel [LMS05, BDI$^+$96] over the randomizers used in the zero-knowledge proofs used when creating a commitment (i.e schnorr signatures and range proof bulletproofs). This will enable the intended recipient to both recognise commitments addressed to her, as well as extract the necessary witnesses to actually spend them at a later time.

8.1 Bulletproof Coloring

A requirement for direct anonymous payments is for an in-band method of privately notifying the recipient of a payment. Coloring offers a way of doing so by modifying the prover’s randomness generation inside a bulletproof. For some of the blinders in the proof the corresponding randomness will be generated as a deterministic function of the payee’s address. This is done in a way that is computationally indistinguishable from the standard version for any outsiders, but enables the intended recipient to both detect the payment as one directed to them, and also extract a short message embedded within.

We begin, by reproducing the standard range proof for commitments as described in [Jiv19]. For brevity, we omit the logarithmic inner product argument as it is unchanged from the original.

In the protocol, the t_1, t_2 values are derived from the following polynomials, where $l(X), r(X)$ are vector polynomials linear in X, and $t(X)$ is a scalar polynomial quadratic in X:

$$\begin{aligned} l(X) &= (\boldsymbol{a}_L - z \cdot \mathbf{1}^n) + \boldsymbol{s}_L \cdot X && \in \mathbb{Z}_p^n[X] \\ r(X) &= \boldsymbol{y}^n \circ (\boldsymbol{a}_R + z \cdot \mathbf{1}^n + \boldsymbol{s}_R \cdot X) + z^2 \cdot X^2 && \in \mathbb{Z}_p^n[X] \\ t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 && \in \mathbb{Z}_p[X] \end{aligned} \tag{1}$$

$\underline{P_{\text{sig}}(gk, crs, (C, b), (r, v))}$		$\underline{V_{\text{sig}}(gk, crs, (C, b))}$
$\alpha \leftarrow \mathbb{Z}_p,\ \rho \leftarrow \mathbb{Z}_p$		
$\boldsymbol{s}_L, \boldsymbol{s}_R \leftarrow \mathbb{Z}_p^n$		
Let $\boldsymbol{a}_L \in \{0,1\}^n$ s.t. $\sum a_i \cdot 2^i = v$		
$\boldsymbol{a}_R = \boldsymbol{a}_L - \mathbf{1}^n \in \mathbb{Z}_p^n$		
$A = b^{\alpha} \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} \in \mathbb{G}$		
$S = b^{\rho} \boldsymbol{g}^{\boldsymbol{s}_L} \boldsymbol{h}^{\boldsymbol{s}_R} \in \mathbb{G}$	$\xrightarrow{\text{A,S}}$	
Let t_1, t_2 be set as per eq. (1)	$\xleftarrow{y, z \leftarrow \mathbb{Z}_p^*}$	Let $h_i' = h_i^{y^{-i+1}}$ for $i \in [1, n]$
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_p$;		
$T_i = g^{t_i} h^{\tau_i}$ for $i = \{1, 2\}$	$\xrightarrow{T_1, T_2}$	Let $P = A \cdot S^x \cdot \boldsymbol{g}^{-z} \cdot ((\boldsymbol{h}')^{z \cdot \boldsymbol{y}^n + z^2 \cdot \boldsymbol{2}^n}) \in \mathbb{G}$
$\boldsymbol{l} = l(x)$	$\xleftarrow{x \leftarrow \mathbb{Z}_p^*}$	Accept iff:
$\boldsymbol{r} = r(x)$		
$\hat{t} = \langle \boldsymbol{l}, \boldsymbol{r} \rangle \in \mathbb{Z}_p$		
$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot r \in \mathbb{Z}_p$		$g^{\hat{t}} b^{\tau_x} = V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2}$
$\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$	$\xrightarrow{\tau_x, \mu, \hat{t}, \boldsymbol{l}, \boldsymbol{r}}$	$P = b^{\mu} \cdot \boldsymbol{g}^{\boldsymbol{l}} \cdot (\boldsymbol{h}')^{\boldsymbol{r}}$
		$\hat{t} = \langle \boldsymbol{l}, \boldsymbol{r} \rangle \in \mathbb{Z}_p$

Figure 1: Range proof for R_ρ

8.1.1 Recipient verification

In the protocol of Figure 1, we observe that α and ρ are random elements of $\mathbb{Z}_p$ to be used as blinders. As a warmup, let us consider a modified protocol where they are instead generated via a keyed PRF. For example, we replace $\boxed{\alpha \leftarrow \mathbb{Z}_p,\ \rho \leftarrow \mathbb{Z}_p}$ by $\boxed{\alpha \leftarrow H(S \| \text{“}alpha\text{”} \| \mathsf{seed}),\ \rho \leftarrow H(S \| \text{“}rho\text{”} \| \mathsf{seed})}$, where S represents the concatenation of reference strings gk, ck, crs with the statement C, and $\mathsf{seed} \in \mathbb{Z}_p$ represents the coloring key.

Theorem 1 (Informal)**.** *The modified range proof protocol is complete, has witness-extended emulation and is computationally HVZK.*

Proof. Completeness holds by inspection, as the modified values of α, ρ are used consistently with the original protocol.

We also observe that witness-extended emulation is not impacted, as the definition is not conditioned on any particular prover. Thus, we are able to use the same extractor to repeat the proof with no changes needed (in a way, we consider the modified prover to be an adversarial one and carry out the extraction regardless).

Assuming the hash outputs are computationally indistinguishable from random ensures that zero knowledge still holds, albeit computationally: if the existing simulator fails then an adversary is able to efficiently distinguish the hash function from a random one. □

Even though security is only minimally impacted, a user with the correct value of seed will be able to generate α', ρ' independently, and use the challege x from the transcript to produce $\mu' = \alpha' + \rho' \cdot x$. She can then "check" the value of μ from the transcript against μ' to determine if she is the intended recipient.

8.1.2 Conditional witness extraction

We now observe we can develop the same idea further. We further modify the protocol so that τ_1, τ_2 are also derived via the PRF. This enables us to revisit τ_x, that is $\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot r$. Assuming we know the correct value of seed, this enables us to reconstruct τ_1, τ_2, and using our knowledge of x, this allows us to extract r.

8.1.3 In-Band Data

We are able to expand on our previous idea, by adding some additional data δ as a perturbation to α, i.e. $\alpha \leftarrow H(S||\text{"}alpha\text{"}||k) + \delta$. This enables us to recover δ from μ by the above equation. We are still able to check if the value of seed is correct by enforcing a specific range of allowed values for δ that is small compared to the group order q, and accepting the possibility of false positives.

8.1.4 Additional Data

If we additionally know the value specified via the bulletproof, we can also add (and extract) additional data as perturbations in the $\boldsymbol{s}_L, \boldsymbol{s}_R$ values. We can perform the extraction from the $\boldsymbol{l}, \boldsymbol{r}$ vectors since they only depend on v and public data.

8.1.5 Bulletproof Coloring in practice

In the logarithmic version used in the implementation, we are able to hide one value mod q in each of $\boldsymbol{s}_L, \boldsymbol{s}_R$, as they are subsequently compressed, preventing further extraction. We are still able to extract the randomizer r, as well as a third value mod q using α. Extracting any value is conditional on knowing the correct seed used in the PRF, and for $\boldsymbol{s}_L, \boldsymbol{s}_R$, we also need to know the value v. For this reason, we assume v is contained in the value contained inside α. Note here, that the range of values is significantly smaller than q, so there is additional space for data.

In total, we are able to extract 2 arbitrary values $\mod q$ as well as v, r, and in the case of a CLA transaction, the asset index i and randomizer a. We leave the additional data unspecified, but they may be used to facilitate transaction metadata or future functionality. The only additional information required for the receiver to be able to spend the output is knowledge of the log of the serial number pre-image. We will cover this in the next section, as it is not included in the bulletproof.

8.2 Sending and Receiving Shielded Values

We now revisit our main goal: enabling senders to create shielded outputs they cannot spend (and allowing receivers to spend them).

The bulletproof coloring technique described above gives us ample space to extract data from π_{SO}, including one of the needed witnesses (r) for C_{SO}. We will now focus our attention to enabling the receiver to obtain the PRF seed as well as to recognise the transaction. We do so by applying similar techniques to the $\hat{C}_{SO}, \sigma_{SO}$

part of the shielded output (i.e the ticket), and also adding a DH-like protocol so that receivers can determine seed as well as the serial number pre-image.

8.2.1 Addresses

A receiver first creates a triple keypair $P_{1,g}, P_{1,k}, P_2$ where $P_{1,g} = g^{s_1}, P_{1,k} = k^{s_1}$ and $P_2 = k^{s_2}$ and $s_i \leftarrow \mathbb{Z}_q$. The corresponding secret keys are (s_1, s_2), used for recognizing and spending payments. As such, s_1 can be revealed to a semi-trusted service with no risk to funds. The triple $(P_{1,g}, P_{1,k}, P_2)$ is a user's address, and it enables other users to directly pay her.

8.2.2 Tickets

To make a commitment recognisable, the sender will create $\hat{C}_{SO}$ in a specific way. First the sender samples a random value $w \leftarrow \mathbb{Z}_q$ and sets $skp \leftarrow H_1(w)$. Then $Q \leftarrow P_2^{skp}$. Finally $s \leftarrow H(Q)$ and $\hat{C}_{SO} \leftarrow g^s k^w$. Looking forward, the sender does not know the log of Q w.r.t g. The intended recipient however could calculate it as $w \cdot s_2$, if provided with w.

The commitment is used to seed the randomness of the generalised Schnorr signature σ_{SO} as follows: $\mathsf{seed} \leftarrow (P_{1,g}^s \cdot P_{1,k}^w)^{H_1(\hat{C}_{SO})}$. This implies $\mathsf{seed} = \hat{C}_{SO}^{s_1 \cdot H_1(\hat{C}_{SO})}$ i.e the recipient can determine the seed from s_1 and $\hat{C}_{SO}$. Once the key is determined, the recipient can extract the witnesses s, w from the signature, using the techniques of section 8.1.3 which constitutes a full opening of $\hat{C}_{SO}$. Combined with the above, this also allows the recipient to calculate the log of Q, which is required for spending, check that the seed value is valid, and finally extract the values from the bulletproof.

8.2.3 Ticket Distribution

Looking forward, when the receiver attempts to spend the received commitment, the value of Q will be revealed as Q_{SI} in the shielded input. This will allow the sender to detect that a particular shielded input has been spent. Conceptually, this can be mitigated by having the receiver immediately re-send the same value to herself, but that would be wasteful in practice. A more efficient solution is to have the receiver pre-compute a number of tickets and make them available to known parties. This can substitute distributing addresses.

The only required change is that the receiver needs to communicate seed in addition to $\hat{C}_{SO}, \sigma_{SO}$, as the calculation to produce it requires either the pair (s, w) or s_1, of which the sender knows neither. The receiver is able to independently create C_{SO}, π_{SO} at a later time. Note that whilst $\hat{C}_{SO}, \sigma_{SO}$ do not depend on C_{SO}, π_{SO}, the statement of the bulletproof π_{SO} explicitly includes $\hat{C}_{SO}$ so that the two parts are effectively "tied".

References

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[BCC$^+$15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on ddh. In *European Symposium on Research in Computer Security*, pages 243–265. Springer, 2015.

[BDI$^+$96] Mike Burmester, Yvo G Desmedt, Toshiya Itoh, Kouichi Sakurai, Hiroki Shizuya, and Moti Yung. A progress report on subliminal-free channels. In *International Workshop on Information Hiding*, pages 157–168. Springer, 1996.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM conference on Computer and communications security – CCS 1993*, pages 62–73. ACM, 1993.

[FOS19] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 657–689, Cham, 2019. Springer International Publishing.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—Crypto'86*, pages 186–194. Springer, 1987.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – EUROCRYPT 2008*, pages 379–396. 2008.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.

[Jed16] TE Jedusor. Mimblewimble. Defunct hidden service, 2016. Copy retrieved from `https://github.com/mimblewimble/docs/wiki/Mimblewimble-origin`.

[Jiv19] Aram Jivanyan. Lelantus: A new design for anonymous and confidential cryptocurrencies. Cryptology ePrint Archive, Report 2019/373, 2019. `https://ia.cr/2019/373`.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.

[LMS05] Matt Lepinski, Silvio Micali, and Abhi Shelat. Fair-zero knowledge. In *Theory of Cryptography Conference*, pages 245–263. Springer, 2005.

[PBF$^+$18] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. In *International Conference on Financial Cryptography and Data Security*, pages 43–63. Springer, 2018.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[ZYD$^+$19] Yi Zheng, Howard Ye, Patrick Dai, Tongcheng Sun, and Vladislav Gelfer. Confidential assets on mimblewimble. Cryptology ePrint Archive, Report 2019/1435, 2019. `https://eprint.iacr.org/2019/1435`.