Oscausi - Practical Private Electronic Cash from Lelantus and MimbleWimble

Lasse Herskind, Panagiota Katsikouli*, and Nicola Dragoni
DTU Compute, Lyngby, Denmark
s153746@student.dtu.dk, {panka,ndra}@dtu.dk

Abstract

After challenging the privacy guarantees of Bitcoin, a lot of alternatives have been proposed to enhance the privacy-properties of Bitcoin. While Zcash (one of such alternatives) significantly improves the privacy of Bitcoin, its two-coin design with a public base-coin permits critical attacks to happen. In this paper we propose Oscausi, an anonymous payment system that supports practical confidential and anonymous transactions without a public basecoin nor a trusted setup. The scheme is inspired by the Lelantus and MimbleWimble schemes, joining the ideas into one system, with a confidential basecoin and privacy through a shielded pool. Our proposal supports non-interactive transaction aggregation across shielded and unshielded transactions. The scheme is built upon well-known cryptography, is easily auditable and requires no trusted setup.

Keywords: Anonymity, Confidentiality, Cryptocurrencies, Electronic Cash, Privacy, Zero-Knowledge

1 Introduction

In 2008, the pseudonymous actor Satoshi Nakamoto published a new monetary system called Bitcoin [38], a purely digital currency managed by a network of untrusted users (or nodes). Bitcoin ensures computational integrity of each transaction taking place within the network, since those are subject to re-evaluation by every node in the network. That is possible exactly because the full set of transaction information is publicly available, limiting the privacy of the transaction history and of the system in general [35, 25, 1, 17, 11, 33].

A variety of protocols has been proposed to combat this. Some of these protocols attempt to provide a shield against attacks that exploit the graphs presenting transactions between network nodes, or the P2P network itself, and vary from cryptographic schemes utilising zero-knowledge proofs and shielded addresses to network propagation mechanisms that hide the IP-address of the transaction source. A brief summary of those, borrowed from [26], is presented in Table 1; we note however, that this work will only be relating to transaction graph approaches.

In particular, Monero [47] and Zcash (with its Zerocash implementation [3]) are perhaps among the most popular of these protocols, utilising ring-confidential transactions [39] and Zero-Knowledge Succinct Argument of Knowledge proofs (ZK-SNArKs) [3] respectively. Although Monero does not rely on a trusted setup, it provides small anonymity-sets with substantial proofs [32, 37, 48, 49]. On the other hand, Zcash provides ample anonymity-sets (2^{32}) with small proofs, but requires a trusted setup and enforces privacy as an opt-in feature, with most transactions occurring on a Bitcoin-like public coin[31]. Following the two-layer idea of Zcash, the Lelantus scheme [28], also provides privacy on top of a public coin, but without the trusted setup. To contend privacy-by-choice, the scheme MimbleWimble was introduced [41], which offers confidentiality by default without the enhanced anonymity of Zcash.

Journal of Internet Services and Information Security (JISIS), volume: 10, number: 2 (May 2020), pp. 16-34
DOI: *10.22667/JISIS.2020.05.31.016*

*Corresponding author: Department of Computer Science and Mathematics, Technical University of Denmark, 115/322, 2800 Kongens Lyngby, Denmark

One of the conclusions in [26] is that Zero-Knowledge systems offer a better chance of truly hiding the relationship between sources and destinations of transactions than decoybased schemes. However, the use of a trusted setup in a un-trusted environment is a problematic weakness that allows for possible attacks. It has become apparent, therefore, the need of a practical scheme that ensures confidentiality and anonymity, without the requirement of a trusted setup.

The goal of this paper is to fill the gap discussed in the previous paragraphs. Taking inspiration from the privacy properties offered by Lelantus and MimbleWimble, we propose a variation of the Lelantus scheme [28], which runs on top of the cryptocurrency scheme MimbleWimble [41], thereby working as a second-layer privacy solution on top of the confidentiality provided by Pedersen commitments [40]. Our proposal is enforced through primitives which supports non-interactive transaction aggregation across shielded and unshielded transactions, thereby permitting joinsplit transactions. The scheme is built upon well-known cryptography, is easily auditable and requires no trusted setup.

Paper Outline: In Section 2 we introduce the general notation used in the paper, as well as introductory knowledge of commitment schemes and zero-knowledge proofs utilised in our scheme. Section 3 and section 4 introduce the MimbleWimble and Lelantus, respectively. In section 5 we present the details of our scheme, called Oscausi, which we validate in Section 6. Conclusive remarks and future directions are highlighted in Section 7.

Method	Objective	Disadvantages	Papers
Centralised Mixing	Improve taint resistance	Require trust in a server. Limited anonymity if low server usage.	[35, 7]
Decentralised Mixing	Improve taint resistance	Limited anonymity-set. Coordination with other participants, vulnerable to DoS and sybil attacks.	[34, 43, 44, 42]
Non-interactive Mixing	Improve taint resistance	Anonymity-set is limited in size, small blocks provide bad anonymity.	[45, 41]
Coin Swapping	Imrpove tain resistance	Require trust in the ”sender”, as he can leak the true transaction.	[51]
Ring signatures	Improve taint resistance	Limited anonymity-set. Rings can be ”pruned”.	[47, 19, 50, 37, 48, 49]
Stealth addresses	Hide receiver	Participant need to check if they are receiver of incoming transactions.	[50, 47]
Confidential transactions	Hide amounts	Requires proofs for verification.	[47, 41, 40]
ZK-proofs	Unlinkability	Computationally expensive to verify in comparison to public (or SNArKS).	[8]
ZK-SNArKs	Unlinkability	Requires a trusted setup, and *stronger* assumptions.	[20, 15, 31, 3, 36, 21]
Dandelion(++)	Hide IP	Vulnerable to DoS and Sybil-attacks	[5, 14]
Mixnets	Hide IP	Vulnerable to DoS and sybil-attacks.	[44, 42, 11, 30]
Tor & I2P	Hide IP	Vulnerable to Relative Reality attacks.	[13, 33, 4]

Table 1: A summary overview of privacy-enhancing proposals against attacks exploiting analysis of transaction graphs and P2P network analysis. Table extracted from [26]

2 Preliminaries

In this section we present the notation followed in the paper, as well as the preliminary theory of commitment schemes and zero-knowledge proofs, which constitute the essential building blocks in our approach.

2.1 General Notation

Let $\mathbb{G}$ denote a cyclic group of prime order p, $\mathbb{Z}_p$ denote the ring of integers modulo p, and $\mathbb{Z}_p^*$ be $\mathbb{Z}_p \backslash \{0\}$. Drawing a random element from $\mathbb{Z}_p^*$ is then written $\leftarrow_\$ \mathbb{Z}_p^*$. Within the cyclic group $\mathbb{G}$, generators are denoted by $\mathsf{g}, \mathsf{h}, \mathsf{j} \in \mathbb{G}$ and elements by capital letters, e.g., $C \in \mathbb{G}$. A field element of the cyclic group is denoted with a lower-case letter, e.g., $a \in \mathbb{Z}_p$. A proof is denoted by $\pi_{\text{type}} = \text{type}(w, X)$, where w is the witness for some commitment X. Let $\mathbb{Z}_p^n$ and $\mathbb{G}^n$ be the vectorspaces of $\mathbb{Z}_p$ and $\mathbb{G}$ respectively; then a bold font is a vector, e.g., $\boldsymbol{a} \in \mathbb{Z}_p^n$ is a vector $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}_p$, $\mathsf{g} \in \mathbb{G}^n$ a vector of generators etc. Let $\mathscr{H}$ denote a hashfunction, acting as a random oracle [2]. Let σ_q be a signature with the private key q. Further we let $\overset{?}{\cdot}$ denote an assertion that the relation $\cdot$ holds.

2.2 Commitment schemes

A commitment is a cryptographic primitive that allows a user to commit to a chosen value v in such a manner that v is hidden until the user chooses to reveal it. This is done by utilising a random value r (also known as the blinding factor), to create $\text{com}(v, r)$, which is a commitment to v using r. By revealing the opening (v, r), anybody can validate that v was indeed the value hidden by the commitment. The commitment scheme we will use, must be hiding and binding.

Hiding: A commitment scheme is hiding if it does not reveal the value v. Let a probabilistic polynomial-time algorithm $\mathscr{A}$ act as an adversary that generates two values, v_0 and v_1. Given a commitment com to either v_0 or v_1, the adversary returns $a = \{0, 1\}$ such that com is the commitment of v_a. If the commitment scheme enforces Equation 1 [24], then the adversary has a negligible advantage over random guessing and the scheme is said to be hiding. If the probability in Eq. 1 is exactly $\frac{1}{2}$, the scheme is said to be perfectly hiding, meaning that an adversary has no advantage over random guessing.

$$\begin{aligned} &\Pr\left[(v_0, v_1) \leftarrow \mathscr{A}; a \leftarrow_\$ \{0,1\}; \right. \\ &\qquad \left. C = \text{com}(v_a) : \mathscr{A}(C) = a\right] \approx \frac{1}{2} \end{aligned} \tag{1}$$

Binding: A commitment scheme is binding if it can only open to one value v. Let us again assume an adversary $\mathscr{A}$ that generates two openings (v_0, r_0) and (v_1, r_1), which open the same commitment. If the probability of finding two such openings is negligible (see Equation 2), the commitment scheme is said to be *binding*. If the probability of finding such two openings is exactly 0, the scheme is said to be perfectly binding. Strongly binding is a commitment that can only be opened by one opening, i.e., to one value v with one random factor r.

$$\begin{aligned} \Pr\left[(v_0, v_1, r_0, r_1) \leftarrow \mathscr{A} : (v_0) \neq (v_1)\right. \\ \left. \wedge\, \text{com}(v_0, r_0) = \text{com}(v_1, r_1)\right] \approx 0 \end{aligned} \tag{2}$$

Homomorphism: A commitment scheme is homomorphic if *adding* the commitments $\text{com}(v_0, r_0)$ and $\text{com}(v_1, r_1)$, results in a commitment to $\text{com}(v_0 + v_1, r_0 + r_1)$. In this paper, we use multiplicative notation for adding commitments (see Equation 3)[1] in order to distinguish between the commitment and the field elements operations.

$$\text{com}(v_0, r_0) * \text{com}(v_1, r_1) = \text{com}(v_0 + v_1, r_0 + r_1) \tag{3}$$

Nevertheless, we denote a *"sum"* of commitments to be derived as $\prod_{i=0}^{n-1} C_i$, because it is equal to a commitment to the sums of v's and r's, i.e., $\prod_{i=0}^{n-1} C_i = \text{com}\left(\sum_{i=0}^{n-1} v_i, \sum_{i=0}^{n-1} r_i\right)$. Bluntly put, in our work, *adding* commitments will be denoted by multiplicative operator, and *subtracting* by divisive.

[1] In the literature we also meet additive notation, $\text{com}(v_0, r_0) + \text{com}(v_1, r_1) = \text{com}(v_0 + v_1, r_0 + r_1)$ [9]

Pedersen commitment: The *Pedersen* commitment scheme [40] is a homomorphic scheme that is perfectly hiding and computationally binding, under the discrete log assumption. The scheme uses generators within the cyclic group $\mathbb{G}$, to provide commitments that are group-elements. The commitment is straightforward and is computed as:.

$$\text{com}(v, r) = \mathsf{g}^v * \mathsf{h}^r \tag{4}$$

The scheme can be extended to support commitments to vectors, by using a vector of generators g:

$$\text{com}(\boldsymbol{v}; r) = \mathsf{g}_0^{v_0} * \mathsf{g}_1^{v_1} * \cdots * \mathsf{g}_{n-1}^{v_{n-1}} * \mathsf{h}^r \tag{5}$$

From Eq. 5, a double-blinded commitment [28] can be built as:

$$\text{com}(s, v, r) = \mathsf{g}^s * \mathsf{h}^v * \mathsf{j}^r \tag{6}$$

2.3 Zero-Knowledge Proofs

Informally, a zero-knowledge proof supports a *Prover* in convincing a *Verifier* about a statement (or relation) R between a witness w and a commitment X without revealing the witness [22]. As an example, a range proof [9] uses the public commitment $V = \text{com}(v, \gamma)$ to prove that the value v is within a range without disclosing it. The relation is seen in Equation 7.

$$R = \{\mathsf{g}, \mathsf{h}, V \in \mathbb{G}; v, \gamma \in \mathbb{Z}_p \mid V = \mathsf{g}^v * \mathsf{h}^\gamma \wedge v \in [0, 2^n - 1]\} \tag{7}$$

For such proofs to be zero-knowledge, they must satisfy the following properties [24]:

- **Completeness:** If the *Prover* knows the *witness* w for some commitment X where $(w, X) \in R$, the *Verifier* must be convinced with probability 1. I.e., a valid proof will never be rejected by an honest *Verifier*.

- **Soundness**: If $(w, X) \notin R$, a *Verifier* that follows the protocol honestly will only be convinced otherwise with negligible probability.

- **Zero-Knowledge**: If $(w, X) \in R$, and the *Prover* follows the protocol, the *Verifier* will learn nothing more other than the fact that the statement holds.

An interactive protocol (or Σ-protocol) is a 3-move protocol [24] that supports zero-knowledge proofs. These 3 moves are: *i)* an initial message with statements (or commitments) by the *Prover*, *ii)* a challenge from the *Verifier* and *iii)* the *Prover's* response based on the initial message and the Verifier's challenge. After this interaction, the *Verifier* can accept or reject the proof as convincing. Because the *Prover* does not know what challenge the *Verifier* will pick, and has already committed to the values within her proof, she will only be able to construct a valid proof if she indeed knows the witness w, or by negligible chance. However, the interactive protocol is not fit for blockchain setting, as the *Prover* would need to interact with every miner to convince them.

Consequently, the proof needs to be made *non-interactive*, i.e., we should be able to simulate the *Verifier*. A transformation of the Σ-protocol into a non-interactive protocol is possible [16, 2] but out of scope for this work.

3 MimbleWimble

In this section, we introduce the MimbleWimble[2] scheme [41], and elaborate on the nature of transaction creation.

In the MimbleWimble protocol, an output is a Pedersen commitment [40] to a value v with a blinding factor r, $\text{com}(v,r)$ [41]. We denote a "source" as an unspent output, and a "destination" as a freshly generated output. Because an output is merely a commitment, it includes no knowledge of the owner and the values v and r cannot be extracted. An output can be used as a source by a user if it is unspent and the user knows its opening (i.e., its value and blinding factor).

A transaction consists of a list of sources, a list of destinations and their range proofs, an excess (a public key) and a signature of the transaction. The excess and signature are often referred to as the kernel of a transaction. The excess is defined as the difference between the sum of the source and the sum of the destination outputs[3], i.e., $excess = \frac{\prod_{j=0}^{m-1} d_j}{\prod_{i=0}^{n-1} s_i}$. An excess is a commitment to zero if and only if no value is created or destroyed in the transaction, i.e., $\text{com}(0,r)$.

The MimbleWimble blockchain supports non-interactive transaction aggregation. This means that a block can be seen as one large transaction, hence a miner can non-interactively aggregate all the transactions in the block. An aggregated transaction can be verified by verifying the kernels of the combined transactions, and validating that the sum of the excesses is the difference in sums of combined sources and destinations. Following the excess definition, an output included in both the sources and destinations will counter itself and can be excluded. A block containing the transactions $A \rightarrow B$ and $B \rightarrow C$ can therefore prune the intermediate output B; this is called a "cut-through".

Let $\pi_{\text{RP}}(d_j)$ denote the range proof [9] proving that $val(d_j) \in [0, 2^{64}-1]$, where d_j is a destination and $val(d_j)$ its value. The transaction message, will then be defined as:

$$M = (s_0, \ldots, s_{n-1}; d_0, \ldots, d_{m-1}; \pi_{\text{RP}}(d_0), \ldots, \pi_{\text{RP}}(d_{m-1})) \tag{8}$$

For the transaction to be valid, the following three properties must hold:

- Every destination must have a valid range proof.
- The sum of the transaction, defined as $\frac{\prod_{j=0}^{m-1} d_j}{\prod_{i=0}^{n-1} s_i} * excess$ must be a commitment to zero with zero blinding [41], i.e., $\text{com}(0,0)$.
- The signature of the transaction message must be valid and accepted if the *excess* is used as the public key for validation.

By requiring the signature to be verified with the excess as public-key, we require the creator(s) of the transaction to know all values and blinding factors, thereby proving ownership of the sources and knowledge of the new destinations.

Since the outputs in *MimbleWimble* have no owner in the form of an address [41], transfers between users cannot be performed in the traditional way. Let Alice and Bob be the sender and receiver respectively in a transfer performed using the MimbleWimble protocol. For a valid transaction, they need to prove that they together know all blinding factors, but none of them knows every term. For example, Alice can prove that she knows the blinding factors of the sources, while Bob proves that he knows the blinding factors of the destinations. As a result, the protocol ensures that no user can spend the funds

[2] MimbleWimble is a tongue-tying spell from the Harry Potter universe.

[3] Recall that we use multiplicative notation for commitments, subsection 2.2

of another participant by altering the transaction, and due to the excess, no user can learn information held by the other participants (e.g., Alice cannot learn the blinding factors of the destinations). Without the excess, Alice could derive the openings of Bob, as the blinding factors must add up. For example, if *Alice* has one source and two destinations, one destination being her change and the other being *Bob's,* she can derive Bob's blinding factor[4] by solving one equation with one unknown.

Recent implementations of the protocol[5], use a Schnorr multisignature [46] for signing the transaction message. The full structure of a MimbleWimble Transaction is presented in Protocol 1.

Alice$(\mathsf{g},\mathsf{h},C_a \in \mathbb{G}; v_a, v_c, v_o, r_a \in \mathbb{Z}_p)$		**Bob**$(\mathsf{g},\mathsf{h} \in \mathbb{G})$	**Public**(**utxo**)	
1 : $a, r_c \leftarrow_{\$} \mathbb{Z}_p^*$				
2 : $R_a = \mathsf{g}^a$				
3 : $C_c = \mathsf{h}^{v_c} * \mathsf{g}^{r_c}$				
4 : $x = r_c - r_a$				
5 : $X = \mathsf{g}^x$	$\xrightarrow{C_a, C_c, X, v_o}$			
		$b, r_b \leftarrow_{\$} \mathbb{Z}_p^*$		6
		$R_b = \mathsf{g}^b, P_b = \mathsf{g}^{r_b}$		7
		$C_b = \mathsf{h}^{v_o} * \mathsf{g}^{r_b}$		8
		$e = \mathscr{H}(R_a * R_b \| X * P_b)$		9
	$\xleftarrow{C_b, R_b, s_b, P_b, \pi_{\mathrm{RP}(C_b)}}$	$s_b = b + e * r_b$		10
11 : $e = \mathscr{H}(R_a * R_b \| X * P_b)$				
12 : $s_a = a + e * x$				
13 : $s = s_a + s_b, R = R_a * R_b$				
14 : $excess = X * P_b$				
	$\xrightarrow{excess, (s,R), C_a, C_b, C_c, \pi_{\mathrm{RP}(C_b)}, \pi_{\mathrm{RP}(C_c)}}$		$C_a \stackrel{?}{\in}$ **utxo**	15
			$excess \stackrel{?}{=} \frac{C_b * C_c}{C_a}$	16
			$\mathsf{g}^s \stackrel{?}{=} R * (excess)^e$	17
			$\mathrm{Verify}(\pi_{\mathrm{RP}}(C_b), C_b) \stackrel{?}{=} 1$	18
			$\mathrm{Verify}(\pi_{\mathrm{RP}}(C_c), C_c) \stackrel{?}{=} 1$	19
			remove C_a from **utxo**	20
			add C_b, C_c to **utxo**	21

Protocol 1: Building a MimbleWimble Transaction

On the one hand, this structure allows the participant to create a valid transaction. On the other hand, it also allows an attacker to distinguish multiple aggregated transactions, by brute force on every possible transaction and use of the excess for verification.To mitigate this issue, the *excess* is split into two parts, namely $excess_{new}$ and excess_offset so that the following relation is satisfied:

$$excess = excess_{new} * \mathsf{g}^{\text{excess\_offset}} \tag{9}$$

[4]I.e., the blinding factor of Bob's destination

[5]The *Grin* (`https://grin.mw/`) and *Beam* (`https://beam.mw/`) projects.

Incorporating the offset in the original protocol requires simply a modification in line 16 of Protocol 1 to validate individual transactions. However, for aggregated transactions, only the sum of the offsets will be stored, thereby making it infeasible to reconstruct and validate the individual transactions. Instead, the excess of the aggregated transaction must be validated as a whole as seen in Equation 11.

$$\text{offset} = \sum(\text{excess\_offset}) \tag{10}$$

$$\prod(excess_{new}) * g^{\text{offset}} = \frac{\prod_{d \in destinations}(d)}{\prod_{s \in sources}(s)} \tag{11}$$

With regards to non-interactive transactions, *Fuchsbauer et al.* [18] note that those can be performed by having one party of a transfer generate a full transaction, and then pass this along with the opening of one destination to the other party. The receiving party will then create a transaction with the opening and its destination as a source, and a new destination of the same value but a different blinding factor. By aggregating the two transactions and performing *cut-through* the receiver has a valid transaction, similar to the interactive scheme, with an additional kernel.

4 Lelantus

In this section we introduce *Lelantus*[6] *One-Out-Of-Many* proofs (OOOM for short)[28], a scheme that allows a prover to prove knowledge of a source without disclosing the openings *nor* the specific used source. First, we provide an overview of the system as a whole, and then describe the Lelantus proof.

The system builds on a two-coin basis, a base-coin and a shielded pool. For intuition, let us consider a fully public base currency where each coin is owned by some address and a pool of *shielded* checks. A shielded check is a double-blinded commitment $\text{com}(s,v,r)$, blinding the value v using the blinding factors r and s. To make transfers anonymous, we can *mint* a fresh check by burning owned base currency. The *One-Out-Of-Many* proofs allows us to prove that we own a check with serial number s, and spend the hidden value by publishing s *without* directly pointing at the check as a source, but merely proving that the check is within a large set of checks (anonymity set). By picking such sets to be large enough, linking a source check to a destination becomes infeasible. For example, *Lelantus* proposes anonymity sets of size 2^{16} as a practical size[7]. Storing the full set of shielded checks, as well as the used serial numbers on the Blockchain offers protection against double-spending.

The *One-Out-Of-Many* proof that is used in *Lelantus* is inspired by [24], but has some differences. For example, the *Lelantus One-Out-Of-Many* scheme proves that a zero-opening of a double-blinded commitment, i.e., $\text{com}(0,v,r)$, as opposed to an ordinary zero-commitment $\text{com}(0,r)$, is known by the *Prover*. *Lelantus* thus offers anonymity, without requiring the anonymity-set to contain checks of equal denominator[28]. Without the double-blinding, multiple pools of varying denominators were needed. Furthermore, without the anonymity sets of checks with equal denominator, it would be trivial to track unique amounts.

In terms of performance, Lelantus supports proofs with anonymity-set size 2^{16} (1567 bytes), generated in 5253 ms and verified in 947 ms. While this verification may seem slow, it accounts for a single proof; multiple proofs can be verified in batches, using 35.6 ms per proof when batching 1000 proofs. Therefore, Lelantus is suitable for anonymity-solutions without a trusted setup.

4.1 The Lelantus Proof

Here we present the One-Out-Of-Many (OOOM) proof. To begin with, the following notation is in place.

[6] A Titan in Greek mythology who was moving unseen

[7] Verification is possible in 35 ms with a size of 1.5 KB [28].

Let **clist** be a set of unspent public coins, which link an amount with an owner; i.e., if (v,P) is a set item, v is the unspent value and P is the public key that owns it. Next, let **CMList** be the set of shielded checks and **spent** the set of spent serial numbers. Note that these can be stored in a public blockchain as shared immutable truth. Equation 12 gives the content of the blockchain, where $N = |\mathbf{CMList}|$, $N_{\text{clist}} = |\mathbf{clist}|$ and $N_{\text{spent}} = |\mathbf{spent}|$. Generators shared among the peers are also included in the BC content.

$$\begin{aligned} \text{BC} = \big(\mathsf{g},\mathsf{h},\mathsf{j} \in \mathbb{G}; \mathbf{clist} \in \mathbb{Z}_p^{N_{\text{clist}}} \times \mathbb{G}^{N_{\text{clist}}}; \\ \mathbf{CMList} \in \mathbb{G}^N; \mathbf{spent} \in \mathbb{Z}_p^{N_{\text{spent}}}\big) \end{aligned} \tag{12}$$

Let the set of commitments $(C_0,\ldots,C_{N-1})$ be stored in an n-ary tree (i.e., the **CMList**), where the leaves of the tree are the commitments and m is the height of the tree. Then, $N = n^m$ is the total number of commitments within the tree. Let $C_\ell = \text{com}(0,v,r)$ be the zero commitment and $\delta_{\ell,i}$ be Kronecker's delta, where the function returns 1 if $\ell = i$ and 0 otherwise.

To prove knowledge of C_ℓ we could simply disclose ℓ, and let the *Verifier* calculate the commitment $C_v = \prod_{i=0}^{N-1} C_i^{\delta_{\ell,i}}$, which will be equal to C_ℓ. Obviously, neither ℓ *nor* C_ℓ are hidden. Following common practice within the field of Zero-Knowledge, we replace a leaking function ($\delta_{\ell,i}$) with a blinding function that relies on random noise and some challenge x. Let $a_{j,i}$ be a randomly sampled element from $\mathbb{Z}_p$, and x be the challenge provided by the *Verifier*. The blinding function $f_{j,k}$ is then defined as in Equation 13.

$$f_{j,k} = \begin{cases} \delta_{\ell_j,k} * x - a_{j,k} & \text{if } k = 0 \\ \delta_{\ell_j,k} * x + a_{j,k} & \text{otherwise} \end{cases} \tag{13}$$

Note that k is a value in $[0,n-1]$ and **not** an index in $[0,N-1]$. By using the blinding function f, C_v is updated to $C_v = \prod_{i=0}^{N-1} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}(x)}$. The exponent $\prod_{j=0}^{m-1} f_{j,i_j}(x)$, can be written as a polynomial $p_i(x)$, as in Equation 14. We note that only the polynomial $p_\ell(x)$ will be of order m, with every other polynomial being of order $m-1$, as x^m will only be included in p_i when $\ell = i$, due to the $\delta_{\ell,i}$ term.

$$\begin{aligned} p_i(x) &= \prod_{j=0}^{m-1} f_{j,i_j}(x) \\ &= \prod_{j=0}^{m-1} (\delta_{\ell_j,i_j} * x \pm a_{j,i_j}) \\ &= \prod_{j=0}^{m-1} (\delta_{\ell_j,i_j} * x) + \sum_{k=0}^{m-1} p_{i,k} * x^k \\ &= \delta_{\ell,i} * x^m + \sum_{k=0}^{m-1} p_{i,k} * x^k \end{aligned} \tag{14}$$

While everyone with possession of the $f_{j,k}$ values can evaluate the polynomials at x, the underlying polynomials are known only by the Prover. Therefore, the Prover can construct a counter-argument[8] which, when multiplied with C_v, will nullify every non m-order term of the evaluation. In order to avoid leaking C_ℓ, the Prover must add an offset to the nullifier in this fashion: $C_v *$ "counter-argument" becomes $C_\ell * \text{com}(0, v_{\text{offset}}, r_{\text{offset}})$ instead of C_ℓ. Because the Prover has knowledge of the offset, she can prove knowledge of the openings of C_ℓ.

[8] In Protocol 2, the counter-argument is $\prod_{k=0}^{m-1} (G_k * Q_k)^{-x^k}$

This scheme is a Σ-protocol (introduced earlier), which ensures that it can be made non-interactive (and thereby usable in a blockchain setting) by use of the Fiat-Shamir heuristics [16]. The relation proved by the One-Out-Of-Many proofs is expressed in Equation 15. The protocol is presented in Protocol 2 and the reader is referred to [28] for more details.

$$\begin{aligned} R = \{\, (C_0, \ldots, C_{N-1}), (\ell, v, r) \mid \forall_i : C_i \in \mathbb{G} \\ \wedge\, \ell \in \{0, \ldots, N-1\} \wedge v, r \in \mathbb{Z}_p \\ \wedge\, C_\ell = \mathrm{com}(0, v, r) \,\} \end{aligned} \tag{15}$$

Prover$(\mathsf{g}, \mathsf{h}, \mathsf{j} \in \mathbb{G}; \boldsymbol{C} \in \mathbb{G}^N; \ell, v, r \in \mathbb{Z}_p)$		**Verifier**$(\mathsf{g}, \mathsf{h}, \mathsf{j} \in \mathbb{G}; \boldsymbol{C} \in \mathbb{G}^N)$	
$r_A, r_B, r_C, r_D \leftarrow_\$ \mathbb{Z}_p^*$			
$\forall_j \in [0, \ldots, m-1]$			
$\quad a_{j,1}, \ldots, a_{j,n-1} \leftarrow_\$ \mathbb{Z}_p^*$			
$\quad a_{j,0} = -\sum_{i=1}^{n-1} a_{j,i}$			
$A = \mathrm{com}(a_{0,0}, \ldots, a_{m-1,n-1}; r_A)$			
$B = \mathrm{com}(\delta_{\ell_{0,0}}, \ldots, \delta_{\ell_{m-1,n-1}}; r_B)$			
$C = \mathrm{com}\left(\left\{a_{j,i} * (1 - 2 * \delta_{\ell_{j,i}})\right\}_{j,i=0}^{m-1,n-1}; r_C\right)$			
$D = \mathrm{com}(-a_{0,0}^2, \ldots, -a_{m-1,n-1}^2; r_D)$			
$\forall_k \in [0, \ldots, m-1]$			
$\quad \rho_k, \tau_k, \gamma_k \leftarrow_\$ \mathbb{Z}_p^*$			
$\quad G_k = \prod_{i=0}^{N-1} C_i^{p_{i,k}} * \mathsf{j}^{-\gamma_k}$			
$\quad Q_k = \mathsf{j}^{\gamma_k} * \mathrm{com}(0, \rho_k, \tau_k)$	$\xrightarrow{A, B, C, D, \{G_k, Q_k\}_{k=0}^{m-1}}$		
$\forall_j \in [0, \ldots, m-1], \forall_i \in [1, \ldots, n-1]$	$\xleftarrow{x \leftarrow_\$ \mathbb{Z}_p^*}$		
$\quad f_{j,i} = \delta_{\ell_{j,i}} * x + a_{j,i}$			
$z_A = r_B * x + r_A$			
$z_C = r_C * x + r_D$			
$z_V = v * x^m - \sum_{k=0}^{m-1} \rho_k * x^k$			
$z_R = r * x^m - \sum_{k=0}^{m-1} \tau_k * x^k$	$\xrightarrow{z_A, z_C, z_V, z_R, f_{0,1}, \ldots, f_{m-1,n-1}}$	$\forall_j : f_{j,o} = x - \sum_{i=1}^{n-1} f_{j,i}$	19
		$B^x * A \stackrel{?}{=} \mathrm{com}(f_{0,0}, \ldots, f_{m-1,n-1}; z_A)$	20
		$C^x * D \stackrel{?}{=} \mathrm{com}\left(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C\right)$	21
		$\mathrm{com}(0, z_V, z_R) \stackrel{?}{=} \prod_{i=0}^{N-1} C_i^{\prod_{j=0}^{m-1} f_{j,i_j}} * \prod_{k=0}^{m-1} (G_k * Q_k)^{-x^k}$	22

Protocol 2: *One-Out-Of-Many* proofs, relation R (Equation 15)

Within the proof detailed in Protocol 2, some design-choices might be non-obvious. First, it may not be clear why $f_{j,0}$ is not passed to the *Verifier*; recall that this is done in order to limit the data that is to be transferred by m elements, as by the definition $f_{j,0}$ can be derived from $f_{j,i}$ for $i > 0$. Furthermore, it may not be clear why G_k and Q_k exist individually, as they could simply be combined into one term and the *One-Out-Of-Many* proof would still be validated correctly. However, as $\mathrm{com}(0, \rho_k, \tau_k)$ in Q_k is

used to prove that no value is generated or lost, the separation of the terms facilitates the proof while still hiding C_ℓ through an offset γ.

5 Design of Oscausi System

In this section we present the details of our proposed system, called Oscausi. Our scheme gets inspiration and builds on top of MimbleWimble and Lelantus, on the following basis. The straightforward nature of MimbleWimble makes it hard for an attacker to gain any knowledge without participating in the network. However, if an attacker were to observe the network, they could construct the transaction graph that links sources with destinations. Lelantus addresses this issue. Contrarily to Lelantus, which uses a public and transparent base-coin, Oscausi will adopt the commitments from MimbleWimble. On top of this, our system will also support the non-interactive aggregation of transactions from MimbleWimble.

The *Oscausi* system[9] is designed on a *per output* basis and introduces primitives which can be used to move between *MimbleWimble* and *Lelantus* outputs, while still supporting the original structure of the *MimbleWimble* transaction, i.e., sources, destinations and a kernel. The two primitives introduced by Oscausi are Minting and Spending, modified from the respective operations in the Lelantus scheme. Minting will be used whenever we have a shielded destination and Spending for shielded sources. Because of these primitives JoinSplit is not directly built into the system, but is supported by combining Minting, Spending and ordinary MimbleWimble trasnactions. We present Minting, Spending and JoinSplit in Oscausi in more detail in the following subsections.

Before that, and since we are presenting a setting where also the public coin is confidential (and therefore enforced as commitments), we define the set of unspent public coins as **CList** and the blockchain content as in Equation 16.

$$\begin{aligned} \mathrm{BC} = \big(&\mathsf{g},\mathsf{h},\mathsf{j} \in \mathbb{G}; \mathbf{CList} \in \mathbb{G}^{N_{\mathrm{CList}}}; \\ &\mathbf{CMList} \in \mathbb{G}^{N}; \mathbf{spent} \in \mathbb{Z}_p^{N_{\mathrm{spent}}}\big) \end{aligned} \tag{16}$$

5.1 Minting

Minting a coin is the action of taking a coin from a base-currency, and creating a shielded version of it while destroying the original, i.e., the action of moving a coin to a shielded pool. Let $C = \mathsf{h}^v * \mathsf{j}^{r_i}$ be a *MimbleWimble* output, which a user wishes to use as a source for her transaction, and $D = \mathsf{g}^s * \mathsf{h}^v * \mathsf{j}^{r_o}$ be the double-blinded check in the pool. For the *mint* to be valid, the user must prove that *i)* she knows the openings of C i.e., she is an eligible spender, and *ii)* that the destination D does not create or destroy value. Luckily, this can be done by providing *Generalised Schnorr Proofs*. A generalised Schnorr proof proves knowledge of the solution to a double discrete logarithm problem [28], i.e., proves knowledge of the openings to a Pedersen Commitment [40]. Furthermore, as the proof is zero-knowledge, the prover may do so for a commitment $A = \mathsf{g}^s * \mathsf{j}^r$ without disclosing s or r. The relation is shown in Equation 17 and the generalised Schnorr proof is presented in Protocol 3.

$$R = \{\mathsf{g},\mathsf{j}, A \in \mathbb{G}; s, r \in \mathbb{Z}_p \mid A = \mathsf{g}^s * \mathsf{j}^r\} \tag{17}$$

For minting, in particular, two generalised Schnorr proofs are required; one should use the generators g,j (proving no hidden value) and the other the generators h,j (proving no hidden serial number). Then, a user must first prove that she knows the opening of C using the generators h,j and then prove that the

[9] `https://github.com/LHerskind/ConfidentialTransactions/tree/master/src/main/java/Lasse_Herskind`

Prover$(\mathsf{g},\mathsf{j},A\in\mathbb{G};s,r\in\mathbb{Z}_p)$			**Verifier**$(\mathsf{g},\mathsf{j},A\in\mathbb{G})$
1 :	$s_0,t_0 \leftarrow_\$ \mathbb{Z}_p^*$		
2 :	$T = \mathsf{g}^{s_0} * \mathsf{j}^{t_0} \in \mathbb{G}$	$\xrightarrow{\quad T \quad}$	
3 :	$s_1 = s_0 + s * x \in \mathbb{Z}_p$	$\xleftarrow{\quad x \leftarrow_\$ \mathbb{Z}_p^* \quad}$	
4 :	$t_1 = t_0 + r * x \in \mathbb{Z}_p$	$\xrightarrow{\quad S = \mathsf{g}^{s_1} * \mathsf{j}^{t_1} \in \mathbb{G} \quad}$	$S \stackrel{?}{=} A^x * T$

Protocol 3: Generalised Schnorr proof of the relation R as defined in Equation 17

difference between C and D is of the form $\mathsf{g}^s * \mathsf{j}^r$, i.e., **not** hiding any value. The scheme is shown in Protocol 4. Minting a shielded coin can be written as a tuple in the following way:

$$tx_{\text{mint}} = (\pi_{\text{GSP}_{\text{g,j}}}, \pi_{\text{GSP}_{\text{h,j}}}, D, C) \tag{18}$$

Because the knowledge of q of the shielded check is not used before spending, it is possible to mint

Prover$(C\in\mathbb{G};v,r_i\in\mathbb{Z}_p;\text{BC})$			**Verifier**(BC)	
1 :	$q, r_o \leftarrow_\$ \mathbb{Z}_p$			
2 :	$P = \mathsf{g}^q$			
3 :	$s = \mathscr{H}(P)$			
4 :	$D = \mathsf{g}^s * \mathsf{h}^v * \mathsf{j}^{r_o}$			
5 :	$r = r_o - r_i$			
6 :	$\pi_{\text{GSP}_{\text{h,j}}} = \text{GSP}_{\text{h,j}}((v,r_i),C)$			
7 :	$\pi_{\text{GSP}_{\text{g,j}}} = \text{GSP}_{\text{g,j}}\left((s,r),\frac{D}{C}\right)$	$\xrightarrow{\quad \pi_{\text{GSP}_{\text{h,j}}}, \pi_{\text{GSP}_{\text{g,j}}}, D, C \quad}$	$C \stackrel{?}{\in}$ **CList**	8
			$\text{Verify}\left(\pi_{\text{GSP}_{\text{h,j}}}, C\right) \stackrel{?}{=} 1$	9
			$excess = \frac{D}{C}$	10
			$\text{Verify}\left(\pi_{\text{GSP}_{\text{g,j}}}, excess\right) \stackrel{?}{=} 1$	11
			remove C from **CList**	12
			add D to **CMList**	13

Protocol 4: Minting a shielded check (double-blinded pedersen commitment) D from a ordinary Pedersen commitment C.

directly into another users account. Such can be done following the ideas presented in the Lelantus paper [28], and the followup [29], and will not be explored further in this work.

5.2 Spending

Spending a shielded check is harder than minting it. For a spending to be valid, it must satisfy three properties:

i) every shielded source is unspent, and owned by the spender;

ii) every destination is non-negative, and hides no serial number;

iii) no value is generated or destroyed in the transaction.

The first property is the hardest to satisfy. Assume that we have a list **CMlist** of the shielded commitments, and that we have the indexes and private information of the checks we are to spend (i.e., ℓ, q, v, r for each check). We remind the reader that a commitment in **CMlist** is of the form $\mathrm{com}(s,v,r)$ or else written: $\mathsf{g}^s * \mathsf{h}^v * \mathsf{j}^r$ (i.e., there are no zero-commitments). Let us assume that a Prover publishes $P = \mathsf{g}^q$ of a shielded check, allowing anyone to derive the serial number $s = \mathscr{H}(P)$. Then, the Prover derives a new set of commitments **CMlist**$'$ where $C'_i = \frac{C_i}{\mathsf{g}^s}$. In **CMlist**$'$, the commitment C'_ℓ will be a commitment to zero; the *Prover* can, therefore, use a *One-Out-Of-Many* proof to prove that she knows the opening of a commitment within the set **CMlist**$'$, without disclosing it. To protect against double-spending, every spent serial number needs to be stored and any transactions that attempt to use an already spent serial number will be rejected. The *Prover* can prove ownership of the serial number by signing with the private-key q.

By utilising range proofs for Pedersen commitments (Equation 7), the second property is satisfied. To address the third property, we need to modify the scheme to accommodate the MimbleWimble scheme. Recall from section 3 that the MimbleWimble scheme utilises a kernel to prove that no value is generated or lost. Also, the excess (a public key) of this kernel is derived from the sources and destinations of the transaction. However, Lelantus' purpose is to hide the sources; therefore including them here would defeat the purpose of the scheme. Alternatively, we utilise values from the *One-Out-Of-Many* proof to derive a "decoy"-source, which can take the place of the source. These sources can then be derived as in Equation 19.

$$S_i = \mathsf{h}^{z_{V,i}} * \mathsf{j}^{z_{R,i}} * \prod_{k=0}^{m-1} Q_{i,k}^{x^k} \tag{19}$$

The excess (Equation 20) can then be derived by using the "sources" S and the destinations C similarly to a MimbleWimble setting, with a modification to the destinations, countering the x^m exponent of the sources.

$$excess = \frac{\prod_{j=0}^{l-1} C_j^{x^m}}{\prod_{i=0}^{n-1} S_i} \tag{20}$$

Observe that only the *Prover* is capable of generating a valid signature with the *excess* as a public key. The main issue here is that we must be able to link the destination to the actual transaction, due to the x^m term being unique to the transaction. The updated spending is presented in Protocol 5.

5.3 JoinSplit

A *Joinsplit* transaction takes a positive number of shielded coins as sources, and a non-negative number of freshly generated shielded outputs as destinations. Extraction of a non-negative value from the pool is also supported.

In our scheme, JoinSplit is not implemented as an independent action (as in Lelantus), but by using Minting, Spending and the ordinary MimbleWimble transactions. We do this in order to follow the MimbleWimble structure, a set of sources and destinations and to support aggregation of transaction-primitives. In essence, the end result is similar to the JoinSplit in Lelantus.

For example, let us consider a transaction in which a user holds a shielded coin $D = \mathrm{com}(s_d, 25, r_d)$ and a MimbleWimble output $C = \mathrm{com}(0, 10, r_c)$. The user wants to create a transaction, where the output is a shielded coin $E = \mathrm{com}(s_e, 30, r_e)$, and a *MimbleWimble* output $F = \mathrm{com}(0, 5, r_f)$. To achieve this, we create three different transactions and then aggregate them to one larger transaction, as follows:

Prover$(\boldsymbol{\ell} \in [0,N-1]^n; \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{r} \in \mathbb{Z}_p^n; \hat{\boldsymbol{v}}, \hat{\boldsymbol{r}} \in \mathbb{Z}_p^l; \text{BC})$

$\forall_i \in [0,n-1]$

$P_i = \mathsf{g}^{q_i}, \quad s_i = \mathscr{H}(P_i)$

$\forall_j \in [0,N-1]$

$\text{CMList}'_j = \dfrac{\text{CMList}_j}{\mathsf{g}^{s_i}}$

$w_i = (\ell_i, v_i, r_i)$

$\pi_{\text{OOOM}_i} = \text{OOOM}(w_i, \textbf{CMList}')$

$\xrightarrow{\{A_i, B_i, C_i, D_i, \boldsymbol{Q}_i, \boldsymbol{G}_i\}_{i=0}^{n-1}}$

$\xleftarrow[x \leftarrow_\$ \{0,1\}^\lambda]{}$

$\boldsymbol{\gamma}_i = \pi_{\text{OOOM}_{i,\gamma}}$

$\forall_j \in [0,l-1]$

$C_j = \mathsf{h}^{\hat{v}_j} * \mathsf{j}^{\hat{r}_j}$

$\pi_{\text{RP}_j} = \text{RP}((\hat{v}_j, \hat{r}_j), C_j)$

$e = \sum_{j=0}^{l-1} (\hat{r}_j * x^m) - \sum_{i=0}^{n-1} \left(r_i * x^m + \sum_{k=0}^{m-1} \gamma_{i,k} * x^k \right)$

$\xrightarrow{\boldsymbol{\pi}_{\text{OOOM}}, \boldsymbol{\pi}_{\text{RP}}, \sigma_e, \boldsymbol{\sigma}_{\boldsymbol{q}}, \boldsymbol{P}, \boldsymbol{C}}$

Verifier(BC)

$\textbf{t\_spent} = \emptyset$ 12

$\forall_i \in [0,l-1]$ 13

$\text{Verify}(\pi_{\text{RP}_i}, C_i) \stackrel{?}{=} 1$ 14

$\forall_i \in [0,n-1]$ 15

$s_i = \mathscr{H}(P_i)$ 16

$s_i \notin \textbf{spent} \cup \textbf{t\_spent} \stackrel{?}{=} 1$ 17

add s_i to **t_spent** 18

$\text{Verify}(\sigma_{q_i}, P_i) \stackrel{?}{=} 1, \quad \boldsymbol{Q} = \pi_{\text{OOOM}_{i,\boldsymbol{Q}}}$ 19

$z_V = \pi_{\text{OOOM}_{i,z_V}}, \quad z_R = \pi_{\text{OOOM}_{i,z_R}}$ 20

$S_i = \mathsf{h}^{z_V} * \mathsf{j}^{z_R} * \prod_{k=0}^{m-1} Q_k^{x^k}$ 21

$\forall_j \in [0,N-1]$ 22

$\text{CMList}'_j = \dfrac{\text{CMList}_j}{\mathsf{g}^{s_i}}$ 23

$\text{Verify}(\pi_{\text{OOOM}_i}, \textbf{CMList}') \stackrel{?}{=} 1$ 24

$excess = \dfrac{\prod_{j=0}^{l-1} C_j^{x^m}}{\prod_{i=0}^{n-1} S_i}$ 25

$\text{Verify}(\sigma_e, excess) \stackrel{?}{=} 1$ 26

add **t_spent** to **spent** 27

add $\boldsymbol{C}$ to **CList**

Protocol 5: Spending multiple shielded coins in the Oscausi scheme.

Transaction 1, will be a spending of the shielded coin, D, generating a *MimbleWimble* destination $A = \text{com}(0, 25, r_a)$. Transaction 2, will use A and C as sources to an ordinary *MimbleWimble* transaction, and generate the two *MimbleWimble* outputs: $B = \text{com}(0, 30, r_b)$ and F. Transaction 3, will then mint the shielded coin E from a *MimbleWimble* coin B.

The flow of outputs can be expressed as (D) $\rightarrow$ (A), (A, C) $\rightarrow$ (B, F), (B) $\rightarrow$ (E), which can be combined as (D, A, C, B) $\rightarrow$ (A, B, F, E) and reduced to (D, C) $\rightarrow$ (E, F). Recall however that the coins going directly into, or out from, *Lelantus* cannot just be removed, as they are also included in the proof or act as coinbases; some knowledge *will* be retained. This new transaction will be valid if the individual proofs and signatures are valid, and the summed excess, i.e., the difference between (D, C) and (E, F),

is equal to the public commitments of proofs combined.Since the excess of the *Lelantus-MimbleWimble* transactions is directly linked to the proofs and outputs, it is not necessary to include an offset.

5.4 Improved Proof Verification Time

Through a series of modifications, we can use the slightly faster original *One-Out-Of-Many* proofs by *Groth* [24] instead of the *Lelantus* version (proof not included here). Recall that the simpler proof by *Groth* [24] proves knowledge of the openings to a commiment $\text{com}(0,r)$ and not to $\text{com}(0,v,r)$ as we do. We note, further, that from an algebraic point of view, $\text{com}(0,r) = \text{com}(0,0,r)$. To ensure that one element in the set will have this structure, it is not enough to counter g^s. Let $C = \text{com}(v, r_o)$ be a *MimbleWimble* output, where v is the same values as with the shielded coin, and r_o is randomly picked. To spend a shielded coin, we publish $s = \mathscr{H}(\mathrm{g}^q)$, a signature with q, and C. We then derive **CMList′** as $\text{CMList}'_j = \frac{\text{CMList}_j}{\mathrm{g}^s * C}$, thereby the coin to spend becomes $\text{com}(0,0,r-r_o)$, and the spender can prove that she knows the openings with the original *One-Out-Of-Many* proof [24]. The output C will then be the destination of the transaction and can be spent as a source in a future *MimbleWimble* transaction. This will however mean that spending would be one shielded output to one MimbleWimble output action (as opposed to many-to-many, where by many we assume ≥ 1).

6 Validation

In this section, we evaluate the proposed *Zero-Knowledge* scheme. First, we evaluate how well the design fits in with the principles of the *MimbleWimble* proposal. Secondly, we compare the anonymity properties of our scheme with well-known currencies, as Bitcoin, Monero and Zcash. Finally, we compare our proposal with the *Lelantus MimbleWimble* proposal by *Beam*[10].

MimbleWimble Support: As described in section 5, by modifying the *Lelantus* protocol we managed to support confidential outputs as a source for *minting* shielded coins which can be used for anonymous spending. Our scheme locks onto *MimbleWimble* and allows transaction-aggregation as in *"normal"* transactions, however, with the limitation that it is not possible to perform *cut-throughs* on outputs that are either a source or a destination in an anonymous transaction. While the lack of *cut-throughs* does not impact the soundness of the *MimbleWimble* system, it does impact the ability to aggressively prune the system. Without the ability to massively prune, our proposed scheme does not closely follow the ethos of *MimbleWimble*.

Anonymity properties: As shown in related works ([35, 25, 1, 17, 11, 33]) *Bitcoin* provided no anonymity beyond simply pseudonyms, and could easily be broken and analysed by an attacker. As analysed in [26], the main issue is that the transaction graph is fully public and thereby can be analysed by anyone. Furthermore, the system permits users to utilise the same addresses multiple times, increasing thus the links between transactions. Our proposed system removes entirely the notion of address, obfuscates the transaction graph and supports zero-knowledge transactions which allows no information to be leaked to the transaction graph. However, transactions require more storage and computation and may not be as accessible as *Bitcoin* due to the interactive nature of the scheme.

Because our system's anonymity is derived from Lelantus [28], it provides larger anonymity-set sizes than *Monero* (2^{16} instead of 11), with smaller proof size (1.5 KB vs. 2.1 KB [28]) and verification time (35 ms vs. 47 ms [28]).

While being a significant improvement for anonymity over *Monero*, our solution cannot compete with *Zcash* [3], which supports an anonymity set of size 2^{32}, with unbeatable size and verification time (192 bytes and 8 ms [27, 10, 23, 28]). However, our solution does **not** rely on the *trusted* setup *nor*

[10] `https://github.com/BeamMW/beam/wiki/Lelantus-MW`

"complicated and un-auditable" cryptography, enabling it to be broader and more easily adopted than *ZK-SNArKs*.

An overview of the properties and efficiency for Oscausi and other privacy solutions is visible from Table 2.

	Basecoin	Anonymity Set Size	Trusted Setup	Cryptographic Assumptions	Proof Size (KB)	Proof Time (s)	Verification Time (ms)
Monero	None	11	No	Well-tested	2.1	1	47
Zcash	Public	2^{32}	Yes	Relatively New	0.2	1-20	8
Lelantus	Public	2^{16}	No	Well-tested	1.5	5.2	35[1]
Oscausi	Confidential	2^{16}	No	Well-tested	1.5	5.2	35[2]

[1] Average cost of verifying a proof computed by batch-verifying 1000 proofs [28]
[2] Utilising the Lelantus OOOM proof with same batch-verification.

Table 2: Security properties and efficiency for privacy solutions, extracted from Lelantus [28] and extended with the type of basecoin and updated for Monero anoymity set size and Zcash Sapling upgrade.

Comparison with Beam: The team of *Beam* has looked into a *Lelantus-MimbleWimble* system[11]. While the information provided by the *Beam* team is currently limited to blog-posts and their *GitHub*, we have discussed the *Beam* scheme with *Vladislav Gelfer*[12] during the *Amsterdam ZK-Proof 2019* event. The implementation of *Beam* follows the possible improvement as described in subsection 5.4 and utilises the simpler *One-Out-Of-Many* proof by *Groth* [24] to support spending. Although this difference seems minor, it alters the system in such a manner that spending must be performed individually, i.e., every shielded coin must be extracted to a coin, and can then be combined afterwards. Our system supports spending of n shielded coins and retrieving m different coins as long as no value is generated or destroyed. This difference gives the *Beam* version a slight edge on the performance of the proofs, however, it requires additional steps to combine the extracted coins. In comparison, our system supports direct transfer to one destination. Because the destinations of a *spending* cannot be pruned, Beam requires additional storage on-chain which raises a trade-off issue between performance and required memory.

7 Conclusive Remarks

In this work, we have presented a trustless second-layer zero-knowledge scheme, which supports good anonymity through large anonymity-sets (2^{16}), while still being practical in a *Blockchain* setting, due to batch-verification and small memory footprint. Our scheme is built on Lelantus and designed to operate on top of the cryptocurrency scheme MimbleWimble, which supports confidential transactions through *homomorphic commitment schemes* and can easily be modified to support other currencies that are also based on homomorphic commitments schemes, e.g., Monero. The scheme pays for privacy through sacrificing storage, as it does not support pruning of the shielded outputs.

A number of issues remain open for further research. First, batching of the *One-Out-Of-Many* proofs is an interesting subject to explore, as multiple proofs will often occur within the same spending, and batching could lead to both performance and storage improvements. Recently, Benjamin E. Diamond presented *Many-Out-Of-Many* proofs [12] based on the original *OOOM* [24], providing a great stepping stone for the specialized *OOOM* proofs used in Lelantus and Oscausi. Next, performing *cut-throughs*

[11] `https://github.com/BeamMW/beam/wiki/Lelantus-MW`
[12] Developer at *Beam*, `https://beam.mw/team`

on shielded outputs in such a manner that it will not impact anonymity is an interesting research direction, especially within the MimbleWimble community. Last but not least, the use of the *BLS* signature scheme [6] should be explored, as the scheme could support aggregation of signatures for the *MimbleWimble* transactions, as well as the signatures proving ownership of the shielded sources in *Oscausi*.

References

[1] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *Proc. of the 17th International Conference on Financial Cryptography and Data Security (FC'13), Okinawa, Japan*, volume 7859 of *Lecture Notes in Computer Science*, pages 34–51. Springer, Berlin, Heidelberg, April 2013.

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of the 1st ACM Conference on Computer and Communications Security (CCS'93), Fairfox, Virginia, USA*, pages 62–73. ACM, December 1993.

[3] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Proc. of the 2014 IEEE Symposium on Security and Privacy (SP'14), San Jose, California, USA*, pages 459–474. IEEE, May 2014.

[4] A. Biryukov and I. Pustogarov. Bitcoin over tor isn't a good idea. In *Proc. of the 2015 IEEE Symposium on Security and Privacy (SP'15), San Jose, California, USA*, pages 122–134. IEEE, May 2015.

[5] S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath. Dandelion: Redesigning the bitcoin network for anonymity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 22, June 2017.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proc. of the 2001 International Conference on the Theoryy and Application of Cryptology and Information Security (ASIACRYPT'01), Gold Coast, Australia*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

[7] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Proc. of the 2015 IEEE Symposium on Security and Privacy (SP'15), San Jose, California, USA*, pages 104–121. IEEE, May 2015.

[8] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. `https://eprint.iacr.org/2019/191.pdf` [Online; accessed on May 20, 2020], 2019. IACR Cryptology ePrint Archive: Report 2019/191.

[9] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Proc. of the 2018 IEEE Symposium on Security and Privacy (SP'18), San Francisco, California, USA*, pages 315–334. IEEE, May 2018.

[10] M. Campanelli, D. Fiore, and A. Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. In *Proc. of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19), London, UK*, pages 2075–2092. ACM, November 2019.

[11] M. Conti, K. E. Sandeep, C. Lal, and S. Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, May 2018.

[12] B. E. Diamond. "many-out-of-many" proofs with applications to anonymous zether. `https://eprint.iacr.org/2020/293` [Online; accessed on May 20, 2020], 2020.

[13] E. Erdin, C. Zachor, and M. H. Gunes. How to find hidden users: A survey of attacks on anonymity networks. *IEEE Communications Surveys and Tutorials*, 17(4):2296–2316, July 2015.

[14] G. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):29:1–29:35, June 2018.

[15] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *Proc. of the 25th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'19), Kobe, Japan*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, Cham, December 2018.

[16] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proc. of the 1986 Conference on the Theory and Application of Cryptographic Techniques (CRYPTO'86), Santa Barbara, California, USA*, volume 263 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, August 1986.

[17] M. Fleder, M. S. Kester, and S. Pillai. Bitcoin transaction graph analysis. *CoRR*, abs/1502.01657, 2015.

[18] G. Fuchsbauer, M. Orrù, and Y. Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. In *Proc. of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'19), Darmstadt, Germany*, volume 11476 of *Lecture Notes in Computer Science*, pages 657–689. Springer, Cham, May 2019.

[19] E. Fujisaki and K. Suzuki. Traceable ring signature. In *Proc. of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC'07), Beijing, China*, volume 4450 of *Lecture Notes in Computer Science*, pages 181–200. Springer, Berlin, Heidelberg, April 2007.

[20] C. Garman, M. Green, and I. Miers. Accountable privacy for decentralized anonymous payments. In *Proc. of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Christ Chuch, Barbados*, volume 9603 of *Lecture Notes in Computer Science*, pages 81–98. Springer, Berlin, Heidelberg, February 2017.

[21] C. Garman, M. Green, I. Miers, and A. D. Rubin. Rational zero: Economic security for zerocoin with everlasting anonymity. In *Proc. of the 2014 International Conference on Financial Cryptography and Data Security (FC'14), Christ Church, Barbados*, volume 8438 of *Lecture Notes in Computer Science*, pages 140–155. Springer, Berlin, Heidelberg, March 2014.

[22] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[23] J. Groth. On the size of pairing-based non-interactive arguments. In *Proc. of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'16), Vienna, Austria*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, Cham, May 2016.

[24] J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Proc. of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'15), Sofia, Bulgaria*, Lecture Notes in Computer Science, pages 253–280. Springer, April 2015.

[25] J. Herrera-Joancomartí. Research and challenges on bitcoin anonymity. In *Proc. of the 2014 International Workshop on Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance (DPM/QASA/SETEP'14), Wroclaw, Poland*, volume 8872 of *Lecture Notes in Computer Science*, pages 3–16. Springer Verlag, September 2015.

[26] L. Herskind, P. Katsikouli, and N. Dragoni. Privacy and cryptocurrencies—a systematic literature review. *IEEE Access*, 8:54044–54059, March 2020.

[27] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. Zcash protocol specification. `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf` [Online; accessed on May 20, 2020], April 2020.

[28] A. Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptology ePrint Archive, Report 2019/373, 2019.*, 2019:373, April 2019.

[29] A. Jivanyan and S. Noether. Enabling untraceable anonymous payments in the lelantus protocol. `https://lelantus.io/enabling-untraceable-anonymous-payments.pdf` [Online; accessed on May 15, 2020], 2019.

[30] G. Kappos and A. M. Piotrowska. Extending the anonymity of zcash. *CoRR*, abs/1902.07337, 2019.

[31] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn. An empirical analysis of anonymity in zcash. In *Proc. of the 27th USENIX Security Symposium (USENIX Security'18), Baltimore, Maryland, USA*, pages 463–477. USENIX, August 2018.

[32] A. Kumar, C. Fischer, S. Tople, and P. Saxena. A traceability analysis of monero's blockchain. In *Proc. of the 22nd European Symposium on Research in Computer Security (ESORICS'17), Oslo, Norway*, volume 10493 of *Lecture Notes in Computer Science*, pages 153–173. Springer, Cham, September 2017.

[33] M. C. Kus Khalilov and A. Levi. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Communications Surveys & Tutorials*, 20(3):2543–2585, March 2018.

[34] S. Meiklejohn and R. Mercer. Möbius: Trustless tumbling for transaction privacy. *Proceedings on Privacy Enhancing Technologies*, 2018(2):105–121, February 2018.

[35] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: Characterizing payments among men with no names. *Proceedings of the 2013 ACM Sigcomm Internet Measurement Conference (IMC'13), Barcelona, Spain*, pages 127–139, October 2013.

[36] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proc. of the 2013 IEEE Symposium on Security and Privacy (SP'13), Berkeley, California, USA*, pages 397–411. IEEE, May 2013.

[37] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, and N. Christin. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 3:143–163, April 2018.

[38] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `http://www.bitcoin.org/bitcoin.pdf` [Online; accessed on May 20, 2020], 2008.

[39] S. Noether, A. Mackenzie, and T. M. Research Lab. Ring confidential transactions. *Ledger*, 1:1–18, December 2016.

[40] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of the 1991 Annual International Cryptology Conference (CRYPTO'91), Santa Barbara, California, USA*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Berlin, Heidelberg, August 1991.

[41] A. Poelstra. Mimblewimble. `https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt` [Online; accessed on May 20, 2020], July 2016.

[42] T. Ruffing and P. Moreno-Sanchez. Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin. In *Proc. of the 2017 International Conference on Financial Cryptography and Data Security (FC'17), Sliema, Malta*, volume 10323 of *Lecture Notes in Computer Science*, pages 133–154. Springer Verlag, April 2017.

[43] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *Proc. of the 19th European Symposium on Research in Computer Security (ESORICS'14), Wroclaw, Poland*, volume 8713 of *Lecture Notes in Computer Science*, pages 345–364. Springer, Cham, September 2014.

[44] T. Ruffing, P. Moreno-Sanchez, and A. Kate. P2p mixing and unlinkable bitcoin transactions. In *Proc. of the 2017 Network and Distributed Systems Security Symposium (NDSS'17), San Diego, California, USA*, pages 1–15. Internet Society, February 2017.

[45] A. Saxena, J. Misra, and A. Dhar. Increasing anonymity in bitcoin. In *Proc. of the 2014 International Conference on Financial Cryptography and Data Security (FC'14), Christ Church, Barbados*, volume 8438 of *Lecture Notes in Computer Science*, pages 122–139. Springer, Berlin, Heidelberg, March 2014.

[46] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, January 1991.

[47] N. Van Saberhagen. Cryptonote v 2.0. `https://cryptonote.org/whitepaper.pdf` [Online; accessed on May 20, 2020], 2013.

[48] D. A. Wijaya, J. Liu, R. Steinfeld, and D. Liu. Monero ring attack: Recreating zero mixin transaction effect. In *Proc. of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE'2018), New York, New York, USA*, pages 1196–1201. IEEE, August 2018.

[49] D. A. Wijaya, J. Liu, R. Steinfeld, D. Liu, and T. H. Yuen. Anonymity reduction attacks to monero. In *Proc. of the 2018 International Conference on Information Security and Cryptology (Inscrypt'18), Fuzhou, China*, volume 11449 of *Lecture Notes in Computer Science*, pages 86–100. Springer, Cham, December 2018.

[50] D. Yang, J. Gavigan, and Z. Wilcox-O'Hearn. Survey of confidentiality and privacy preserving technologies for blockchains. `https://z.cash/wp-content/uploads/static-og/static/R3_Confidentiality_and_Privacy_Report.pdf` [Online; accessed on May 20, 2020], November 2016.

[51] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle. Secure and anonymous decentralized bitcoin mixing. *Future Generation Computer Systems*, 80:448–466, March 2018.

Author Biography

Lasse Herskind was born in Aarhus, Denmark in 1996. In 2018 he received a B.S degree in software technology from the Technical University of Denmark, where he in 2019 received a M.S Degree in Human-Centered Artificial Intelligence. His research interests include blockchain technologies and cryptographic protocols, with a focus on the area of Zero-Knowledge proofs.

Panagiota (Yota) Katsikouli was born in Greece in 1987. She received the Diploma and M.S. degrees in Computer Engineering and Informatics from the Polytechnic University of Patras, Greece, in 2011 and 2013 respectively, and the Ph.D. degree in Informatics from the University of Edinburgh, Scotland, in 2017. From 2017 to 2019, she was a Post-doctoral Researcher with Inria at Lyon, France, after which she spent a short period as Post-doctoral Researcher with University College of Dublin, in Ireland. She is currently a Post-Doctoral Researcher with the Technical University of Denmark. Her research interests include distributed algorithms, blockchain technology, human mobility, smart mobility, analytics for mobile data and distributed algorithms for mobility data.

Nicola Dragoni is Professor in Secure Pervasive Computing at DTU Compute, Technical University of Denmark, Denmark, and part-time Professor in Computer Engineering at Centre for Applied Autonomous Sensor Systems, Orebro University, Sweden. He is also affiliated with the Copenhagen Center for Health Technology (CACHET) and the Nordic IoT Hub. He got a M.Sc. Degree (cum laude) and a Ph.D. in Computer Science at University of Bologna, Italy. His main research interests lie in the areas of pervasive computing and security, with focus on Internet-of-Things, Fog computing and mobile systems. He has co-authored 100+ peer-reviewed papers in international journals and conference proceedings, he has edited 3 journal special issues and 1 book. He is active in a number of national and international projects.