CLEI Electronic Journal, Volume 27, Issue 3, paper 4, August 2024

A Formal Analysis of the Mimblewimble Cryptocurrency Protocol with a Security Approach

Adrián Silveira
Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300,
adrians@fing.edu.uy

and

Gustavo Betarte
Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300,
gustun@fing.edu.uy

and

Carlos Luna
Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300,
cluna@fing.edu.uy

Abstract

A cryptocurrency is a digital currency that enables online transactions for various products and services. Cryptocurrencies are deployed over public blockchains which have the transactions replicated and dispersed across multiple nodes within a computer network. This decentralized mechanism is devised in order to achieve, for example, reliability, transparency, integrity in a network consisting of unreliable nodes. Privacy and anonymity have become crucial in this context. Privacy in blockchain refers to the ability to protect sensitive data and transactions from unauthorized access. Anonymity aims to prevent the linkage of transactions to the real identity of the parties involved. For that reason, formal and mathematical approaches are gaining popularity in order to guarantee the correctness of the cryptocurrency implementations. Mimblewimble is a privacy-oriented cryptocurrency technology which provides security and scalability properties that distinguish it from other protocols of its kind. Mimblewimble combines confidential transactions, CoinJoin and cut-through to achieve a higher level of privacy and security, as well as, scalability. In this work, we present and discuss these security properties and outline the basis of a model-driven verification approach to address the certification of the correctness of the protocol implementations. In particular, we propose an idealized model that is key in the described verification process. Then, we identify and precisely state the conditions for our model to ensure the verification of relevant security properties of Mimblewimble. In addition, we analyze the Grin and Beam implementations of Mimblewimble in their current state of development. We present detailed connections between our model and their implementations regarding the Mimblewimble structure and its security properties. Finally, we analyze the Litecoin soft-fork that enhances privacy over the blockchain based on Mimblewimble features.

Keywords: Security, Formal Verification, Mimblewimble, Idealized Model, Cryptocurrency.

1 Introduction

A cryptocurrency is a digital currency that can be exchanged online for goods and services. It can be converted into cash through a cryptocurrency exchange and vice versa. Many cryptocurrencies work using a technology called blockchain, a distributed ledger of transactions that is duplicated and distributed across the nodes of a computer network. A defining feature of cryptocurrencies is that there is no central trusted authority. The ledger is maintained using a consensus-based validation protocol where transactions are constructed in a peer-to-peer fashion and broadcast to the entire set of participants who work to validate them and build blocks. Therefore, the consensus algorithm is what decides which is the following block to be appended to the blockchain. This decentralized mechanism is devised to achieve reliability in a network consisting of unreliable nodes. Next, we present relevant security aspects of the cryptocurrencies and the importance of applying formal methods for the verification of their implementations. Then, we state the aims and the structure of this paper.

1.1 Background

Cryptocurrency protocols are a valuable target for several attacks since they deal with virtual money. Irreparable losses of money and credibility have been caused because of several attacks against cryptocurrency systems (e.g., [1]). Security and confidentiality are now much more important in this situation. For this reason, the cryptocurrency community is looking for techniques and approaches that can reduce the likelihood of successful attacks. One such approach is the application of formal methods to software implementation. In particular, interest in formally certified implementations and formal proofs has increased [2, 3].

Mimblewimble (MW) is a privacy-oriented cryptocurrency technology with scalability and security that set it apart from similar techonologies. In MW, unlike Bitcoin [4], there is no such concept as an address, and all the transactions are confidential. The method used in this work is based on formal software verification, and it aims to formally verify the fundamental mechanisms of MW and its implementations [5, 6]. Security issues are explored on an idealized model of the MW protocol. Such model delivers a realistic environment while abstracting away the specifics of any particular implementation. Then, verification should be performed on more detailed models, where low-level mechanisms are specified. Finally, it must be proved that the low-level model implements the idealized model.

Security Properties

Some security and privacy properties are crucial for the cryptocurrency protocols, however we can find them with different names in the literature. Next we classify the main security and privacy concepts that will be verified in our idealized model into the following properties.

The property of *ownership* or *security against theft* implies that only a user can spend coins belonging to them [7, 8]. In other words, it means that no one can spend a user's coins as long as they keep their keys safe. Ownership will be defined in Section 4

The property of *no-inflation* or *security against inflation* or *balance* or *No double spending* [7, 8, 9] ensures that the only way money can be created is via the supply of transactions. In other words, money can not be created from thin air. Furthermore, coins can not be double-spent. The property of no-inflation will be verified in 5.1.

The property of *transaction indistinguishability* [7] states that a transaction does not reveal anything about the amount. In other words, the amount involved in a transaction is hidden so that only the sender and the receiver know how much money is involved. In Section 5.2, we define the perfect hiding property of the Pedersen commitment to guarantee this property.

The property of *untraceable* [10] prevents tracing activities of the transactions. In other words, IP addresses should be hidden and can not be used to discover identity or geolocation of the user. In Section 5.4, we define how the broadcast of the transaction should be performed with the aim of fulfilling this property.

1.2 Related Work

The cryptocurrency community is interested in applying formal methods to guarantee correctness and properties over their protocol constructions. For instance, Idelberger et al. [11] proposes the use of defeasible logic frameworks such as Formal Contract Logic for the description of smart contracts. However, the authors did not analyze the necessary conditions the cryptocurrency protocols should satisfy to guarantee security properties.

On the other hand, Hirai [12] employs Lem to formally specify the Ethereum Virtual Machine (EVM), while Grishchenko, Maffei, and Schneidewind [13] use `F*` for the formalization of the EVM. Hildenbrandt et al. achieve the same using the reachability logic system known as $\mathbb{K}$. Pîrlea and Sergey [14] present a Coq [15, 16] formalization of a blockchain consensus protocol, verifying certain properties formally.

Boyd et al. presented a blockchain model in Tamarin [17], which is useful for analyzing certain blockchain based protocols. In addition, a mechanized formal verification of the Pedersen commitment protocol was presented by Metere and Dong [18] using EasyCrypt [19]. Moreover, an abstraction for the analysis of some security properties of MW was introduced by Fuchsbaue et al. [7].

Finally, in Betarte et al. [20] we have outlined several formal methods techniques that are particularly useful for cryptocurrency software. We have provided guidelines for adopting formal methods in cryptocurrency software projects, suggesting that set-based formal modeling (or specification), simulation, prototyping, and automated proof should be applied before considering more advanced approaches such as code formal verification. Specifically, we have presented excerpts from a set-based formal specification of a consensus protocol and the Ethereum Virtual Machine. Additionally, we have demonstrated how prototypes can be generated from these formal models and how simulations can be run on them. Lastly, we have illustrated how test cases can be generated from the same models and how automated proofs can be used to evaluate the correctness of these models. The work we present here closely follows the approach described in [20].

We believe that cryptocurrency protocols should undergo a formal security assessment in order to prove and guarantee security properties from a formal point of view. A complete formal model of the cryptocurrency is crucial. For instance, Ruffing et al. [21] show an attack against Zerocoin [22] exposing the lack of an important missing property in the formal security analysis of the cryptocurrency.

1.3 Contributions

The goal of the present work was to identify and analyze the main components of the MW protocol with the aim of building an idealized model and verifying its security properties.

The specific objectives were to i) identify and specify the foundational schemes and protocols underlying Mimblewimble, ii) identify and define the principal components of our model, iii) define and verify relevant security properties and iv) compare our model with the most popular implementations of MW, Grin and Beam, and the MW based soft-fork of Litecoin.

This work expands upon and extends the findings and insights presented in previously published papers [23, 24, 25]. In those papers, we have delineated components that constitute the foundational phases in developing a thorough formalization of the MW cryptocurrency protocol, coupled with an analysis of its properties. More specifically, we have defined the essential elements of our idealized model and articulated conditions for validity to guarantee the correctness of the blockchain. Furthermore, we have identified and precisely stated the conditions within our model to ensure the verification of relevant security properties associated with MW.

We have studied both the protocol and the essential security properties that a MW blockchain should have, along with analyzing the security properties inherent in the Pedersen commitment and range proofs schemes. The primary contribution of our work is the proposed idealized model, complemented by an analysis showcasing the essential properties it successfully validates. Furthermore, we have introduced and discussed the foundations of a model-driven verification approach aimed at certifying the correctness of a protocol's implementation. Lastly, we have conducted a comparison between two MW implementations, `Grin` [5] and `Beam` [6], with our model and we have discussed certain features that distinguish them from each other. In addition, we have have described the Litecoin soft-fork main features and compare them with our model.

The paper is organized as follows. Section 2 provides a brief description of MW. Section 3 defines the schemes and protocols our model is based on. Section 4 outlines the building blocks of a formal idealized model depicting the computational behavior of MW. Section 5 specifies the verification activities being implemented to validate both the protocol and its corresponding implementation.

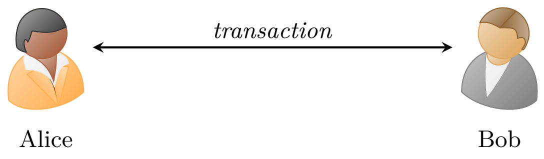
Then, Section 6 examines the `Grin` and `Beam` implementations of MW in their present stage of development and the Litecoin soft-fork based on Mimblewimble features. Final remarks are presented in Section 7 and directions for future work are detailed in Section 8.

2 The Mimblewimble protocol

In August 2016, someone using the pseudonym "Tom Elvis Jedusor" (the French name for Voldemort in Harry Potter) shared a text file on the IRC Channel outlining a cryptocurrency protocol with a distinctive approach compared to Bitcoin. This article titled 'Mimblewimble' [26] addressed some privacy concerns and explored the capability to compress the transaction history of the chain without compromising the validity of verification. In October 2016, given that this document left certain questions open, Andrew Poelstra published a paper to address these concerns [27]. He described, in more detail, the design of a blockchain based on MW.

Next, we describe the process of money transfer within the MW protocol.

Let's consider a scenario where Alice and Bob mutually decide on a money transfer. Alice intends to send v coins to Bob. They engage in off-chain communication and create the MW transaction, including the transaction amount v.



Then, the transaction should be added to a block which is distributed across the nodes of a computer network to be added to the blockchain.

In MW, transactions are conducted through Confidential transactions [28, 29]. A transaction allows a sender (Alice) to encrypt the amount v of bitcoins by using blinding factors. In a confidential transaction, only the two parties involved possess knowledge of the amount of bitcoins being exchanged. Nevertheless, for any external observer, validating the transaction's legitimacy is feasible by comparing the number of inputs and outputs. If these quantities are equal, the transaction is deemed valid. This process ensures that no funds have been generated arbitrarily, playing a key role in maintaining the integrity of the system.

In MW transactions, the recipient (Bob) randomly selects a range of blinding factors provided by the sender, which are then used as proof of ownership by the receiver.

The MW protocol aims at providing the following properties [26, 5]: i) Verification of zero sums without disclosing the specific amounts involved in a transaction, thereby ensuring confidentiality; ii) Authentication of transaction outputs without signing the transaction; and iii) Good scalability, while preserving security, by generating smaller blocks or reducing the size of old blocks, generating a blockchain with a size that doesn't increase over time as extensively as, for example, Bitcoin's. The first two properties rely on Elliptic Curves Cryptography operations and properties. The third one is a consequence of the first two.

3 Schemes and Protocols

A commitment scheme is a cryptographic primitive that allows a player in a protocol to choose a value and commit to his choice such that he can no longer change his mind. The value is kept hidden from others with the ability to reveal the committed value later. In MW transactions, the transaction amounts and blinding factors are hidden in Pedersen commitments. We say that it is hard (in terms of complexity) for someone observing the transaction to know the transaction amount or the blinding factor. In addition, since the transaction amounts are hidden, it should be possible to verify that the values are positive without revealing any information about them. Range proofs should be provided to guarantee the transactional amount lies in some range. Moreover, a transaction contains a signature to guarantee it was honestly constructed. Next, we define the schemes and protocols our model is based on.

3.1 Commitment Scheme

A commitment scheme [30] is a two-phase cryptographic protocol between two parties: a sender and a receiver. At the end of the commit phase, the sender is committed to a specific value that he cannot change later, and the receiver should have no information about the committed value. A non-interactive commitment scheme [31] can be defined as follows:

Definition 1 (Non-interactive Commitment Scheme) *A non-interactive commitment scheme $\zeta(Setup, Com)$ consists of two probabilistic polynomial time algorithms, Setup and Com, such that:*

- *Setup generates public parameters for the scheme depending on the security parameter λ.*
- *Com is the commitment algorithm: $Com : M \times R \to C$, where M is the message space, R the randomness space and C the commitment space. For a message $m \in M$, the algorithm draws uniformly at random $r \leftarrow R$ and computes the commitment $com \leftarrow Com(m, r)$.*

We have simplified the notation, but it is important to remember that Com, M, R, and C depend on the parameters generated by *Setup*.

We say the commitment algorithm is a linear function if:

$$\forall\, m_1, m_2 \in M, r_1, r_2 \in R:$$

$$Com(m_1, r_1) + Com(m_2, r_2) = Com(m_1 + m_2, r_1 + r_2)$$

In other words, Com is additive in both parameters.

Transactions in MW are derived from Confidential transactions [28], which are enabled by Pedersen commitments with homomorphic properties over elliptic curves. We define the non-interactive Pedersen commitment scheme we will use in our model, based on Definition 1, as follows:

Definition 2 (Pedersen Commitment Scheme with Elliptic Curves) *Let M and R be the finite field $\mathbb{F}_n$ and let C be the set of points determined by an elliptic curve $\mathcal{C}$ of prime order n. As in Definition 1, the probabilistic polynomial time algorithms are defined as:*

- *Setup generates the order n (dependent on the security parameter λ) and two generator points G and H on the elliptic curve $\mathcal{C}$ of prime order n whose discrete logarithms relative to each other are unknown.*
- *$Com(v, r) = r.G + v.H$, with v the transactional value and r the blinding factor chosen randomly in $\mathbb{F}_n$.*

Each MW transaction contains a list of range proofs of the transactional values. Next, we define the range proof scheme we will analyze in our model.

3.2 Range Proof Scheme

Range proofs aim at proving that a secret value is in a particular range without revealing the value. Transactions in MW contain a list of range proofs proving that the transactional values are positive and less than a specific upper bound to avoid overflow errors. We define a non-interactive zero-knowledge (NIZK) range proof scheme from a commitment scheme as follows:

Definition 3 (NIZK Range Proof Scheme) *Let $\zeta(Setup, Com)$ be a Pedersen commitment scheme as in Definition 2. Let P be the range proof space for the values $v \in M$ such that v lies in some range $[a, b]$. A non-interactive zero-knowledge range proof scheme $\eta(Prove, Verify)$ consists of two probabilistic polynomial time algorithms, Prove and Verify, such that:*

- *$Prove : M \times R \times C \to P$, which receives a value v, the random value r and the commitment $c = Com(v, r)$ and computes the range proof for the value v.*
- *$Verify : C \times P \to bool$, that given a commitment value and a range proof, decides if the value is in the range.*

Notice that the *Prove* algorithm computes a zero-knowledge proof to the commitment to verify that the committed value is in a particular range. In other words, the guarantee enables a prover to convince a verifier that the statement holds without revealing any information about the secret value. In addition, since the transactional values should be positive, the amount should lie in $[0, b]$ such that b is large enough to guarantee privacy concerns.

3.3 Schnorr Signature Protocol

The construction of the MW transaction is made off-chain by the parties. For simplicity, we shall work with a signature protocol between two parties, but this can be generalized to multi-parties.

During the transaction construction, as we will see in Section 4.2, Alice needs to verify Bob's Schnorr signature. Schnorr signature protocols can be applied over any group where discrete logarithm is hard, in our case, over an elliptic curve $\mathcal{C}$. Next, we define the Schnorr signature protocol used by them during the transaction construction. The message m can be the empty string.

Definition 4 (Schnorr Signature Protocol) *Let $\mathcal{C}$ be an elliptic curve of prime order n with generator G. Let $hash : \{0,1\}^* \to \mathbb{F}_n$ be a cryptographic hash function over the finite field $\mathbb{F}_n$. Alice secretly knows $k_A \in \mathbb{F}_n$ whose public key is $K_A = k_A.G$*

Signing

The following steps are followed to create a signature on a message $m \in \{0,1\}^*$:

1. *Alice chooses nonce $n_A \xleftarrow{\$} \mathbb{F}_n$ where $\xleftarrow{\$}$ denotes that n_A is drawn uniformly at random from $\mathbb{F}_n$.*
2. *She computes public key $N_A = n_A.G$*
3. *She computes $e = hash(N_A \mid K_A \mid m)$ and $s_A = n_A + e.k_A$ where $\mid$ denotes concatenation and N_A, K_A are represented as a bit string.*

4. *The signature σ is defined as follows:*

$$\sigma = (s_A, N_A), \textit{with public key } K_A$$

Validation
A signature $\sigma = (s_A, N_A)$ is valid if the following holds:

$$s_A.G = N_A + e.K_A$$

Each MW transaction contains a signature σ made by the parties during the transaction construction, which can be seen as a Schnorr multi-signature.

Next, a Schnorr signature protocol aggregation is defined according to our model.

Definition 5 (Schnorr Signature Protocol Aggregation) *Let $\mathcal{C}$ be an elliptic curve of prime order n with generator G. Let $hash : \{0,1\}^* \to \mathbb{F}_n$ be a cryptographic hash function over the finite field $\mathbb{F}_n$. Alice and Bob secretly know $k_A, k_B \in \mathbb{F}_n$ whose public keys are $K_A = k_A.G$ and $K_B = k_B.G$ respectively.*
Signing
The following steps are followed to create a multisignature on a message $m \in \{0,1\}^*$:

1. *Alice and Bob choose nonces $n_A, n_B \xleftarrow{\$} \mathbb{F}_n$ respectively*
2. *They compute public keys $N_A = n_A.G$ and $N_B = n_B.G$*
3. *They compute $e = hash(N_A + N_B \mid K_A + K_B \mid m)$ and respectively compute:*

$$s_A = n_A + e.k_A \qquad s_B = n_B + e.k_B$$

4. *The aggregate signature σ is defined as follows:*

$$\sigma = (s_a + s_B, N_A + N_B)$$

with the aggregate public key $K_A + K_B$

Validation
A signature $\sigma = (s_a + s_B, N_A + N_B)$ is valid if the following holds:

$$(s_A + s_B).G = N_A + N_B + e.(K_A + K_B) \tag{1}$$

Next, we show that a signature σ honestly constructed will be valid.
If we consider the signing process, we know that:
$s_A = n_A + e.k_A$ and $s_B = n_B + e.k_B$
By applying algebraic properties on elliptic curves, the left term on the equality 1 can be written as:

$$(s_A + s_B).G = (n_A + e.k_A).G + (n_B + e.k_B).G =$$

$$n_A.G + e.k_A.G + n_B.G + e.k_B.G =$$

$$n_A.G + n_B.G + e.(k_A.G + k_B.G)$$

So, if we substitute the left term on the equality 1, we have:

$$n_A.G + n_B.G + e.(k_A.G + k_B.G) = N_A + N_B + e.(K_A + K_B)$$

The above equality holds because:

$$N_A = n_A.G,\ K_A = k_A.G \text{ and } N_B = n_B.G,\ K_B = k_B.G$$

(N_A, K_A) are Alice's public keys, and (N_B, K_B) are Bob's public keys. Since we are working over the elliptic curve $\mathcal{C}$ where the discrete logarithm is hard, the only ones who know the private keys (n_A, k_A) and (n_B, k_B) are Alice and Bob respectively.

4 Idealized Model

The basic elements of our model are transactions, blocks and chains. Each node in the blockchain maintains a local state. The main components are the local copy of the chain and the set of transactions waiting to be validated and added to a new block. Moreover, each node keeps track of unspent transaction outputs (UTXOs).

Properties such as zero-sum and the absence of double spending in blocks and chains must be proved for local states. The blockchain global state can be represented as a mapping from nodes to local states. Next, we define all the elements which compose our idealized model.

4.1 Transactions

Given two fixed generator points G and H on the elliptic curve $\mathcal{C}$ of prime order n (whose discrete logarithms relative to each other are unknown), we define a single transaction as follows:

Definition 6 (Transaction) *A single transaction t is a tuple of type:*

$$Transaction \overset{\text{def}}{=} \{i : I^*,\ \ o : O^*, tk : TxKernel,$$

$$tko : KOffset\}$$

with X^* representing the lists of elements of type X and where:

- *$i = [c_1, ..., c_n]$ and $o = [o_1, ..., o_m]$ are the lists of inputs and outputs. Each input c_i and output o_i are points over the curve $\mathcal{C}$ and they are the result of computing the Pedersen commitment $r.G + v.H$ with r the blinding factor and v the transactional value in the finite field $\mathbb{F}_n$.*
- *$tk = \{rp, ke, \sigma\}$ is the transaction kernel where:*
  - *$rp = [rp_1, ..., rp_m]$ is a list of range proofs of the outputs. The $j - th$ item rp_j in rp corresponds to the $j - th$ item o_j in o*
  - *ke is the transaction excess represented by $(\sum_1^m r' - \sum_1^n r - tko).G$*
  - *σ is the kernel Schnorr signature (for simplicity, fees are left aside)*
- *$tko \in \mathbb{F}_n$ is the transaction kernel offset.*

Inputs are previous transaction outputs. The transaction kernel offset will be used to construct a block to satisfy security properties.

The ownership of a coin is given by the following definition:

Definition 7 (Ownership) *Given a transaction t, we say S owns the output o if S knows the opening (r, v) for the Pedersen commitment $o = r.G + v.H$.*

The strength of this security definition is directly related to the difficulty of solving the logarithm problem. If the elliptic curve discrete logarithm problem in $\mathcal{C}$ is hard, then given a multiple Q of G, it is computationally infeasible to find an integer r such that $Q = r.G$.

It is important to notice that the sender and the receiver do not learn their respective blinding factors during the construction of the transaction. Instead, they build a Schnorr signature that is used to guarantee the authenticity of the transaction's excess value.

We say that a transaction is valid if the following property holds:

Property 1 (Valid Transaction) *A transaction t is valid (valid_transaction(t)) if t satisfies:*

i. The range proofs of all the outputs are valid.

ii. The transaction is balanced.

iii. The kernel signature σ is valid for the excess.

These three properties have a straightforward formalization in our model. The first property we should guarantee is that all the range proofs of all the outputs are valid.

Definition 8 (Valid Range Proof Outputs Transaction) *Let $t = \{i, o, tk, tko\}$ be a transaction as in Definition 6, with transaction kernel $tk = \{rp, ke, \sigma\}$ where $o = [o_1, ..., o_m]$ is the list of outputs and $rp = [rp_1, ..., rp_m]$ is the list of the range proof outputs. Let $\eta(Prove, Verify)$ be a NIZK scheme as in Definition 3 with P the range proof space where $rp_j \in P$ proves that o_j lies in the range $[0, 2^n]$ where n is small enough to not cause overflow errors. We say all the range proof output transactions are valid if: for all $rp_j \in rp$, $Verify(o_j, rp_j) = true$.*

The list of range proof outputs provides proof that each transactional output is positive without revealing further information.

The second property is defined as follows:

Definition 9 (Balanced Transaction) *A transaction $t = \{i, o, tk, tko\}$, with transaction kernel $tk = \{rp, ke, \sigma\}$, is balanced if the following holds:*

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = ke + tko.G$$

A balanced transaction guarantees no money is created from thin air.

The kernel signature σ is a Schnorr signature aggregation with the kernel excess ke as the public key. Note that, for simplicity during the transaction construction, in Definition 5 we consider a Schnorr signature aggregation between two parties, however, once the transaction is constructed it is not necessary to know the parties involved.

The third property is defined as follows:

Definition 10 (Valid Signature for the kernel excess) *Let $t = \{i, o, tk, tko\}$ be a transaction as in Definition 6 with transaction kernel $tk = \{rp, ke, \sigma\}$ where:*

- *rp is a list of range proofs of the outputs.*
- *ke is the transaction excess.*
- *$\sigma = (s, N)$ is the kernel Schnorr signature aggregation as in Definition 5 on the empty string m.*

We say the kernel signature σ is valid with public key ke if the following holds:
$s.G = N + e.ke$ such that $e = hash(N \mid ke)$

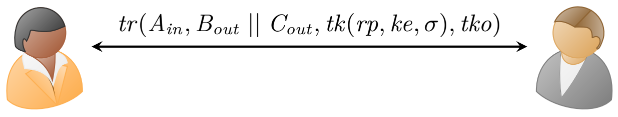
To illustrate the above definition, we detail the transaction construction between two parties.

4.2 Transaction construction

Suppose *Alice* wants to send v_B coins to *Bob*. They need to construct a transaction tr, as in Definition 6, which contains:

- Alice's Input A_{in}, such that she knows the opening (r_A, v_A) with r_A the blinding factor.
- Bob's Output B_{out} such that he knows the opening (r_B, v_B) with r_B the blinding factor and Alice's change C_{out} such that she knows the opening (r_C, v_C) with r_C the blinding factor. Let v_C be $v_A - v_B$.

The following image illustrates the target transaction tr.


$tr(A_{in}, B_{out} \mid\mid C_{out}, tk(rp, ke, \sigma), tko)$

The symbol || is the list concatenation operator.

To construct the transaction, Alice and Bob will exchange the information using a data structure called *slate*.

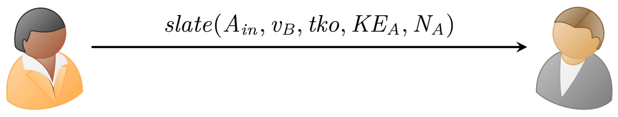
Step 1

- Alice adds A_{in} and the amount v_B to the slate.
- Alice chooses $r_C \xleftarrow{\$} \mathbb{F}_n$ (blinding factor) and computes $C_{out} = r_C.G + v_C.H$ (Definition 2). Additionally, she computes the output range proof $rp_C = Prove(v_C, r_C, C_{out})$ (Definition 3) which will be added to the transaction in the step 3.

- Alice chooses the kernel offset $tko \xleftarrow{\$} \mathbb{F}_n$ and computes Alice's kernel excess secret key as:

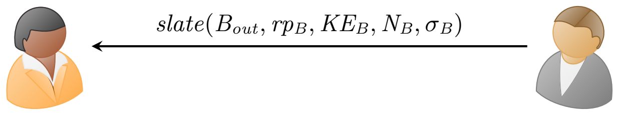
$$ke_A = r_C - r_A - tko \tag{2}$$

- Alice adds to the slate: tko and Alice's kernel excess as $KE_A = ke_A.G$
- Alice chooses nonce $n_A \xleftarrow{\$} \mathbb{F}_n$ and adds the nonce public key $N_A = n_A.G$ to the slate.
- Alice sends the slate to Bob.



Step 2

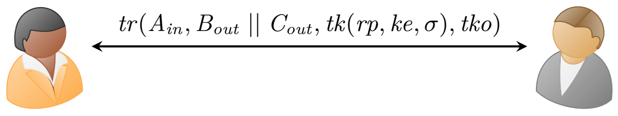
- Bob chooses $r_B \xleftarrow{\$} \mathbb{F}_n$ (blinding factor) and computes $B_{out} = r_B.G + v_B.H$ Additionally, he computes the output range proof $rp_B = Prove(v_B, r_B, B_{out})$. He adds B_{out} to the slate.
- Bob computes Bob's kernel excess as:
$$KE_B = r_B.G \tag{3}$$
and adds it to the slate.
- Bob chooses nonce $n_B \xleftarrow{\$} \mathbb{F}_n$ and adds the nonce public key $N_B = n_B.G$ to the slate.
- Bob calculates the receiver Schnorr signature on the empty string as $\sigma_B = (s_B, N_B)$ where:
$s_B = n_B + e.r_B$ such that $e = hash(N_A + N_B \mid KE_A + KE_B)$
- Bob adds $\sigma_B = (s_B, N_B)$ to the slate.
- Bob sends the slate to Alice.



Step 3

- Alice verifies Bob's signature $\sigma_B = (s_B, N_B)$ as in Definition 4:
$s_B.G = N_B + e.KE_B$ such that $e = hash(N_A + N_B \mid KE_A + KE_B)$
- Alice computes the sender Schnorr signature on the empty string as $\sigma_A = (s_A, N_A)$ where:
$s_A = n_A + e.ke_A$ such that $e = hash(N_A + N_B \mid KE_A + KE_B)$
- Alice sets the kernel excess $ke := KE_A + KE_B$.
- Alice sets the kernel signature $\sigma := (s_A + s_B, N_A + N_B)$.

Finally, Alice and Bob have computed all the remaining fields of the transaction:


$tr(A_{in}, B_{out} \mid\mid C_{out}, tk(rp, ke, \sigma), tko)$

where: $rp := [rp_A, rp_B]$; $ke := KE_A + KE_B$; and $\sigma := (s_A + s_B, N_A + N_B)$.
It is important to notice that the kernel excess ke is the same as in Definition 6 represented by $(\sum_1^m r' - \sum_1^n r - tko).G$ as shown by the following equations:

- by Equation 2, $KE_A = ke_A.G = (r_C - r_A - tko).G$
- by Equation 3, $KE_B = r_B.G$

Then, $ke = KE_A + KE_B = (r_C - r_A - tko).G + r_B.G = (r_C + r_B - r_A - tko).G$, which is the sum of the output blinding factors minus the input blinding factor minus the transaction kernel offset.

In addition, we can highlight that during the transaction construction:

- Alice does not learn Bob's blinding factor r_B.
- Bob does not learn Alice's blinding factor r_C.
- Bob does not learn Alice's change amount $v_A - v_B$.

4.3 Aggregate Transactions

A single transaction can be seen as the sending of money between multiple parties. An aggregate transaction represents many transactions.

Definition 11 (Aggregate Transaction) *An aggregate transaction tx is a tuple of type:*

$$TransacAgg \overset{\text{def}}{=} \{i : I^*,\ o : O^*, tks : TxKernel^*,$$
$$tko : KOffset\}$$

A single transaction is a particular case where the transaction kernel list contains a single element.

We say that an aggregate transaction is valid if the following property holds:

Property 2 (Valid Aggregate Transaction) *Let $tx = \{i, o, tks, ko\}$ be an aggregate transaction with $tks = [tk_1, ..., tk_t]$ the list of transaction kernels where the j-th item in tks is of the form $tk_j = \{rp_j, ke_j, \sigma_j\}$. Then tx is valid if the following are satisfied:*

i. all the range proofs rp_j are valid.

ii. the transaction is balanced.

iii. all the kernel signatures σ_j are valid for the excess ke_j.

A balanced aggregate transaction is defined as follows:

Definition 12 (Balanced Aggregate Transaction) *Let $tx = \{i, o, tks, ko\}$ be an aggregate transaction with $tks = [tk_1, ..., tk_t]$ the list of transaction kernels where the j-th item in tks is of the form $tk_j = \{rp_j, ke_j, \sigma_j\}$. We say tx is balanced if the following holds:*

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = ko.G + \sum_{ke_j \in tks} ke_j$$

Transactions can be merged non-interactively to construct an aggregate transaction. This process can be applied recursively to add more transactions into one aggregate transaction. The CoinJoin mechanism [32] makes it possible. It combines all inputs and outputs from separate transactions to form a single transaction, and the signatures can be composed by the parties. A Transaction Join can be understood as a simple way to perform CoinJoin with no composite signatures.

Definition 13 (Transaction Join) *Given a valid transaction t_0 and an aggregate transaction tx:*

$$t_0 = \{i_0, o_0, tk_0, tko_0\} \text{ and } tx = \{i, o, tks, tko\}$$

a new aggregate transaction can be constructed as:

$$tx = \{i_0 \mid\mid i, o_0 \mid\mid o, tk_0 \mid\mid tks, tko_0 + tko\}$$

The validity of the transactional parties guarantees the validity of an aggregate transaction during the construction process.

Lemma 1 (Invariant: CoinJoin Validity) *Let t_0 be a valid transaction and tx be a valid aggregate transaction. Let tx' be the result of aggregating t_0 into tx as in Definition 13. Then, tx' is valid.*

Proof.

Let $t_0 = \{i_0, o_0, tk_0, tko_0\}$ be a transaction with $tk_0 = \{rp_0, ke_0, \sigma_0\}$. Let $tx = \{i, o, tks, tko\}$ be an aggregate transaction with $tks = [tk_1, ..., tk_t]$, the list of transaction kernels where each tk_i is $tk_i = \{rp_i, ke_i, \sigma_i\}$.

Applying Definition 13, we have that the resulting tx' is of the form:

$$tx' = \{i', o', tks', ko'\}$$

$$\text{with } i' = i_0 \mid\mid i,\ o' = o_0 \mid\mid o,\ tks' = (tk_0, tk_1, ..., tk_t),$$

$$ko' = tko_0 + ko$$

According to Property 2, we need to show that:
i) *the range proofs of all the transaction outputs are valid.*
It means that, according to Definition 8, it is necessary to prove that:
for all $rp_j \in rp$, $Verify(o_j, rp_j) = true$ where $rp = [rp_0, rp_1, ..., rp_t]$
Since, t_0 and tx are valid transactions, in particular it holds that the range proofs of all the transaction outputs are valid:

- transaction t_0: $Verify(o_0, rp_0) = true$
- transaction tx: *for all* $rp_j \in rp$, $Verify(o_j, rp_j) = true$ where $rp = [rp_1, rp_1, ..., rp_t]$

ii) *the transaction tx' is balanced.*
According to Definition 12, we need to prove the following equality holds for the aggregate transaction tx':

$$\sum_{o_j \in o'} o_j - \sum_{c_j \in i'} c_j = ko'.G + \sum_{ke_j \in tks'} ke_j$$

Each term can be written as follows:

$$(\sum_{o_j \in o_0} o_j + \sum_{o_j \in o} o_j) - (\sum_{c_j \in i_0} c_j + \sum_{c_j \in i} c_j) =$$

$$(tko_0 + ko).G + ke_0 + \sum_{ke_j \in tks} ke_j$$

Rearranging the equality and using algebraic properties on elliptic curves, we have:

$$\begin{gathered}(\sum_{o_j \in o_0} o_j - \sum_{c_j \in i_0} c_j) + (\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j) = \\ (ke_0 + tko_0.G) + (ko.G + \sum_{ke_j \in tks} ke_j)\end{gathered} \tag{4}$$

Now, we apply the hypothesis concerning the validity of t_0 and tx. In particular, applying Definition 9 for t_0 and Definition 12 for tx, we have the following equalities are true:

$$\sum_{o_j \in o_0} o_j - \sum_{c_j \in i_0} c_j = ke_0 + tko_0.G \tag{5}$$

and

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = ko.G + \sum_{ke_j \in tks} ke_j \tag{6}$$

Now, if we substitute the left part of Equation 4 with the right parts of Equation 5 and Equation 6, we have:

$$(ke_0 + tko_0.G) + (ko.G + \sum_{ke_j \in tks} ke_j) =$$

$$(ke_0 + tko_0.G) + (ko.G + \sum_{ke_j \in tks} ke_j)$$

iii) *all the kernel signatures are valid for the excess.*
The list of transaction kernels of tx' is $tks = [tk_0, tk_1, ..., tk_t]$ where each tk_i is $tk_i = \{rp_i, ke_i, \sigma_i\}$.
We need to prove that, for each $i \in \{0, .., t\}$, σ_i is valid for the excess ke_i which holds trivially:

- since t_0 is a valid transaction, according to Property 1, σ_0 is valid for the excess ke_0 .
- since tx is a valid aggregate transaction, according to Property 2, for each $i \in \{1, .., t\}$, σ_i is valid for the excess ke_i.

□

Although in our model aggregate transactions and blocks are essentially the same, we are interested in distinguishing them. That is because the unconfirmed transaction pool will contain aggregate transactions and the chain will contain blocks. Since our idealized model is being built in an incremental iterative way, this distinction allows us to identify and add components in a separate way. For instance, we could add and analyze block headers to state validity conditions over the chain. On the other hand, aspects of different security properties will be analyzed on aggregate transactions and blocks.

4.4 Unconfirmed Transaction Pool

The unconfirmed transaction pool (mempool) contains the transactions which have not been confirmed in a block yet.

Definition 14 (Mempool) *A mempool mp is a list of type:*
$\mathit{Mempool} \stackrel{\text{def}}{=} \mathit{AggregateTransaction}^*$

4.5 Blocks and chain

The genesis block *Gen* is a particular block since it is the first block ever recorded in the chain. Transactions can be merged into a *block*. A block is a significant transaction with aggregated inputs, outputs, and transaction kernels.

Definition 15 (Block) *A Block b is either the genesis block Gen, or a tuple of type:*

$$\mathit{Block} \stackrel{\text{def}}{=} \{i : I^*,\ o : O^*,\ \mathit{tks} : \mathit{TxKernel}^*, \mathit{ko} : \mathit{KOffset}\}$$

where:

- $i = [c_1, ..., c_n]$ *and* $o = [o_1, ..., o_m]$ *are the lists of inputs and outputs of the transactions.*
- $\mathit{tks} = [\mathit{tk}_1, ..., \mathit{tk}_t]$ *is the list of t transaction kernels.*
- $\mathit{ko} \in \mathbb{F}_n$ *is the block kernel offset which covers all the transactions of the block.*

We can say a block is balanced if each aggregated transaction is balanced.

Definition 16 (Balanced Block) *Let* $b = \{i, o, \mathit{tks}, \mathit{ko}\}$ *be a block with* $\mathit{tks} = [\mathit{tk}_1, ..., \mathit{tk}_t]$ *the list of transaction kernels where the j-th item in tks is of the form* $\mathit{tk}_j = \{\mathit{rp}_j, \mathit{ke}_j, \sigma_j\}$*. We say the block b is balanced if the following holds:*

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = \mathit{ko}.G + \sum_{\mathit{ke}_j \in \mathit{tks}} \mathit{ke}_j$$

We assume the genesis block *Gen* is valid. We define the notion of block validity as follows:

Property 3 (Valid Block) *A block b is valid if b is the genesis block Gen or it satisfies:*

i. The block is balanced.

ii. For every transaction kernel, the range proofs of all the outputs are valid, and the kernel signature σ *is valid for the transaction excess.*

Blocks can be constructed by aggregating transactions (analogous to Definition 13). We have proved that block aggregation preserves the validity of blocks, i.e., block validity is invariant concerning block aggregation. The proof is analogous to Lemma 1 [24].

In our model, a chain is defined as a list of blocks.

Definition 17 (Chain) *A chain is a non-empty list of blocks:*

$$\mathit{Chain} \stackrel{\text{def}}{=} \mathit{Block}^*$$

For a chain c and a valid block b, we can define a predicate $\mathit{validate}(c, b)$ representing the fact that is correct to add b to c. This relation must verify, for example, that all the inputs in b are present as outputs in c; in other words, they are UTXOs.

5 Properties

Since we deal with virtual money, we should guarantee privacy and security properties on our idealized model. In particular, the property of ownership ensures that only the coins' owner can spend them. Furthermore, we should prevent an attacker from spending a coin more than once and creating virtual money from thin air. Next, we detail some relevant properties that can be verified in our model.

5.1 Protocol Properties

The property of *no coin inflation* or *zero-sum* guarantees that no new funds are produced from thin air in a valid transaction. The property can be stated as follows.

Lemma 2 (No Coin Inflation) *Let $t = \{i, o, tk, tko\}$ be a valid transaction with transaction kernel $tk = \{rp, ke, \sigma\}$. Then, the transaction excess only contains the blinding factor and the kernel offset.*

Proof.

We know the transaction t is valid, in particular, the transaction is balanced. Applying Definition 9, we know that:

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = ke + tko.G$$

Using Definition 6, we start to unfold the terms in the equality:

$$\sum_1^m r'.G + v'.H - \sum_1^n r.G + v.H =$$

$$(\sum_1^m r' - \sum_1^n r - tko).G + tko.G$$

Applying algebraic properties on elliptic curves, we have:

$$\sum_1^m v'.H - \sum_1^n v.H = (\sum_1^m r'.G - \sum_1^n r.G)$$

$$-(\sum_1^m r'.G - \sum_1^n r.G) - tko.G + tko.G = 0$$

Therefore,

$$(v'_1 + ... + v'_m).H - (v_1 + ... + v_n).H =$$

$$(v'_1 + ... + v'_m - v_1 - ... - v_n).H = 0.H = 0$$

It means that all the inputs and outputs add to zero. In other words, they summed to the commitment to the kernel offset plus the commitment to the excess blinding factor. $\square$

Thus, we have proved that no coins are being created or destroyed in the transaction. In addition, we have seen that a valid transaction guarantees that all the range proof outputs are valid, which means that every transactional output is positive.

The cut-through process is an essential feature of MW. This process aims to erase redundant outputs that are used as inputs within the same block. Let C be a list of coins that appear as an output in the block b. If the same coins appear as an input within the block, then C can be removed from the list of inputs and outputs after applying the cut-through process. The only remaining data are the block headers, transaction kernels and UTXOs. After applying cut-through to a valid block b, ensuring that the resulting block b' is still valid is essential. We can say that the validity of a block should be invariant concerning the cut-through process.

Lemma 3 (Invariant: Cut-through Block Validity) *Let $b = \{i, o, tks, ko\}$ be a block with i and o the list of inputs and outputs, $tks = [tk_1, ..., tk_t]$ the list of transaction kernels and ko the block kernel offset. Let $b' = \{i', o', tks, ko\}$ be the resulting block after applying the cut-through process to b where:*

- $i' = i \setminus (i \cap o)$
- $o' = o \setminus (i \cap o)$

Hence, if b is a valid block, then b' is valid too.

Proof.

Let $b = \{i, o, tks, ko\}$ be a block with $tks = [tk_1, ..., tk_t]$ the list of transaction kernels where the j-th item in tks is of the form $tk_j = \{rp_j, ke_j, \sigma_j\}$.

Let r be $r = i \cap o = \{r_0, r1, ..., r_n\}$ where we assume $r \neq \emptyset$ because otherwise the lemma holds trivially as $b' = b$.

Let $b' = \{i', o', tks, ko\}$ be a block with $tks = [tk_1, ..., tk_t]$, the list of transaction kernels, $i' = i \setminus r$ and $o' = o \setminus r$.

We need to prove that b' is valid. According to Property 3, we need to show that:

i) *The block b' is balanced.*

According to Definition 16, we need to prove:

$$\sum_{o_j \in o'} o_j - \sum_{c_j \in i'} c_j = ko.G + \sum_{ke_j \in tks} ke_j$$

By hypothesis, we know that b is a valid block. Applying Property 3, we know that b is balanced. According to Definition 16, the following equality holds for block b:

$$\sum_{o_j \in o} o_j - \sum_{c_j \in i} c_j = ko.G + \sum_{ke_j \in tks} ke_j$$

Applying the definition of r, we can rewrite the above equality as follows:

$$(\sum_{o_j \in o \setminus r} o_j + \sum_{o_j \in r} o_j) - (\sum_{c_j \in i \setminus r} c_j + \sum_{c_j \in r} c_j) = ko.G + \sum_{ke_j \in tks} ke_j$$

Rearranging the equality, we have:

$$(\sum_{o_j \in o \setminus r} o_j - \sum_{c_j \in i \setminus r} c_j) + (\sum_{o_j \in r} o_j - \sum_{c_j \in r} c_j) = ko.G + \sum_{ke_j \in tks} ke_j$$

Now, we can observe that we are subtracting the sum of all the elements belonging to the same set r. Thus, the term is equal to zero.

Then, if we remove the term, we have:

$$\sum_{o_j \in o \setminus r} o_j - \sum_{c_j \in i \setminus r} c_j = ko.G + \sum_{ke_j \in tks} ke_j$$

By hypothesis, we know that $i' = i \setminus r$ and $o' = o \setminus r$; therefore we can rewrite the above equality as:

$$\sum_{o_j \in o'} o_j - \sum_{c_j \in i'} c_j = ko.G + \sum_{ke_j \in tks} ke_j$$

ii) *For every transaction kernel, the range proofs of all the outputs are valid, and the kernel signature σ is valid for the transaction excess.*

Since the range proofs of the remaining outputs are the same, they remain valid.

According to definition 10, a valid signature for the kernel excess is defined in terms of the list of range proofs of the outputs rp, transaction excess ke, and the kernel signature σ. Notice that these three elements remain unchanged during the cut-through process. Since b is a valid block, it holds trivially. $\square$

5.2 Privacy and Security Properties

In blockchain systems, the notion of privacy is crucial: sensitive data should not be revealed over the network. In particular, it is desirable to ensure properties such as confidentiality, anonymity, and unlinkability of transactions. Confidentiality refers to the property of preventing other participants from knowing certain information about the transaction, such as the amounts and addresses of the owners. Anonymity refers to hiding the real identity of the parties involved in a transaction. In contrast, unlinkability refers to the inability to link different transactions of the same user within the blockchain.

In the case of MW, no addresses or public keys are used; there are only encrypted inputs and outputs. Privacy concerns rely on confidential transactions, cut-through, and CoinJoin. As we have seen, CoinJoin combines inputs and outputs from different transactions into a single aggregated transaction, and cut-through removes outputs and inputs spent within the same block. We have shown in Lemma 1 and Lemma 3 that the validity of transaction and block is guaranteed after applying both processes.

```
Game $\mathsf{G}_{\mathsf{Binding}} \overset{\text{def}}{=}$
    $(G, H, n) \leftarrow SetUp(1^\lambda)$
    $(v_1, r_1), (v_2, r_2) \leftarrow \mathcal{A}_{Binding}(G, H, n);$
    return $Com(v_1, r_1) = Com(v_2, r_2) \wedge v_1 \neq v_2$
```

Figure 1: Game Binding Commitment

The security problem of double-spending refers to spending a coin more than once. All the nodes keep track of the UTXO set, so the node checks that the inputs come from it before confirming a block to the chain. If we refer to our model, that validation is performed in the predicate *validate* mentioned in Section 4.5.

5.2.1 Security properties of Pedersen commitments

In MW transactions, input and output amounts are hidden in Pedersen commitments (Definition 1).

We state that the Pedersen commitment is expected to satisfy hiding and binding properties. The former implies that the transaction amount of coins remains private for the rest of the network over time. The latter means that senders cannot change their commitments to a different transaction amount. If that were possible, it would mean that an adversary could spend coins that have already been committed to a UTXO, which would allow the creation of coins out of thin air.

There are two possible specifications for these properties. Computational hiding or binding is when all adversaries, running in polynomial time, can break the security property with negligible probability. This asymptotic security is parameterized by a security parameter λ and adversaries run in polynomial time in λ and their other inputs. On the other hand, we talk about perfectly hiding or binding, when even with infinite computing power it would be not possible to break the security property.

Notice that a commitment scheme cannot be perfectly hiding and binding at the same time. So, for cryptocurrencies systems is better to provide stronger security in order to guarantee the hiding property. In other words, we prefer a commitment scheme with computational binding and perfectly hiding. We can understand this by first assuming adversaries break the binding property. It means that they could create money from thin air from a certain point in time but this would not affect the blockchain history. On the other hand, if the adversary breaks the hiding property, history could be inspected and all the transactions revealed which breaks one of the main principles of a privacy-oriented cryptocurrency.

Pedersen commitments are computational binding

This property relies on the discrete logarithm assumption. In provable security, security is proved to hold against any probabilistic time adversary by showing an efficient way to break the cryptography protocol implies a way to break the underlying mathematical problem which is supposed to be hard (security reduction). The adversary is modeled as a procedure.

Definition 18 (Computational Binding Commitment) *Let $\zeta(Setup, Com)$ be a Pedersen commitment scheme as in Definition 2. Let $\mathcal{A}_{Binding}$ be a polynomial probabilistic time adversary against the binding property running in the context of the game $G_{Binding}$ as in Figure 1. We say that the Pedersen commitment scheme ζ is computational binding if the success probability of $\mathcal{A}_{Binding}$ winning game $G_{Binding}$ is negligible.*

In game $G_{Binding}$, the scheme is first set up by choosing two generator points, G and H, over the elliptic curve $\mathcal{C}$ of prime order n. All these parameters are public. Secondly, the adversary $\mathcal{A}_{Binding}$ performs the attack attempting to find out two different transactional values v_1 and v_2 that commit to the same commitment. Once the adversary finishes the attack, two pair of different opening values are returned. The adversary succeeds if both pairs commit to the same value and $v_1 \neq v_2$.

As we mentioned before, the computational binding property is based on a security reduction. In terms of Pedersen commitment, it means that if the adversary $\mathcal{A}_{Binding}$ could perform the attack in the context of game $G_{Binding}$ and could win with non-negligible probability, an adversary $\mathcal{E}_{Dlog}$ attacking a game against the discrete logarithm problem on the group $\mathcal{C}$ could use $\mathcal{A}_{Binding}$ to win the game with non-negligible probability.

Recall that MW uses Pedersen commitment with elliptic curves (Definition 2). The discrete logarithm problem on this context means: given a point y over the elliptic curve $\mathcal{C}$ with generator G, it is hard to find x such that $y = x.G$. In this case, its security is shown against the discrete logarithm relation assumption which is as hard as breaking the discrete logarithm problem. It means that an adversary cannot find a non-trivial discrete logarithm relation between generators of a group independently chosen. In our case, finding a non-trivial discrete logarithm relation between G and H over the elliptic curve $\mathcal{C}$.

```
Game G_DLog $\overset{\text{def}}{=}$
    $(G, H, n) \leftarrow SetUp(1^\lambda)$
    $y \leftarrow \mathcal{E}_{Dlog}(G, H, n)$
    if $y = failure$ then
        return $failure$
    else
        return $H = y.G$
```

```
$\mathcal{E}_{Dlog}(G, H, n) \overset{\text{def}}{=}$
    $(v_1, r_1), (v_2, r_2) \leftarrow$
$(\mathcal{A}_{Binding}(G, H, n))$
    if $Com(v_1, r_1) =$
$Com(v_2, r_2)$
$\wedge\ v_1 \neq v_2$
then
        return $\dfrac{r_2 - r_1}{v_2 - v_1}$
    else
        return $failure$
```

Figure 2: Game Extractor DLog

The following lemma captures the semantics of that security reduction.

Lemma 4 (Computational Binding) *Let $\zeta(Setup, Com)$ be a Pedersen commitment scheme as in Definition 2. Let $\mathcal{A}_{Binding}$ be an adversary against the computational binding commitment (Definition 18) in the commitment scheme ζ.*

Let us assume that $\mathcal{A}_{Binding}$ succeeds in finding two distinct pair of opening values that commit to the same commitment with ϵ probability. Therefore, there exists an extractor $\mathcal{E}_{Dlog}$ which can find out a non-trivial discrete logarithm relation between the generators G and H, independently chosen, on the elliptic curve $\mathcal{C}$ with ϵ' probability using the adversary $\mathcal{A}_{Binding}$.

Hence, if ϵ is non-negligible, ϵ' is non-negligible too.

Proof.

The goal of the proof is to show how to transform the efficient adversary $\mathcal{A}_{Binding}$ that is able to break the computational binding commitment into an algorithm that efficiently solves the discrete logarithm assumption. The extractor $\mathcal{E}_{Dlog}$ will provide a simulation context in which the adversary $\mathcal{A}_{Binding}$ will perform its attack. The attack of the extractor $\mathcal{E}_{Dlog}$ will be successful if $\mathcal{A}_{Binding}$ is successful and the simulation does not fail.

According to game $G_{Binding}$, when the adversary $\mathcal{A}$ succeeds we have two identical commitments $Com(v_1, r_1) = Com(v_2, r_2)$ and $v_1 \neq v_2$ such that (Definition 2):

$$r_1.G + v_1.H = r_2.G + v_2.H$$

So we can compute:

$$H = \frac{r_1 - r_2}{v_2 - v_1}.G$$

which means that we have computed the discrete logarithm of H with respect to G.

Figure 2 shows the game G_{Dlog} which captures the semantic of the reduction. The failure event captures when the adversary $\mathcal{A}_{Binding}$ fails and therefore, the adversary $\mathcal{E}_{Dlog}$ fails too.

The probability of success of $\mathcal{E}_{Dlog}$ is equal to the probability of success of $\mathcal{A}_{Binding}$. □

Pedersen commitments are perfectly hiding

Basically, it is because, given a commitment $Com(v, r) = r.G + v.H$, there are many combinations of (v', r') that satisfies $Com(v, r) = r'.G + v'.H$. Despite the adversary have infinite computing power and could attempt all possible values, there would be no way to know which opening values (v', r') were the original ones. Furthermore, r is a random value of the finite field $\mathbb{F}_n$ so $r.G + v.H$ is a random element of $\mathcal{C}$.

Definition 19 (Perfectly Hiding Commitment) *Let $\zeta(Setup, Com)$ be a Pedersen commitment scheme as in Definition 2. Let $\mathcal{A}_{Hiding}$ be a computationally unbounded adversary against the hiding property running in the context of the game G_{Hiding} as in Figure 3. We say that the Pedersen commitment scheme ζ is perfectly binding if the success probability of $\mathcal{A}_{Hiding}$ winning game G_{Hiding} holds:*

$$Pr(b = b') = \frac{1}{2}$$

In the game described in Figure 3, first the game is set up and then the adversary chooses two distinct transactional values v_0 and v_1. Then, one of these values is randomly chosen as v_b, as well as with the blinding factor r. The commitment of (v_b, r) is computed and the adversary $\mathcal{A}_{Hiding}$ performs the attack attempting to find out which one of the values was committed.

```
Game $\mathsf{G}_{\mathsf{Hiding}} \overset{\text{def}}{=}$
    $(G, H, n) \leftarrow SetUp(1^{\lambda})$
    $(v_0, v_1) \leftarrow \mathcal{A}_{Hiding}(G, H, n)$
    $b \xleftarrow{\$} \{0, 1\}$
    $r \xleftarrow{\$} \mathbb{F}_n$
    $com = Com(v_b, r)$
    $b' \leftarrow \mathcal{A}_{Hiding}(com, G, H, n)$
    return $b = b'$
```

Figure 3: Game Hiding Commitment

5.3 Switch commitments

As already mentioned, if an attacker succeeds in breaking the computational binding property of a commitment then money can be created from thin air. Switch commitments [33] were introduced to enable the transition from *computational bindingness* to *statistical bindingness*, specially to the commitments stored in the blockchain. The notion of statistical security implies that a computationally unbounded adversary cannot violate the property except with negligible probability.

If in a certain moment we believe that the bindingness of the commitment scheme gets broken, we could make a soft fork on the chain and switch existing commitments to this new validation scheme which is backwards compatible.

Below, we show the changes that are needed for our model to also encompass Switch commitments. In Pedersen commitment definition (Definition 2) we add a third point generator J of the elliptic curve $\mathcal{C}$ whose discrete logarithm relative to G and H is unknown. We define the new commitment algorithm as follows:

$$Com(v, r) = r'.G + v.H, \textit{with } v \textit{ the transactional value and}$$
$$r' = r + hash(v.H + r.G, r.J),$$

where r is the blinding factor randomly chosen in the finite field $\mathbb{F}_n$ and J is the third point generator.

Note that r is still randomly distributed and the hash value of *ElGamal* commitment is computed which is the combination of ElGamal encryption [34] and a commitment scheme.

5.3.1 Security properties of range proofs

The goal of zero-knowledge proofs is to prove that a statement is true without revealing any information beyond the verification of the statement. In MW, we need to ensure that the amount is positive in every transaction so that users cannot create coins. Here, the hard part is to prove that without revealing the amount. In our model, the output amounts are hidden in the form of a Pedersen commitment and the transaction contains a list of range proofs of the outputs to prove that the amount is positive. This verification is performed as the first step of the validation of the transaction (Property 1).

We state that the range proof scheme (Definition 3) is expected to satisfy the properties of completeness, soundness, and zero-knowledge. Completeness states that if the statement holds for a witness v, the argument provided by the prover can convince the verifier. Soundness says that if the statement does not hold for a witness v, the prover cannot convince the verifier about the statement. Zero-knowledge states that the argument does not leak any information about the witness, except whether the statement is true or false.

In our model, a range proof scheme is expected to have perfect completeness and computational soundness with negligible error ϵ_S. We define these properties as follows:

Definition 20 (Perfect Completeness NIZK Scheme) *Let $\zeta(Setup, Com)$ be a Pedersen commitment as in Definition 2 where $c = Com(v, r)$ is the commitment to the transactional value v and the blinding factor r. Let $\eta(Prove, Verify)$ be a NIZK scheme as in Definition 3. We say $\eta(Prove, Verify)$ has perfect completeness if for every v in the range $[a, b]$, exists a range proof rp such that $rp = Prove(v, r, c)$ and $Verify(c, rp) = true$.*

Definition 21 (Computational Soundness NIZK Scheme) *Let $\zeta(Setup, Com)$ be a Pedersen commitment as in Definition 2 where $c = Com(v, r)$ is the commitment to the transactional value v and the blinding factor r. Let $\eta(Prove, Verify)$ be a NIZK scheme as in Definition 3. We say $\eta(Prove, Verify)$ has $\epsilon_S -$ soundness if for every $v \notin [a, b]$ and $rp = Prove(v, r, c)$, it holds that $Pr(\,Verify(c, rp) = true) \leq \epsilon_S$.*

Recall that, the *Prove* algorithm computes a non-interactive zero-knowledge proof to the commitment in order to verify that the committed value is in certain range. We need to ensure that the proof does not leak any information about the secret value, other than the fact it lies in a certain range.

```
Game $\mathsf{G}_{\mathsf{ZKw}} \overset{\text{def}}{=}$
    $(G, H, n) \leftarrow SetUp(1^{\lambda})$
    $(v_0, v_1) \leftarrow \mathcal{A}_{ZKw}(G, H, n)$
    $b \xleftarrow{\$} \{0, 1\}$
    $r_0 \xleftarrow{\$} \mathbb{F}_n$
    $r_1 \xleftarrow{\$} \mathbb{F}_n$
    $com_0 = Com(v_0, r_0)$
    $com_1 = Com(v_1, r_1)$
    $rp = Prove(v_b, r_b, com_b)$
    $b' \leftarrow \mathcal{A}_{ZKw}(rp, com_0, com_1, G, H, n)$
    return $b = b'$
```

Figure 4: Game Zero-knowledge range proof

Definition 22 (Perfect ZK Range Proof) *Let $\zeta(Setup, Com)$ be a Pedersen commitment as in Definition 2, where $c = Com(v, r)$ is the commitment to the transactional value v and the blinding factor r. Let $\eta(Prove, Verify)$ be a NIZK scheme as in Definition 3 where rp is a range proof such that $rp = Prove(v, r, c)$. Let $\mathcal{A}_{ZKw}$ be a computationally unbounded adversary running in the context of the game G_{ZKw} as in Figure 4. We say that the range proof rp is perfect zero-knowledge if the success probability of $\mathcal{A}_{ZKw}$ winning game G_{ZKw} is $Pr(b = b') = 0.5$. In other words, the adversary cannot learn anything about the opening value v from rp.*

In Figure 4, first the game is set up and then the adversary chooses two distinct transactional values v_0 and v_1. Then, one of these values is randomly chosen as v_b, as well as with the blinding factors r_0 and r_1. Both commitments of (v_0, r_0) and (v_1, r_1) are computed. Then, a zero-knowledge proof rp is generated for the value v_b. Finally, the adversary $\mathcal{A}_{ZKw}$ performs the attack attempting to find out which value $\{v_0, v_1\}$ corresponds to the range proof rp; in other words, for which value holds $Verify(com_b, rp) = true$.

Bulletproofs [31] is a non-interactive zero-knowledge proof protocol. A bulletproof is a short proof (logarithmic in the witness size) with the aim of proving that the committed value is in a certain range without reveling it. Proof generation and verification times are linear in the length of the range. Moreover, aggregation of range proofs is supported which enables the parties to generate a single proof, without revealing their inputs to each other. They do not require a trusted setup and are non-interactive applying the Fiat-Shamir heuristic [35] [36] which replaces the verifier's random challenge by a hash. MW uses Bulletproofs, which helps to maintain its size more compact than other Bitcoin blockchains as it uses a compact version of the inner product argument (*rp* in Definition 3). Bulletproofs are computational soundness and rely on the discrete logarithm assumption. In this case, its security is shown against the discrete logarithm relation assumption which means that an adversary cannot find a non-trivial discrete logarithm relation between generators of a group independently chosen. In our model, it means to be proved over the elliptic curve $\mathcal{C}$ and it could be finding a non-trivial discrete logarithm relation between G and H (Definition 6). Solving this is as hard as breaking the discrete logarithm problem. We could say that the discrete logarithm relation problem can be reduced to the computational soundness security property. In [25] we have stated a lemma that captures that semantic of the computational soundness property.

So far we have analyzed that Pedersen commitments are computational binding and perfectly hiding. On the other hand, Bulletproofs are perfect zero-knowledge and computational soundness.

Let us suppose we have a commitment c such that $c = Com(r, v) = r.G + v.H$ and the *Prover* computes the range proof *rp* for the value v as in the range proof scheme Definition 3. Now, we need to pass c and *rp* to the *Verifier* in order to verify whether *rp* the argument is valid. Since we know the property of perfect completeness holds, the *Verifier* will accept *rp*. In other words, an honest *Prover* succeed in convincing the *Verifier*. Moreover, since the commitment is perfectly hiding we have not revealed any information about the commitment by passing it to the *Verifier*.

5.4 Unlinkability and Untraceability

In our model, each node has a pool of unconfirmed transactions in the *mempool.* These transactions are waiting for the miners to be included in a block. We can distinguish two security properties of the transactions. Untraceability refers to the transactions in the mempool and unlinkability to the transactions in the block. In our model, these two notions are formalized as follows. The Transaction Untraceability property states that for every transaction in the mempool, it is impossible to relate the transaction to the IP address of the node that originated it. The Transaction Unlinkability property states that given a valid block b, it is computationally infeasible to know which input cancels which output. Both properties are defined in [24]. In particular, for the unlinkability property, we have proved the following lemma. Moreover, the operations cut-through and CoinJoin, which were described above, also contribute to this property.

Lemma 5 (Transaction Unlinkability) *For any valid block b and for any polynomial probabilistic time adversary $\mathcal{A}$, the probability of $\mathcal{A}$ in finding a balanced transaction within b is negligible.*

Proof.

Let $b = \{i, o, tks, ko\}$ be a valid block with $tks = (tk_1, ..., tk_t)$ the list of transaction kernels. The j-th item in tks is of the form $tk_j = \{rp_j, ke_j, \sigma_j\}$.

The goal of the adversary $\mathcal{A}$ is to find a tuple of the form $\{i', o', ke'\}$ where the list of inputs i' is a subset of i and the list of outputs o' is a subset of o, satisfying Definition 9 of a balanced transaction. It means that, the following equality must be true for the tuple:

$$\sum_{o_j \in o'} o_j - \sum_{c_j \in i'} c_j = ke' + tko'.G$$

where ke' is the transaction excess and tko' the transaction kernel offset.

If we refer to the construction process the transaction kernel offsets were added to generate a single aggregate offset ko to cover all transactions in the block. It means that we do not store the individual kernel offset tko' of the transaction in b once the transaction is aggregated to the block.

The challenge is trying to solve the adversary $\mathcal{A}$ could be seen as the subset sum problem (NP-complete) but, in this case, tko' is unrecoverable. So, although many transactions have few inputs and outputs, it is computationally infeasible, without knowing that value, to find the tuple. □

Regarding the untraceability property, we should refer mainly, to the broadcast of the transactions. Once the transactions are created, they are broadcasted to the network and they are included in the mempool. Each node could track the IP address from the node which received the transaction. At that point nodes could record the transactions, allowing them to build a transaction graph.

We define that the broadcast of a transaction should be performed with confusion as a way to obscure the IP address node.

Definition 23 (Broadcast with confusion) *Let's say node A sends a transaction to node B. We say B receives the transaction with confusion if given the IP address of node A, the node B does not know if the transaction was originated by the node A or not.*

In other words, it can be said that if some malicious nodes, working together, construct a graph of the pairs $(transaction, IP\ address\ node)$, the IP address node will not convey information about what node originated the transaction. Therefore, in our model, we require this property to hold before the broadcast takes place. In order to achieve this, we can establish that the node broadcasting the transaction should be far enough from the one which originated it. Moreover, CoinJoin could be performed before the broadcast.

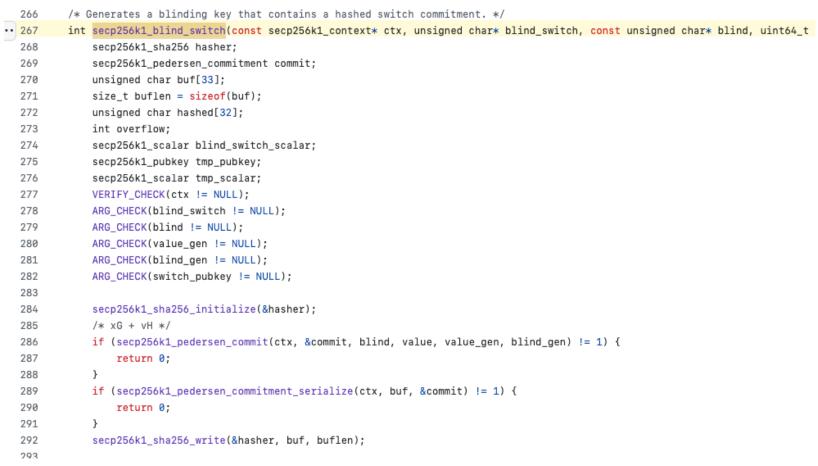
Dandelion, proposed by Bojja et al. [37], is a protocol for transaction broadcasting intended to resist that deanonymization attack. Dandelion is not part of the MW protocol, however this kind of protocols should be implemented by each node to lower the risk of creating the transaction graph. In Dandelion, broadcasting is performed in two phases: the *stem* phase and the *fluff* phase. In the *stem* phase the transaction is broadcasted randomly to one node, which then randomly sends it to another, and so on. This process finishes when the *fluff* phase is reached, and the transaction is broadcasted to the network using a gossip protocol. In [24] we have defined the routines which capture the semantic of the phases.

6 Implementations

Because of the its robust security, privacy and scalability, there are several implementations of Mimblewimble. In 2019, the first two practical implementations were launched: `Grin` and `Beam`. Although, there are some design and technical differences in both projects, they implement and extend the core of the MW protocol. Next, we first describe the main features of their design and compare them with our model. Then, we mention some features that set apart `Grin` from `Beam`. On the other hand, a privacy enhancing approach can be deployed as an upgrade to existing blockchains such as Litecoin. Litecoin, created in October 2011, is one of the earliest alternative coins after Bitcoin. In November 2019, in order to achive higher privacy, the Litecoin network made a soft-fork based on Mimblewimble features. Finally, we describe the Litecoin soft-fork main features and compare them with our model.

6.1 Grin

`Grin` [5] is an open source software project with a simple approach to MW. As we will see below, its design is a straightforward interpretation of our model.



```
/* Generates a blinding key that contains a hashed switch commitment. */
int secp256k1_blind_switch(const secp256k1_context* ctx, unsigned char* blind_switch, const unsigned char* blind, uint64_t
    secp256k1_sha256 hasher;
    secp256k1_pedersen_commitment commit;
    unsigned char buf[33];
    size_t buflen = sizeof(buf);
    unsigned char hashed[32];
    int overflow;
    secp256k1_scalar blind_switch_scalar;
    secp256k1_pubkey tmp_pubkey;
    secp256k1_scalar tmp_scalar;
    VERIFY_CHECK(ctx != NULL);
    ARG_CHECK(blind_switch != NULL);
    ARG_CHECK(blind != NULL);
    ARG_CHECK(value_gen != NULL);
    ARG_CHECK(blind_gen != NULL);
    ARG_CHECK(switch_pubkey != NULL);

    secp256k1_sha256_initialize(&hasher);
    /* xG + vH */
    if (secp256k1_pedersen_commit(ctx, &commit, blind, value, value_gen, blind_gen) != 1) {
        return 0;
    }
    if (secp256k1_pedersen_commitment_serialize(ctx, buf, &commit) != 1) {
        return 0;
    }
    secp256k1_sha256_write(&hasher, buf, buflen);
```

Figure 5: `Grin Switch commitment` source code [39]

Blocks and Transactions

`Grin` transactions are based on confidential transactions. In [24] we have described how each transaction contains a list of inputs and outputs. Each input and output is in the form of a Pedersen commitment. For instance, in the input structure there is a field that stores the commitment pointing to the output being spent.

In addition, the transaction structure has a list of transaction kernels (of type *TxKernel*) with the transaction excess and the kernel signature. All this data has a straightforward relation to our definition of transaction (Definition 6).

Moreover, a `Grin` transaction also includes the block number at which the transaction becomes valid. We have not added this data to the transaction structure yet and we also should include it in the signature process. In `Grin`, not only the transaction fee is signed, the signing process also takes into account the absolute position of the blocks in the chain. In this way, if a kernel block points to a height greater than the current one, it is rejected. If the relative position points to a specific kernel commitment, `Grin` has the same behavior.

`Grin` Blocks also stores a kernel offset which is the sum of all the transaction kernel offsets added to the block. In our model, the kernel offset is defined within a block and the notion of adding a transaction into a block is formalized on the block aggregation.

Privacy and Security Properties

The cut-through process, as explained in Section 5.1, provides scalability and further anonymity. `Grin` performs this process in the transaction pool, which we formalized as *mempool* (Definition 14). Outputs which have already been spent as new inputs are removed from the mempool, using the fact that every transaction in a block should sum to zero.

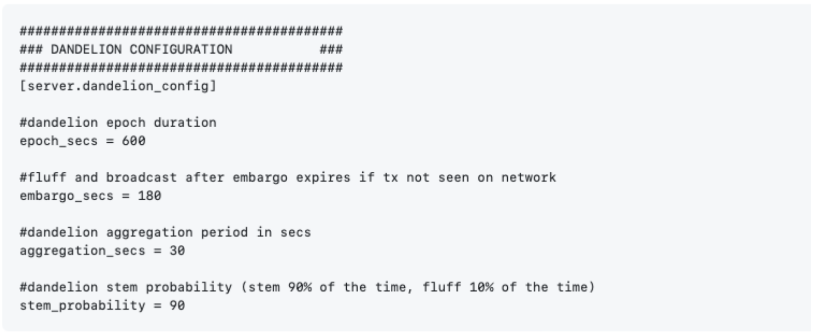
CoinJoin, as we have mentioned, combines inputs and outputs from multiple transactions into a single transaction in order to obfuscate them. In `Grin`, every block is a CoinJoin of all other transactions in the block.

As highlighted in Section 5.2, Switch commitments offer perfect hiddenness and statistical bindingness.

`Grin` incorporates a switch commitment, as described in [38], within a transaction output to enhance security beyond computational bindingness (Definition 18). Pedersen commitments are employed in the Confidential Transaction, where the blinding factor r' is not solely chosen randomly but is calculated by adding the hash of an ElGamal commitment to another random factor r as we have defined in Section 5.3. The operation that implements it can be seen in Figure 5 and it is crucial in defense against quantum adversaries.

6.2 Beam

`Beam` [40] was the other Mimblewimble project launched on January 2019. This open source system has a founding model and a dedicated development team.



```
########################################
### DANDELION CONFIGURATION          ###
########################################
[server.dandelion_config]

#dandelion epoch duration
epoch_secs = 600

#fluff and broadcast after embargo expires if tx not seen on network
embargo_secs = 180

#dandelion aggregation period in secs
aggregation_secs = 30

#dandelion stem probability (stem 90% of the time, fluff 10% of the time)
stem_probability = 90
```

Figure 6: `Grin Dandelion` configuration file [42]

Blocks and Transactions

`Beam` transactions are confidential transactions implemented by the Pedersen commitment scheme. This follows the same approach as our model.

In Section 4 we have described how each node maintains a local state. The state keeps track of the UTXO set. `Beam` extends the behavior of that set, supporting the incubation period on a UTXO. This means that `Beam` sets the minimum number of blocks created after the UTXO entered the blockchain, before it can be spent in a transaction. This number is included in the transaction signature. `Beam`'s output stores the number of blocks corresponding to the incubation period [24]. If we relate this functionality to our model, we should check that every output with certain incubation period on a block was 'lawfully' spent for the entire blockchain (global state). In other words, if we have an output transaction o with an incubation period d on a confirmed block b over the chain and a later confirmed block b' containing o as an input, then b' should be, at least, d blocks away from b on the blockchain.

Privacy and Security Properties

`Beam` supports cut-through as we described above. In addition, `Beam` adds a scalable feature to eliminate all intermediate transaction kernels [41] in order to keep the blockchain as compact as possible. It would be important to prove that the resulting transaction is still valid in Property 1.

6.3 Discussion

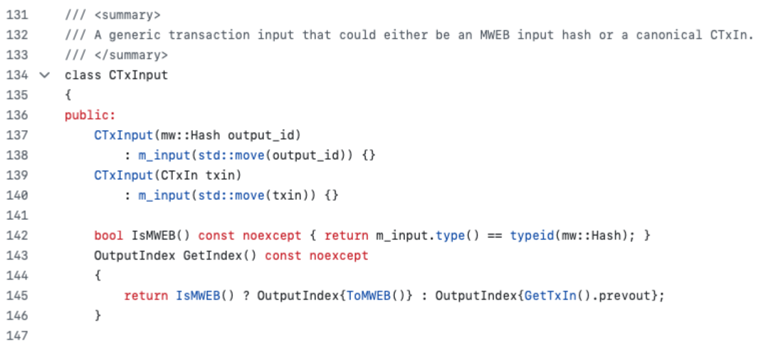
Both `Grin` and `Beam` implementations address the main features of the MW protocol, namely the properties of confidentiality, anonymity and unlinkability comprised in our work.

Broadcasting Protocol

Both `Grin` and `Beam` use the Dandelion scheme as broadcasting protocol [37]. We have formalized that a broadcasting protocol should hold the property of Transaction Untraceability. It should not be possible to link transactions and their originating IP addresses, in other words, to deanonymize users. Broadcast with confusion, as we describe in Property 23, should be carried out to satisfy Transaction Untraceability. We have also described the stem and fluff phases of the Dandelion scheme.

Grin implements a simplified version of the Dandelion++ protocol. Each individual node pseudorandomly determines whether it functions as a *stem* or a *fluff* node (Section 5.4) at regular intervals known as epoch periods. As we can see in Figure 6 this is configurable via `epoch_secs` and it is set to last for 10 minutes. At the start of an epoch, the node randomly selects a single connected peer to serve as its outbound relay and decides in which mode to operate. This decision is randomized, with the probability of selecting *stem* mode set at 0.9 (parameter `stem_probability`, Figure 6). As we have defined in Section 4.4, any transaction received from inbound peers or transaction originated from the node itself are first added to the `mempool`. If the node is in *stem* mode, after being added to the `mempool`, received stem transactions are forwarded onto the their relay node as a *stem* transaction. If the node is in *fluff* mode, then transactions received from inbound nodes are kept in the `mempool`. The system verifies if any transactions are older than 30 seconds (configurable as `aggregation_secs`, Figure 6). If found, these transactions are aggregated and subsequently fluffed. This aggregation is performed by applying Transaction Join (Definition 13, Section 4.3).

Furthermore, `Grin`'s implementation, in the *stem* phase, allows for cut-through, which provides greater anonymity to the transactions before they are broadcasted to the entire network.



```
131     /// <summary>
132     /// A generic transaction input that could either be an MWEB input hash or a canonical CTxIn.
133     /// </summary>
134     class CTxInput
135     {
136     public:
137         CTxInput(mw::Hash output_id)
138             : m_input(std::move(output_id)) {}
139         CTxInput(CTxIn txin)
140             : m_input(std::move(txin)) {}
141
142         bool IsMWEB() const noexcept { return m_input.type() == typeid(mw::Hash); }
143         OutputIndex GetIndex() const noexcept
144         {
145             return IsMWEB() ? OutputIndex{ToMWEB()} : OutputIndex{GetTxIn().prevout};
146         }
147
```

Figure 7: `Litecoin MWEB` transaction body source code [45]

In addition, in order to improve privacy, `Beam`'s implementation adds dummy transaction outputs at the stem phase. Each output has a value of zero and it is indistinguishable from regular outputs. Later, after a random number of blocks, the UTXOs are added as inputs to new transactions, i.e., they are spent and removed from the blockchain.

In [24] can be seen how we have extended the stem routine to capture `Beam`'s behavior.

Range proofs

`Grin` and `Beam` implement range proofs using Bulletproofs. Regarding our model, the validity of the range proofs are the first property a transaction should satisfy to be valid (Property 1). Furthermore, for every transaction in a bock, the range proofs of all the outputs should be valid too (Property 3).

6.4 Litecoin

Litecoin [43] is an opensource project [44] and it was born from a copy of Bitcoin's source code. Litecoin features faster transaction confirmation times since it has a 2.5-minute block processing time on average. Meanwhile, Bitcoin's block processing time is 10 minutes on average. In Litecoin transactions, the sender and recipient addresses as well as the amount are public. It means that transaction history can be publicly traced. In November 2019, MWEB (Mimblewimble Extension Blocks) were introduced as a Litecoin improvement proposal. They are a way to offer Confidential Transactions on the Litecoin network via soft-fork.

Blocks and Transactions

At the core, MWEB combines Mimblewimble features over extension blocks. The extension blocks create a side-chain alongside Litecoin regular blocks. Users can decide whether to have their transactions public or private via MWEB. Both private and public transactions exist simultaneously on Litecoin. In Figure 7 (line 142) can be seen as a generic transaction input could either be an MWEB input or a canonical input.

In order to provide privacy and confidentiality guarantees, MWEB transactions are based on confidential transactions. In Figure 8, we can observe that each transaction contains a list of inputs and outputs.

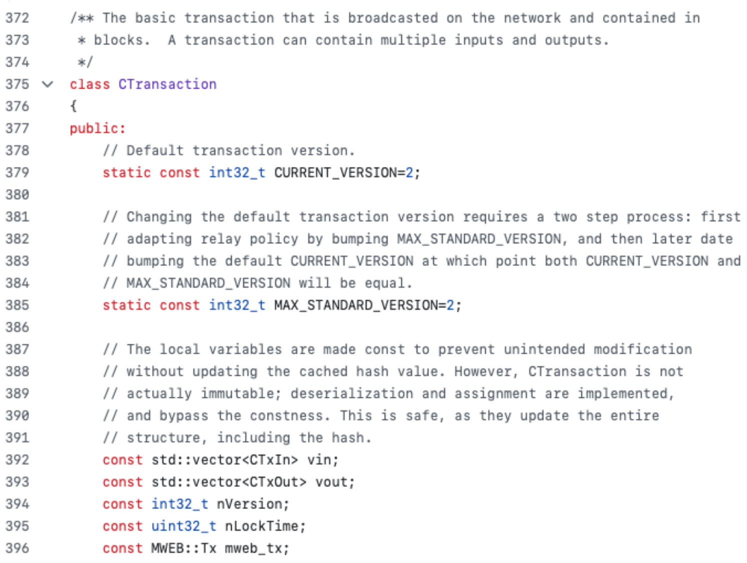
As can be seen in Figure 9 each input transaction refers to the previous transaction's output as we have shown in our model where inputs are previous transaction outputs.

Privacy and Security Properties

The implementation of Mimblewimble over the soft-fork of Litecoin enhances privacy providing users the option of sending confidential Litecoin transactions. The amount is only known between the parties and addresses are kept private. In MWEB, CoinJoin can be performed to combine multiple inputs from different parties into a single transaction. On the other hand, when a new node joins the Litecoin network for the first time, it must verify the entire history of the blockchain. In MWEB, the cut-through process can be performed in order to verify the compact blockchain.

7 Conclusions

MW represents a significant advancement in safeguarding anonymity and privacy within the realm of cryptocurrencies. Due to its facilitation of traceability and the validation process, both `Grin` and `Beam` have incorporated the MW protocol into their implementations.

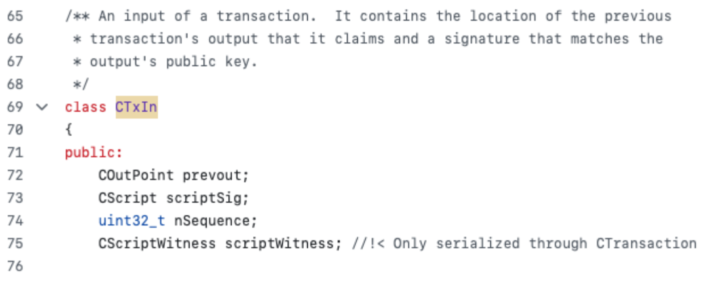


```
/** The basic transaction that is broadcasted on the network and contained in
 * blocks.  A transaction can contain multiple inputs and outputs.
 */
class CTransaction
{
public:
    // Default transaction version.
    static const int32_t CURRENT_VERSION=2;

    // Changing the default transaction version requires a two step process: first
    // adapting relay policy by bumping MAX_STANDARD_VERSION, and then later date
    // bumping the default CURRENT_VERSION at which point both CURRENT_VERSION and
    // MAX_STANDARD_VERSION will be equal.
    static const int32_t MAX_STANDARD_VERSION=2;

    // The local variables are made const to prevent unintended modification
    // without updating the cached hash value. However, CTransaction is not
    // actually immutable; deserialization and assignment are implemented,
    // and bypass the constness. This is safe, as they update the entire
    // structure, including the hash.
    const std::vector<CTxIn> vin;
    const std::vector<CTxOut> vout;
    const int32_t nVersion;
    const uint32_t nLockTime;
    const MWEB::Tx mweb_tx;
```

Figure 8: `Litecoin MWEB` transaction body source code [45]



```
/** An input of a transaction.  It contains the location of the previous
 * transaction's output that it claims and a signature that matches the
 * output's public key.
 */
class CTxIn
{
public:
    COutPoint prevout;
    CScript scriptSig;
    uint32_t nSequence;
    CScriptWitness scriptWitness; //!< Only serialized through CTransaction

```

Figure 9: `Litecoin MWEB` transaction body source code [45]

We have emphasized components that form crucial steps in the comprehensive formalization of the MW cryptocurrency protocol, the analysis of its properties, and the verification of its implementations. The proposed idealized model plays a pivotal role in the described verification process.

Initially, we have established the primary components of our idealized model, encompassing transactions, blocks, and the chain. Then, we have provided validity conditions to ensure the correctness of the blockchain. We have stated precise conditions for both a valid transaction and a valid block. Moreover, we have defined and demonstrated that the validity of a block remains invariant concerning the cut-through process and CoinJoin.

The main difficulty encountered during this process stemmed from the absence of "official" documentation. Consequently, we have conducted a thorough literature review to comprehensively analyze and conceptualize the key components of MW. Furthermore, despite the availability of online documentation for Grin and Beam, the primary challenge was to construct a model that abstracts away the specifics of their implementations.

Additionally, we have pinpointed and clearly articulated the conditions within our model to guarantee the verification of pertinent security properties of MW, signifying a significant contribution of this work. Firstly, we have demonstrated that a valid transaction does not generate new funds out of thin air. Secondly, given that MW transactions are structured as Pedersen commitments, we have examined the robustness of the scheme concerning the essential security properties that a cryptocurrency protocol must possess. Specifically, we have outlined the computational binding commitment property and we have demonstrated a security reduction through a game-based cryptographic proof approach. Thirdly, we have delved into zero-knowledge proofs to establish that the transaction output is positive without disclosing the specific amount. Additionally, we have described certain security properties that a range proof scheme should meet.

Then, we have defined and analyzed two crucial security properties: unlinkability and untraceability. In particular, we have provided a proof demonstrating that the likelihood of discovering a balanced transaction within a valid block is negligible. Furthermore, we have defined the concept of broadcast with confusion to obfuscate the IP address from which the transaction originates.

Finally, we have conducted an analysis and comparison of the current states of development of the `Grin` and `Beam` implementations, using our model and its properties as a reference basis. We have illustrated comprehensive connections between our model and their implementations concerning the MW structure and its security properties. Specifically, we have expanded our *steam* routine abstraction to encompass Beam's behavior, incorporating dummy transactions and an incubation period to enhance privacy. On the other hand, we have analyzed the MW based soft-fork Litecoin and we have compared with our model.

The primary challenge encountered in addressing the comparison between our model and the implementations was the need to read the source code of `Grin` and `Beam`. This was essential to identify the components of our model and analyze how they are implemented in the respective codes.

The present work is not without limitations. Firstly, the off-chain nature of transaction construction between parties necessitates a thorough analysis of the protocol construction to ensure security and privacy properties during communication. Additionally, some of the security properties rely on an underlying hard problem in terms of provable security. On the other hand, the consensus protocol plays a crucial role in ensuring the integrity of the recorded information. To validate blocks and incorporate them into the chain, the network must reach consensus through an agreed-upon algorithm. Two commonly used algorithms are *Proof of Work* and *Proof of Stake*. Weaknesses in the protocol could expose vulnerabilities to different types of attacks, necessitating a thorough examination of security considerations. Since MW is built on top of a consensus protocol, our idealized model should be extended to formalize and study the security of the consensus protocol.

8 Future Work

Recognizing the growing complexity and potential for errors in cryptographic proofs, our future plan involves specifying our MW model using an interactive prover. This approach aims to automate the verification of the model, encompassing the modeling of security goals and hardness assumptions to validate the security properties we have stated.

Firstly, our plan involves evaluating tools designed for the verification of cryptographic protocols and implementations, including: `EasyCrypt` [19], `CryptoVerif` [46] and `Tamarin` [47]. In particular, we are especially interested in using `EasyCrypt`, an interactive framework tailored for verifying the security of cryptographic constructions within the computational model. Subsequently, our plan involves specifying our model using the chosen tool, adhering to all the definitions articulated in this work. Then, the interactive prover will be employed to verify all the properties we have presented and proven. Furthermore, we will undertake the specification and verification of the security properties. In particular, the game-based cryptographic proof in Lemma 4 where the goal is to formulate a security reduction through a series of games, demonstrating that any attempt to breach the system's security would result in an efficient solution to the discrete logarithm problem.

The findings presented in this work represent a significant contribution, enabling the analysis of the correctness of the MW protocol and its security properties over an idealized model, transcending any specific implementation. Future research directions include the verification of `Grin` and `Beam` as accurate implementations of the idealized model, ensuring the compliance with security properties through a formal and rigorous approach. Indeed, Guy Corem, a member of the *beam.mw* foundation and one of Beam's founders, reached out to us expressing interest in our work. He further shared it through both his personal Twitter account and Beam's official Twitter account. Moreover, our paper is referenced on Beam's website [1].

References

[1] V. Buterin, "Critical update re: Dao vulnerability, 2017," Available online: https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability (accessed on May 7, 2023).

[2] G. Rosu, "Formal Design, Implementation and Verification of Blockchain Languages Using K (Invited Talk)," in *2nd Workshop on Formal Methods for Blockchains (FMBC 2020)*, ser. OpenAccess Series in Informatics (OASIcs), B. Bernardo and D. Marmsoler, Eds., vol. 84. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 1:1–1:1. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/13414

[3] I. Garfatta, K. Klai, W. Gaaloul, and M. Graiet, "A survey on formal verification for solidity smart contracts," in *2021 Australasian Computer Science Week Multiconference*, ser. ACSW '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3437378.3437879

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, 2008." Available online: https://bitcoin.org/bitcoin.pdf (accessed on May 7, 2023).

[5] Grin, "Introduction to MimbleWimble and Grin, 2016," Available online: https://github.com/mimblewimble/grin/blob/master/doc/intro.md (accessed on May 7, 2023).

[6] B. Foundation, "Beam confidential cryptocurrency, 2020," Available online: https://beam.mw/ (accessed on May 7, 2023).

[7] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of mimblewimble," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Ishai and V. Rijmen, Eds., vol. 11476. Springer, 2019, pp. 657–689. [Online]. Available: https://doi.org/10.1007/978-3-030-17653-2_22

[8] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling up private payments without trusted setup - formal foundations and constructions of ring confidential transactions with log-size proofs," *IACR Cryptol. ePrint Arch.*, p. 580, 2019. [Online]. Available: https://eprint.iacr.org/2019/580

[9] K. Rupic, L. Rozic, and A. Derek, "Mechanized formal model of bitcoin's blockchain validation procedures," in *2nd Workshop on Formal Methods for Blockchains, FMBC@CAV 2020, July 20-21, 2020, Los Angeles, California, USA (Virtual Conference)*, ser. OASIcs, B. Bernardo and D. Marmsoler, Eds., vol. 84. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 7:1–7:14. [Online]. Available: https://doi.org/10.4230/OASIcs.FMBC.2020.7

[10] M. C. Kus Khalilov and A. Levi, "A survey on anonymity and privacy in bitcoin-like digital cash systems," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 2543–2585, 2018.

[11] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of logic-based smart contracts for blockchain systems," in *Rule Technologies. Research, Tools, and Applications - 10th International Symposium, RuleML 2016, Stony Brook, NY, USA, July 6-9, 2016. Proceedings*, ser. LNCS, J. Alferes, L. Bertossi, G. Governatori, P. Fodor, and D. Roman, Eds., vol. 9718. Springer, 2016, pp. 167–183. [Online]. Available: https://doi.org/10.1007/978-3-319-42019-6_11

[1] `https://beam.mw/resources`

[12] Y. Hirai, "Defining the ethereum virtual machine for interactive theorem provers," in *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers*, ser. LNCS, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds., vol. 10323. Springer, 2017, pp. 520–535. [Online]. Available: https://doi.org/10.1007/978-3-319-70278-0_33

[13] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of ethereum smart contracts," in *Principles of Security and Trust - 7th International Conference, POST 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, ser. Lecture Notes in Computer Science, L. Bauer and R. Küsters, Eds., vol. 10804. Springer, 2018, pp. 243–269. [Online]. Available: https://doi.org/10.1007/978-3-319-89722-6_10

[14] G. Pîrlea and I. Sergey, "Mechanising blockchain consensus," in *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018*, J. Andronick and A. P. Felty, Eds. ACM, 2018, pp. 78–90. [Online]. Available: https://doi.org/10.1145/3167086

[15] The Coq Team, "The Coq proof assistant reference manual," Available online: http://coq.inria.fr (accessed on May 7, 2023).

[16] Y. Bertot and P. Castéran, *Interactive Theorem Proving and Program Development : Coq'Art : The Calculus of Inductive Constructions*, 2004th ed., ser. Texts in Theoretical Computer Science. Springer, Isbn: 3540208542, May 2004. [Online]. Available: http://www.worldcat.org/isbn/3540208542

[17] C. Boyd, K. Gjøsteen, and S. Wu, "A Blockchain Model in Tamarin and Formal Analysis of Hash Time Lock Contract," in *2nd Workshop on Formal Methods for Blockchains (FMBC 2020)*, ser. OpenAccess Series in Informatics (OASIcs), B. Bernardo and D. Marmsoler, Eds., vol. 84. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 5:1–5:13. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/13418

[18] R. Metere and C. Dong, "Automated cryptographic analysis of the pedersen commitment scheme," in *Computer Network Security - 7th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2017, Warsaw, Poland, August 28-30, 2017, Proceedings*, ser. Lecture Notes in Computer Science, J. Rak, J. Bay, I. V. Kotenko, L. J. Popyack, V. A. Skormin, and K. Szczypiorski, Eds., vol. 10446. Springer, 2017, pp. 275–287. [Online]. Available: https://doi.org/10.1007/978-3-319-65127-9_22

[19] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P. Strub, "Easycrypt: A tutorial," in *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, ser. Lecture Notes in Computer Science, A. Aldini, J. López, and F. Martinelli, Eds., vol. 8604. Springer, 2013, pp. 146–166. [Online]. Available: https://doi.org/10.1007/978-3-319-10082-1_6

[20] G. Betarte, M. Cristiá, C. Luna, A. Silveira, and D. Zanarini, "Set-based models for cryptocurrency software," *CoRR*, vol. abs/1908.00591, 2019. [Online]. Available: https://arxiv.org/abs/1908.00591

[21] T. Ruffing, S. A. Thyagarajan, V. Ronge, and D. Schröder, "Burning zerocoins for fun and for profit: A cryptographic denial-of-spending attack on the zerocoin protocol," Cryptology ePrint Archive, Paper 2018/612, 2018, https://eprint.iacr.org/2018/612. [Online]. Available: https://eprint.iacr.org/2018/612

[22] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 397–411.

[23] G. Betarte, M. Cristiá, C. D. Luna, A. Silveira, and D. Zanarini, "Towards a formally verified implementation of the mimblewimble cryptocurrency protocol," in *Applied Cryptography and Network Security Workshops - ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19-22, 2020, Proceedings*, ser. Lecture Notes in Computer Science, J. Z. et al, Ed., vol. 12418. Springer, 2020, pp. 3–23. [Online]. Available: https://doi.org/10.1007/978-3-030-61638-0_1

[24] A. Silveira, G. Betarte, M. Cristiá, and C. Luna, "A formal analysis of the mimblewimble cryptocurrency protocol," *Sensors*, vol. 21, no. 17, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/17/5951

[25] ——, “A range proof scheme analysis for the mimblewimble cryptocurrency protocol,” in *2021 IEEE URUCON*, 2021, pp. 329–333.

[26] T. Jedusor, “Mimblewimble, 2016,” Available online: https://scalingbitcoin.org/papers/mimblewimble.txt (accessed on May 7, 2023).

[27] A. Poelstra, “Mimblewimble, 2016,” Available online: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf (accessed on May 7, 2023).

[28] G. Maxwell, “Confidential transactions write up, 2020,” Available online: https://web.archive.org/web/20200502151159/https://people.xiph.org/~greg/confidential_values.txt (accessed on May 7, 2023).

[29] A. Gibson, “An investigation into confidential transactions, 2018,” Available online: https://github.com/AdamISZ/ConfidentialTransactionsDoc/blob/master/essayonCT.pdf (accessed on May 7, 2023).

[30] C. Crépeau, “Commitment,” in *Encyclopedia of Cryptography and Security, 2nd Ed*, H. C. A. van Tilborg and S. Jajodia, Eds. Springer, 2011, pp. 224–227. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_239

[31] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short proofs for confidential transactions and more,” in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 315–334.

[32] G. Maxwell, “Coinjoin: Bitcoin privacy for the real world, 2013,” Available online: https://bitcointalk.org/index.php?topic=279249.0 (accessed on May 7, 2023).

[33] T. Ruffing and G. Malavolta, “Switch commitments: A safety switch for confidential transactions,” in *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers*, ser. Lecture Notes in Computer Science, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. A. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds., vol. 10323. Springer, 2017, pp. 170–181. [Online]. Available: https://doi.org/10.1007/978-3-319-70278-0_10

[34] T. E. Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *Advances in Cryptology, Proceedings of CRYPTO ’84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, vol. 196. Springer, 1984, pp. 10–18.

[35] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *ACM Conference on Computer and Communications Security*, 1993, pp. 62–73.

[36] A. Fiat and A. Shamir, “How to prove yourself: practical solutions to identification and signature problems,” in *Proceedings on Advances in cryptology—CRYPTO ’86*. London, UK: Springer-Verlag, 1987, pp. 186–194.

[37] S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath, “Dandelion: Redesigning the bitcoin network for anonymity,” *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3084459

[38] Grin, “Grin source code switch commitments,” Available online: https://github.com/mimblewimble/secp256k1-zkp/blob/73617d0fcc4f51896cce4f9a1a6977a6958297f8/src/modules/commitment/main_impl.h#L267 (accessed on May 7, 2023).

[39] Grin Community, “Grin source code switch commitment,” Available online: https://github.com/mimblewimble/secp256k1-zkp/blob/73617d0fcc4f51896cce4f9a1a6977a6958297f8/src/modules/commitment/main_impl.h#L267 (accessed on Dec 22, 2023).

[40] Beam, “Beam the scalable confidential cryptocurrency,” Available online: https://docs.beam.mw/BEAM_Position_Paper_0.3.pdf (accessed on May 7, 2023).

[41] ——, “Beam description. Comparison with classical MW, 2018,” Available online: https://docs.beam.mw/BEAM_Comparison_with_classical_MW.pdf (accessed on May 7, 2023).

[42] Grin Community, “Grin dandelion configuration,” Available online: https://github.com/mimblewimble/grin/pull/2628 (accessed on Dec 22, 2023).

[43] Litecoin, “Litecoin,” Available online: https://litecoin.org (accessed on Dec 14, 2023).

[44] ——, “Litecoin project,” Available online: https://github.com/litecoin-project/litecoin (accessed on Dec 14, 2023).

[45] ——, “Litecoin mimblewimble blocks,” Available online: https://github.com/litecoin-project/litecoin/tree/master/src/primitives (accessed on Dec 14, 2023).

[46] B. Blanchet, “Composition theorems for cryptoverif and application to TLS 1.3,” in *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 16–30. [Online]. Available: https://doi.org/10.1109/CSF.2018.00009

[47] Tamarin, “Tamarin prover,” Available online: https://tamarin-prover.github.io (accessed on May 7, 2023).