Kepler Whitepaper

Kepler Team

July 2, 2020

Abstract. Most of the cryptocurrencies, represented by Bitcoin, do not support confidential transactions, which require that the sender address, receiver address, and transaction amount of a transaction cannot be obtained by any third party except the transaction parties. There are a variety of privacy solutions in the industry to achieve confidential transactions, either based on UTXO or Smart contracts. However, current UTXO-based solutions only support the confidential transactions of native assets, while smart-contract-based solutions are expensive to use. Moreover, none of these solutions supports confidential assets, in which both transaction details and asset types are hidden.
MimbleWimble is a novel UTXO-based Confidential Transactions solution, where the transaction addresses and amounts are hidden, providing a high level of privacy for blockchain users. All MimbleWimble implementations do not support confidential assets issuance yet.
This paper presents Kepler, a confidential transaction and confidential assets platform based on the MimbleWimble protocol. In addition to the native asset KMW, Kepler blockchain can also issue any number of user-defined assets, and all assets have complete privacy, hiding transaction parties, transaction values, and asset types at the same time. No trusted setup is required for Kepler, and confidential transactions and confidential assets are much cheaper than those in other smart contract platforms.

Keywords: Confidential Assets · Confidential Transactions · MimbleWimble · Blockchain privacy.

1 Introduction

Cryptocurrencies, represented by Bitcoin [1], realize a decentralized distributed ledger that is traceable and immutable by publicly broadcasting all transactions on the network and verifying transactions across the entire network according to specific consensus mechanism.

A typical cryptocurrency transaction mainly includes the senders' addresses, receivers' addresses, and transaction amount. In most cryptocurrency systems, all the above transaction elements are visible throughout the network to achieve decentralized verification of transactions. This solution sacrifices the privacy of the transaction to a certain extent. Confidential transactions require that the sender address, receiver address, and transaction amount of a transaction cannot be obtained by any third party except the transaction parties [2]. But in most

cryptocurrencies such as Bitcoin, they cannot meet the needs of confidential transactions.

The concept of multi-asset blockchain was proposed in 2013 [3], and many blockchains can already support the issuance of multiple asset types. For example, by adding additional protocols to the Bitcoin script, users can issue various colored coins on Bitcoin [4]; through Ethereum smart contracts, users can quickly issue standardized tokens such as ERC-20 and ERC-721 tokens [5] [6]. However, assets issued through these protocols do not have any privacy [7]. According to the definition in [8], confidential assets should satisfy the privacy of both the transaction amount and the asset type.

To a certain extent, the lack of features of confidential transactions and confidential assets restricts the further development and application of cryptocurrencies.

1.1 Motivation

Confidential Transactions Typical cryptocurrencies such as Bitcoin and Ethereum do not support the confidential transaction feature: First, the association between the Bitcoin address and the user's real identity may be exposed in various ways, for example, the exchange has the user's real information; user information would be exposed during offline payment or OTC transactions; some individuals or institutions analyze the information behind the address through data mining, and then mark and track the user's true identity [9]. Therefore, the vast majority of cryptocurrencies, such as Bitcoin, only have pseudonymity. The addresses of the sender and receiver are transparent. Secondly, the transaction details, such as the specific amount and coin allocation in bitcoin transactions, are also public. According to the amount of all transactions occurring in the entire network, the detailed bitcoin rich list can also be completely calculated.

However, the confidential transaction feature is essential in real business application scenarios. Without privacy, the transaction behavior and true identity of individuals or commercial institutions might be fully exposed to the public. Thus, basic business secret requirements cannot be guaranteed. The fund size and transaction details of institutions and individuals might also be exposed. Any significant amount of transactions may cause panic in the market. These are completely contrary to the traditional business model, which is also one of the reasons why cryptocurrency has not been applied to real business scenarios.

Therefore, supporting the confidential transaction feature is the key to further expanding the application scenarios of cryptocurrencies.

Confidential Assets Bitcoin, Ethereum, and other blockchains can support the issuance of multiple digital assets. Supporting multiple assets on the same blockchain helps increase the potential value of the blockchain and may spawn richer application scenarios. For example, the value of Ethereum is the sum of all tokens on Ethereum (various stablecoins, securitization tokens, various utility tokens, etc.), and the mutual conversion of various token assets on Ethereum can

fully meet the needs of decentralized exchanges. In theory, companies that use blockchain as technical support can transfer many different types of assets to the blockchain in the real world. However, in reality, they do not want to expose transaction data to an open environment because it is equivalent to publishing the company's financial ledger. Hiding the identity of the transaction party, the number of assets and the type of assets in the process of asset movement is an essential part of converting real-world asset on-chain.

Confidential assets is a technology that can hide both the transaction address and amount, as well as the type of assets [8]. It can guarantee the privacy of different types of assets on the multi-asset blockchain. This is particularly important for some assets with small transaction volume because the transactions of such assets can easily expose the trader's privacy even without knowing the transaction amount. Users can arbitrarily issue a variety of assets with privacy attributes through this technology.

UTXO-based vs. Smart-Contract-Based Confidential Assets Solutions Confidential assets solutions are mainly divided into UTXO-based and Smart-Contract-Based.

UTXO (Unspent-Transaction-Output) model is the underlying accounting model adopted by Bitcoin. Compared with the account model, the UTXO model has better security (anti-double spending), anonymity, and parallelism, so it is widely used in cryptocurrency projects. There are a variety of UTXO-based privacy solutions in the industry, which apply privacy technology directly to the addresses or amounts of transaction parties to achieve some level of confidential transactions. For example, Dash based on CoinJoin [10], Monero based on Ring Signature [11], ZCash based on zk-SNARKs [12], and Grin based on MimbleWimble [13]. Currently, UTXO-based privacy solutions only support the confidential transactions of native assets, and cannot issue confidential assets, which hinders the further development of the technology.

Smart contracts are Turing-complete and can be used to issue various assets on Ethereum. The Smart-contract-based privacy solution uses smart contracts to implement the confidential transaction function of the issued assets. Projects such as Zether [7] and AZTEC [14] can send token transactions with hidden addresses or amounts on Ethereum. At present, the Smart-contract-based privacy scheme cannot hide the transaction types. Simultaneously, because the confidential transaction generation process requires a lot of calculations, which consumes many gas costs and dramatically increases the cost of use, it is not easy to implement large-scale applications in reality.

UTXO-based privacy solutions can achieve confidential transactions from a consensus level, achieve simplicity, and have a low cost of use. The Smart-contract-based privacy scheme requires some complicated design, which increases the possibility of introducing security holes and has a high cost of use. Therefore, in terms of security and practicality, UTXO-based privacy solutions have their advantages. Realizing the issuance of UTXO-based confidential assets hiding

asset types and transaction details are the technical difficulties that need to be solved at present.

MimbleWimble is a novel UTXO-based Confidential Transactions solution, where the transaction addresses and amounts are hidden, providing a high level of privacy for blockchain users. MimbleWimble also provides a transaction aggregation feature called Cut-through, where the historical transactions are aggregated, and validators in MimbleWimble only store the latest UTXOs state instead of the entire transaction history of the blockchain, enabling extremely scalability and further anonymity. The design of MimbleWimble relies on Elliptic Curve Cryptography, which is easy to understand and audit. Currently, this solution is used in Grin [15] and Beam [16] projects.

The concept of Confidential Assets is from [8], which proposes to use a single blockchain to track multiple types of assets while keeping their privacy. This technology hides both transaction amounts and asset types, improving the privacy of all assets. With Confidential Assets, users can issue their own assets on the blockchain for privacy-related applications. Although it has been described in paper [17], no Confidential Assets platform based on MimbleWimble has been implemented.

1.2 Our Contributions

We proposed Kepler, a confidential transaction and confidential assets platform based on the MimbleWimble protocol. In addition to the native asset KMW, Kepler blockchain can also issue any number of user-defined assets, and all assets have complete privacy.

Our contributions can be summarized as follows:

- **Multi-Confidential-Asset**. Kepler allows users to issue their confidential assets arbitrarily. At the time of issuance, the asset type is public, and the circulation is public. We provide templates and tools for issuing assets, and users can issue custom assets with one click.
- **Confidentiality**. Kepler's native token KMW and user-issued assets are completely private. Using MimbleWimble's confidential transaction solution, Pedersen Commitment and elliptic curve features, transaction parties can hide the addresses, transaction amounts, and transaction asset types.
- **Parallelism**. A transaction on the Kepler network can include multiple inputs and multiple outputs. A single transaction can contain multiple assets at the same time, ensuring that all assets can be traded in parallel, while further enhancing the privacy of the transaction.
- **Scalability**. All assets on Kepler (including native assets and confidential assets issued by users) can be cut-through to delete all intermediate transactions. This aspect reduces the on-chain data that nodes need to store, increasing the scalability. On the other hand, it also deletes the transaction history, making it untraceable, which is an essential supplement to transaction privacy.

- **Flexibility**. All Kepler transactions can specify the asset type used to pay the transaction fee, and do not necessarily need to pay with the native token KMW. This broadens the usage scenarios of various assets on Kepler and simultaneously reduces the cost of using confidential assets for different asset holders on Kepler.
- **ASIC Friendly**. Kepler uses the ASIC-friendly C31 + mining algorithm, which attracts a large number of miners and mining equipment producers to join mining. In the ecology, miners continue to invest in the competition for block rewards. The entire network's computing power continues to increase, making the network less vulnerable to 51% attacks and ensuring the security of the system.

The rest of the whitepaper is organized as follows. We provide some background and detailed technologies of MimbleWimble Confidential Transaction in Section 2. Section 3 provides a detailed description of Kepler Confidential Assets solution, focusing on the privacy of multiple user-defined confidential assets. Section 4 describes the blocks and chain state of Kepler blockchain. Section 5 shows the economic model of Kepler. Section 6 discusses the future works of Kepler project.

2 Confidential Transactions Based on MimbleWimble

This section presents the main technologies of MimbleWimble as the basics of Kepler's confidential assets. We will cover the Elliptic Curve, Pedersen Commitment, signature structure and Bulletproofs, and put them all together to introduce MimbleWimble transaction structure. Finally, we will present the concept of cut-through in MimbleWimble protocol.

2.1 Basis of Elliptic Curve Cryptography

Classic cryptographic systems like RSA are based on numbers we are used to, but modern cryptography (since the 1980s) had moved on to using elliptic curve points instead. Elliptic curve points can support arithmetic operations (in group theory terminology, the points form a Group), so you could just drop them in as numbers into the same equations.

Suppose an Elliptic Curve for cryptography is an infinite set of points, which we call it Curve C. These points can be added, subtracted, or multiplied by integers (also called scalars). Given such a point H, an integer k, and using the scalar multiplication operation, we can compute $k * H$, which is also a point on curve C. Given another integer j we can also calculate $(k + j) * H$, which equals $k * H + j * H$. The addition and scalar multiplication operations on an elliptic curve maintain the commutative and associative properties of addition and multiplication: $(k + j) * H = k * H + j * H$.

Although the multiplication in ECC (Elliptic Curve Cryptography) is trivial, the division by Elliptic Curve points is impossible in practice. That is to say,

suppose an integer k and a point H on Elliptic Curve, even one knows the value of $k * H$, it is extremely difficult to deduce k when k is a huge number. Using this principle, we can treat k as the private key, a huge integer picked by users, while $k*H$ as the corresponding public key. The public key can be used safely in public without exposing the private key. According to this feature, the formula $(k + j) * H = k * H + j * H$ demonstrates that a public key obtained from the addition of two private keys $((k+j)*H)$ is identical to the addition of the public keys for each of those two private keys $(k * H + j * H)$.

These features above are the basic knowledge that will help us to understand how MimbleWimble works and is implemented.

2.2 Pedersen Commitment

In typical UTXO-based blockchain like Bitcoin, the transaction inputs and outputs are all UTXOs, which have a specific value and demonstrate the ownership by Script. In MimbleWimble, however, each input or output of a transaction is expressed by a form called Pedersen commitment:

$$C = rG + vH \tag{1}$$

Where C, G, and H are all points on a given Elliptic Curve E, while r and v are integers. C is called the Pedersen commitment. G and H denote two points randomly selected from the elliptic curve E. For a given asset like the Kepler native coin KMW, the G and H should keep consistent all the time. v is the transaction value. r is a private key chosen by the owner of the input/output, also called the blinding factor. It stands for the ownership, which will be explained later, and makes the value of v difficult to be revealed from the commitment. Through the Pedersen commitment, the value and owners of each input and output are kept confidential.

2.3 Transaction Signature

Suppose we have a transaction with M inputs $\{C_{\text{in},m}\}_{m=1}^{M}$ and N outputs $\{C_{\text{out},n}\}_{n=1}^{N}$ in Kepler, where the inputs and outputs are all Pedersen commitments. The sender and receiver should prove two things before other validators of the network accept the transaction as valid:

- **Balance of values**. The value of outputs should be equal to the value of inputs so that no coins are created from thin air.
- **Ownership of inputs**. The senders have the right private keys to spend the coin.

To prove the two facts above, one can validate an equation according to the inputs and outputs commitments as follows:

$$
\begin{aligned}
&\sum_{n=1}^{N} C_{\text{out},n} - \sum_{m=1}^{M} C_{\text{in},m} \\
&= \sum_{n=1}^{N} \left(r_{\text{out},n} G + v_{\text{out},n} H\right) - \sum_{m=1}^{M} \left(r_{\text{in},m} G + v_{\text{in},m} H\right) \\
&= \left(\sum_{n=1}^{N} r_{\text{out},n} - \sum_{m=1}^{M} r_{\text{in},m}\right) G + \left(\sum_{n=1}^{N} v_{\text{out},n} - \sum_{m=1}^{M} v_{\text{in},m}\right) H \\
&= eG
\end{aligned}
\tag{2}
$$

where e is called the excess value and defined as:

$$
\sum_{n=1}^{N} r_{\text{out},n} - \sum_{m=1}^{M} r_{\text{in},m} = e \tag{3}
$$

The equation does not sum to zero, and we have the excess value e which is the result of the summation of all blinding factors. Because eG is a valid public key for the generator point G the input and output values must sum to zero (ignoring the transactions fees) and the transaction is thus valid, since $x * G + y * H$ is a valid public key for generator point G if and only if $y = 0$, which means:

$$
\sum_{n=1}^{N} v_{\text{out},n} - \sum_{m=1}^{M} v_{\text{in},m} = 0 \tag{4}
$$

so that no additional coin is created.

G and H are random elliptic curve points whose discrete logarithms with respect to each other are unknown (the Elliptic curve discrete logarithm problem). Therefore, the creator of the transaction can generate a signature sig using e obtained through Eq.(3) as the private key. Here, the transaction creator refers to all senders and receivers for simplicity, since the transaction creation process in MimbleWimble requires their participation to complete. The signature is later verified by the blockchain verifier using eG obtained through Eq. (2) as the public key. In this way, the verifiers in the network can easily validate that:

- Balance of amounts is valid. The left side of Eq. (2) is a valid public key on the elliptic curve using the generator point G. That is to say, Eq. (4) holds.
- Ownership of inputs is valid. The transaction senders and receivers know the private key e. Combined with Eq. (3) and Bulletproofs introduced below. It can be further proved that the sum input blinding factors are known.

The eG above is a signature built with the excess value attached to every transaction, together with some additional data (like transaction fees paid to miners), which is called a transaction kernel and is checked by all validators.

Change and Fees For UTXO-based blockchains, there could be several additional change outputs if the value of the inputs is larger than the value of the expected outputs paid to the receiver. The change in a MimbleWimble transaction works similarly. However, since every input and output of MimbleWimble transaction is a Pedersen Commitment, you need to generate another private key as a blinding factor to protect your change output, i.e., parts of the outputs in Eq.(1) might be change outputs, while the equation still holds.

In decentralized blockchains, the users need to pay transaction fees to the miner to get their transactions included in the new block. In MimbleWimble, fees are a particular output of the transaction, since it is published as cleartext without blinding factor, which is included in the transaction kernel. The values of transaction fees are public in MimbleWimble. It is a simple concept so that we will ignore transaction fees in the following discussion for simplicity.

2.4 Bulletproofs

Range Proofs In all the equations in section 2.3, we rely on the transaction values to always be positive. The introduction of negative amounts would enable attackers to create new funds in every transaction while still keep the equation holds.

For example, one could create a transaction with an input of 5 and outputs of 10 and -5 and still keep the transaction balanced. This can't be easily detected because even if v is negative, the corresponding point $v * H$ still looks like a normal point on the Elliptic Curve.

To solve this problem, MimbleWimble leverages another cryptographic concept called range proofs. Range proof allows proving that a secret integer belongs to a certain interval, without revealing the number. In the context of payment systems like Kepler, if party A wants to transfer money to party B, then it is possible to utilize Range Proofs to prove that the amount of money in the transaction is positive, otherwise, if the amount is negative, such transaction would in fact transfer money in the opposite direction, i.e., from B to A.

Bulletproofs There are several different schemes to achieve range proofs, for example, Square decomposition, Signature-based, and Multi-base decomposition branches [18]. Unfortunately, all these schemes depend upon a trusted setup, which may not be suitable in the context of cryptocurrencies. For instance, if adversaries can circumvent this trusted setup, they would be able to create money out of thin air [2].

Bulletproofs technology is a non-interactive zero-knowledge proof protocol with very short proofs and without a trusted setup, the proof size is only logarithmic in the witness size. For a Pedersen commitment like Eq. (1), the proof system would convince the verifier that $v \in [0, 2^n - 1]$ without revealing v and r. The transaction creator needs to generate a proof π for each output commitment C with the knowledge of v, r. Later, π is verified by the verifier with C but without v, r. The proof generation and verification algorithms can be expressed as:

$$\begin{aligned} \text{Generate}(r, v, G, H, n) &\rightarrow \pi \\ \text{Verify}(C, G, H, n) &\rightarrow \{\text{true}, \text{false}\} \end{aligned} \tag{5}$$

Since the knowledge of r is necessary for the proof generation algorithm, the transaction creator must know all output blinding factors $\{r_{\text{out},n}\}_{n=1}^{N}$ to generate proofs for all outputs. According to Eq. (3), since the creator knows e demonstrated by the signature, it proves that the sum of input blinding factors is known to the creator; that is, the creator owns the inputs.

2.5 Put It All Together

To sum up, a typical MimbleWimble transaction should include the following[8]:

- From the senders, a set of inputs that reference and spend a set of previous outputs in the form of Pedersen Commitment: $C_{\text{in},1}$,...,$C_{\text{in,M}}$
- From both the senders and receivers, a set of new outputs $(C_{\text{out},1}, \pi_1)$,..., $(C_{\text{out},1}, \pi_N)$that include:
  - A value and a blinding factor (which is just a new private key) multiplied on a curve and summed up a Pedersen commitment: $C_{\text{out,n}}$
  - A range proof for corresponding Pedersen commitment shows that the value is non-negative: π_n
- The transaction fee included in the kernel, in cleartext,
- A signature whose private key is computed by taking the excess value (the sum of all output values plus the fee, minus the input values).

The transaction would be broadcasted to the network by the creators and later verified by other full nodes (call verifiers) in the network.

2.6 Transaction Verification

When a verifier receives a MimbleWimble transaction above, it needs to verify as following before sending it on to its peers [19]:

- All inputs commitments come from the current UTXO set, which is kept track of by all full nodes.
- All outputs have a valid range proof by verifying against their corresponding commits.
- The values balance by evaluating Eq.(4) to make sure that no coins are created or destroyed in the transaction.
- The signature in the kernel is valid.
- Other consensus checks, such as the transaction fees in the kernel are higher than the minimum.

If all these checks pass, then the verifier node will forward the transaction on to its peers, and it will eventually be mined and be added to the blockchain. At the same time, the senders/receivers and transferred values are kept confidential throughout the process.

3 MimbleWimble Blocks and Chain State

This section introduces another fundamental concept of MimbleWimble called cut-through, and the block layout and chain state of typical MimbleWimble blockchain.

3.1 Transaction Aggregation

According to section 2.6, for a valid transaction in MimbleWimble, Eq.(2) always holds, i.e., the j-th transaction in a block is validated by determining that the kernel excess is a valid public key. Suppose this transaction has M_j inputs $\{C_{\text{in},m}\}_{m=1}^{M_j}$ and N_j outputs $\{C_{\text{out},n}\}_{n=1}^{N_j}$, we have:

$$\sum_{n=1}^{N_j} C_{j,\text{out},n} - \sum_{m=1}^{M_j} C_{j,\text{in},m} \quad = e_j G \tag{6}$$

where $e_j G$ is the kernel access of the j-th transaction.

Since every transaction in a block is a set of inputs, outputs and transaction kernels, it is straightforward that if we sum the outputs, subtract the inputs from it and equating the resulting Pedersen commitment to the sum of the kernel excesses:

$$\begin{aligned} &\sum_{j=1}^{J} \left(\sum_{n=1}^{N_j} C_{j,\text{out},n} - \sum_{m=1}^{M_j} C_{j,\text{in},m} \right) \\ &= \sum_{j=1}^{J} \sum_{n=1}^{N_j} C_{j,\text{out},n} - \sum_{j=1}^{J} \sum_{m=1}^{M_j} C_{j,\text{in},m} \\ &= \left(\sum_{j=1}^{J} e_j \right) G \end{aligned} \tag{7}$$

It is in the same form of a typical MimbleWimble transaction described in Section 2.5, since the order of a transaction in a block makes no difference. In this way, all the transactions in a block can be aggregated to one bigger transaction, and MimbleWimble blocks can be treated exactly as MimbleWimble transactions. Transaction aggregation further increases the privacy of the transaction.

3.2 Kernel Offsets

In some cases, when there are only a few transactions in a block, even though the transactions are aggregated, it is still possible to reconstruct the constituent transactions in a block, which harms the privacy of the transactions.

Suppose we have an aggregated block that contains two transactions.

$$(C_{\text{in},1}, C_{\text{in},2}, C_{\text{in},3}) \rightarrow (C_{\text{out},1}, C_{\text{out},2})\,(kernel_1, kernel_2) \tag{8}$$

where $C_{\text{in},1}, C_{\text{in},2}, C_{\text{in},3}$ are three inputs after aggregation, $C_{\text{out},1}, C_{\text{out},2}$ are outputs, and $kernel_1, kernel_2$ are corresponding kernels.

Since the transaction has only several inputs and outputs, it is not difficult to try all possible permutations to recover one of the transactions, where the Eq.(2) is successfully validated.

$$(C_{\text{in},1}, C_{\text{in},2}) \rightarrow (C_{\text{out},1})\,(kernel_1) \tag{9}$$

The remaining parts of the aggregated transaction can be used to reconstruct the other valid transaction:

$$(C_{\text{in},3}) \rightarrow (C_{\text{out},2})\,(kernel_2) \tag{10}$$

The solution is to redefine the kernel excess e from $r*G$ to $(r-kernel\_offset)*G$ and distribute the kernel offset to be included with every transaction kernel. Thus, the kernel offset is a blinding factor that needs to be added to the excess value to ensure the sum of the commitments to zero:

$$\begin{aligned} &\sum_{n=1}^{N} C_{\text{out},n} - \sum_{m=1}^{M} C_{\text{in},m} \\ &= rG \\ &= (r-a)*G + a*G \\ &= eG + aG \end{aligned} \tag{11}$$

where a is the kernel offset.

For a commitment $r*G+0*H$ with the offset a, the transaction is signed with $(r-a)$ and a is published so that $r*G$ can be calculated to verify the validity of the transaction. During block construction, all kernel offsets are summed to generate a single aggregate kernel offset to cover the whole block. The kernel offset for any individual transaction is then unrecoverable, and the problem is solved.

$$\begin{aligned} &\sum_{j=1}^{J}\sum_{n=1}^{N_j} C_{j,\text{out},n} - \sum_{j=1}^{J}\sum_{m=1}^{M_j} C_{j,\text{in},m} \\ &= \left(\sum_{j=1}^{J} e_j\right) G + kernel\_offset * G \end{aligned} \tag{12}$$

The kernel offset hides which kernel belongs to which transaction, and we only have a summed kernel offset stored in the header of each block.

3.3 Cut-through

Blocks let miners assemble multiple transactions into a single set that is added to the chain.

Within some specific block, some outputs are directly spent by following inputs in the same block. Given that the structure of each transaction does not matter, and since all transactions individually sum to zero, the sum of all transaction inputs and outputs must be zero. Therefore, there are only two things to need to be checked in a block:

- The ownership has been proven, which comes from the transaction kernels;
- The whole block did not create any coins, other than what are allowed as the mining reward.

As a result, matching inputs and outputs can be eliminated, as their contribution to the overall sum cancels out, which leads to a more compact block.

Note that all transaction structure has been eliminated and the order of inputs and outputs does not matter anymore. However, the sum of all inputs and outputs is still guaranteed to be zero.

3.4 MimbleWimble Blocks

By putting all the above together, the basic structure of a MimbleWimble block can be described as:

- A block header (includes the basic information and hash roots of the block)
- The list of all remaining inputs after transaction aggregation and cut-through
- The list of all remaining outputs after transaction aggregation and cut-through
- A kernel offset for the whole block
- The list of transaction kernels, which contains:
  - The public key obtained from the sum of excess commitment for the transaction
  - The signatures generated using the excess value
  - The mining fee in cleartext

In such a block structure, all transactions are in the same form. Since the structure of the transaction are removed, and transactions are only represented by their transaction kernels, it is impossible to tell different inputs and outputs, which provides a very high level of privacy.

3.5 MimbleWimble Chain State

Pruning Pruning takes this same concept with cut-through but goes into past blocks. Therefore, if an output in a previous block gets spent, it will be removed from that block. Pruning makes the ledger smaller and more scalable.

After pruning, the whole blockchain looks just like a huge aggregated block, with the following information [13] [19] :

– The list of block headers in history
– The list of all remaining inputs after transaction aggregation and cut-through of all historical blocks. Since all the coins are coming from mining, the inputs list is just a list of coinbase commitments.
– The list of all remaining outputs after transaction aggregation and cut-through of all historical blocks.
– The transaction kernels for each transaction.

The chain state at any point in time can thus be described with the following information:

– The total amount of coins that have been mined.
– The complete set of UTXOs
– The transaction kernels for each transaction

4 Kepler Confidential Assets

This section introduces the technologies of Kepler Confidential Assets based on MimbleWimble protocol, with which users can issue and transfer multiple confidential assets on Kepler blockchain.

4.1 Multiple Assets

Pedersen Commitment The Pedersen commitment of a single-asset MimbleWimble UTXO is provided in Eq. (1), where G and H are unchanged for a specific type of asset. Suppose there are several different assets on the same blockchain, we can assign different H for a different type of assets. Suppose there are total I assets, the Pedersen commitment for the i-th asset is expressed as:

$$C_i = rG + vH_i \tag{13}$$

where $i \in 1, ..., I$ and H_i is called the asset tag.

Note that the concept above works only if each asset tag is selected randomly, whose discrete logarithm is not known with respect to G and any other asset tags. Otherwise, different types of assets might be mixed-up and cause security issues. When issuing a new asset, a random oracle is needed to map the issuance query to a random and fixed point on the elliptic curve as its asset tag, detailed in the next subsection.

Multiple Assets Signature Eq. (2) can be extended to the case of multiple assets. Suppose a transaction has inputs and outputs from I assets. For the i-th asset, there are M_i inputs $\{C_{\text{in},m}\}_{m=1}^{M_i}$ and N_i outputs $\{C_{\text{out},n}\}_{n=1}^{N_i}$. Then the following equation holds:

$$
\begin{aligned}
&\sum_{i=1}^{I}\sum_{n=1}^{N_i} C_{\text{out},i,n} - \sum_{i=1}^{I}\sum_{m=1}^{M_i} C_{\text{in},i,m} \\
&= \sum_{i=1}^{I}\sum_{n=1}^{N_i} (r_{\text{out},i,n}G + v_{\text{out},i,n}H_i) - \sum_{i=1}^{I}\sum_{m=1}^{M_i} (r_{\text{in},i,m}G + v_{\text{in},i,m}H_i) \\
&= \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r_{\text{out},i,n} - \sum_{m=1}^{M_i} r_{\text{in},i,m}\right)G + \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} v_{\text{out},i,n} - \sum_{m=1}^{M_i} v_{\text{in},i,m}\right)H_i \\
&= eG
\end{aligned}
\tag{14}
$$

in which

$$
\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} v_{\text{out},i,n} - \sum_{m=1}^{M_i} v_{\text{in},i,m}\right) = 0 \text{ for } i \in \{1, ..., I\} \tag{15}
$$

$$
\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r_{\text{out},i,n} - \sum_{m=1}^{M_i} r_{\text{in},i,m}\right) = e \tag{16}
$$

Since no new funds would be created for each asset, i.e.

$$
\sum_{n=1}^{N_i} v_{\text{out},i,n} - \sum_{m=1}^{M_i} v_{\text{in},i,m} = \quad 0 \text{ for } i \in \{1, ..., I\} \tag{17}
$$

It is evident that the excess value e, public key eG, balance of amounts, and ownership of inputs still follow the same relationship as Eq. (2) (3) (4). So the signature is also valid for multiple assets. Still, we will ignore the change and fees in the following discussion for simplicity.

Bulletproofs Changing the value of H would not affect the calculation of Bulletproofs since H is one of the input parameters to the proof generation and verification algorithms in Eq. (5). But the changed value needs to be publicly recorded in the output for the verifier to use. As a result, the output is expanded to the form of (C, π, H).

4.2 Kepler Asset Tags

There has been a lot of research [20] on hash functions that not only can map an arbitrary bit string to a point on an elliptic curve but also are indifferentiable from random oracles. Such functions can be used in several places, for example, in BLS signature [21], the message needs to be hashed to a point in a prime-order subgroup of a pairing-friendly elliptic curve.

The initial solution is from [22] called MapToGroup. It merely tries hashing several times by attaching a number to the message and incrementing it on fail.

Let the message be m. Then if $hash(0||m)$ is not the x-coordinate of a rational point on the elliptic curve, try $hash(1||m)$, $hash(2||m)$ and so on until finally get one that matches. The probability of success for each try is about 1/2.

However, this solution has an obvious drawback: it cannot execute in constant time, that is, time independent of the hash input. For the blockchain, a constant-time hash function is necessary. If someone deliberately chooses a message difficult to hash and sends it to the blockchain, plenty of time would be wasted for nodes to verify it.

Several methods have solved the problem of constant-time mapping. Shallue and van de Woestijne [23] describe a mapping that is computable in deterministic polynomial time. Fouque and Tibouchi [24] give a concrete set of parameters for this mapping geared toward BN curves.

We use an extension of [24] as a random oracle for users to generate asset tags. First, the user needs to create a private key for the new asset. Then, the corresponding public key and other information for the issuance, such as the supply, are combined, as the issuance query m. Finally, a hash function $H(m)$ uses m as its input and outputs a point on the elliptic curve $E(\mathbb{F}_q)$ as the asset tag ($\mathbb{F}_q$ is the finite field with q elements). The user should also use the private key to generate a signature sig' for the issuance query m to demonstrate m is valid, and the issuance is authorized.

Here, the hash function $H(m)$ is expressed as:

$$H(m) = f(h_1(m)) + f(h_2(m)) \tag{18}$$

where $h_1(m), h_2(m)$ are independent hash functions as random oracles to $\mathbb{F}_q$, and $f(t)$ is a mapping from $\mathbb{F}_q$ to E. $h_1(m), h_2(m)$ can be realized by SHA256 as following:

$$h_i(m) = \text{SHA256}(i||m) \bmod q,\ i \in \{1, 2\}$$

The results produced by this method have some bias from the uniform random distribution, but this bias is negligible for most elliptic curves.

Fouque and Tibouchi provide an implementation of $f(t)$ on the elliptic curve of the form:

$$E:\ y^2 = g(x) = x^3 + b,\ b \neq -1$$

over field $\mathbb{F}_q$ with $q \equiv 7 \pmod{12}$. They prove that at least one of the three values:

$$x_1 = \frac{-1 + \sqrt{-3}}{2} - \frac{\sqrt{-3} \cdot t^2}{1 + b + t^2}$$

$$x_2 = \frac{-1 - \sqrt{-3}}{2} + \frac{\sqrt{-3} \cdot t^2}{1 + b + t^2}$$

$$x_3 = 1 - \frac{\left(1 + b + t^2\right)^2}{3t^2}$$

is the x-coordinate of a point on E, denoted x_i; that is, $g(x_i)$ is a square over $\mathbb{F}_q$. Then $f(t)$ can be expressed as:

$$f(t) = \left(x_i, \chi_q(t) \cdot \sqrt{g\left(x_i\right)}\right),\ \text{for } t \in \mathbb{F}_q^*$$

where $\chi_q(t)$ is the sign of t which is used as the sign for the y-coordinate. $\mathbb{F}_q^* = \mathbb{F}_q \backslash \{0\}$, and $f(0)$ can be assigned an arbitrary point. They also prove that $f(t)$ reaches about 9/16ths of all points on the curve.

4.3 Blinding Asset Tags

The asset tag H in each output can be hidden by replacing it with an asset commitment A of the form:

$$A = H + sG$$

where s is a secret value randomly selected by the output owner and used to blind the asset tag. Thus, the asset tag of the output is only known to its owner, which further improves the privacy of assets.

Pedersen Commitment with Blinding Factor On this basis, the Pedersen commitment for the i-th asset in Eq. (13) can be derived as:

$$\begin{aligned} C_i &= rG + vA_i \\ &= rG + v(H_i + sG) \\ &= (r + vs)G + vH_i \end{aligned}$$

where $i \in \{1, ..., I\}$. It is equivalent to Eq. (13), because only the blinding factor r is replaced by another secret value $(r + vs)$, and other parameters remain unchanged. With this replacement, Eq. (14) (16) (17) can still hold, and the signature continues to work.

Multiple Assets Signature with Assets Tag Blinding Assuming $r' = r + vs$, the following equation still holds:

$$\begin{aligned} &\sum_{i=1}^{I}\sum_{n=1}^{N_i} C_{\text{out},i,n} - \sum_{i=1}^{I}\sum_{m=1}^{M_i} C_{\text{in},i,m} \\ &= \sum_{i=1}^{I}\sum_{n=1}^{N_i} \left(r'_{\text{out},i,n}G + v_{\text{out},i,n}H_i\right) - \sum_{i=1}^{I}\sum_{m=1}^{M_i} \left(r'_{\text{in},i,m}G + v_{\text{in},i,m}H_i\right) \\ &= \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r'_{\text{out},i,n} - \sum_{m=1}^{M_i} r'_{\text{in},i,m}\right)G + \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} v_{\text{out},i,n} - \sum_{m=1}^{M_i} v_{\text{in},i,m}\right)H_i \\ &= eG \end{aligned} \tag{19}$$

in which

$$\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} v_{\text{out},i,n} - \sum_{m=1}^{M_i} v_{\text{in},i,m}\right) = 0 \text{ for } i \in \{1, ..., I\} \tag{20}$$

$$\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r'_{\text{out},i,n} - \sum_{m=1}^{M_i} r'_{\text{in},i,m}\right) = e \tag{21}$$

It is equivalent to Eq. (14), because only the blinding factor r is replaced by another secret value $(r + vs)$, and other parameters remain unchanged. With this replacement, Eq. (19) (20) (21) can still hold, and the signature continues to work.

Bulletproofs A is also a point on the elliptic curve so that it can replace H as the input to the proof generation and verification algorithms. Thus, the output should be recorded as (C, π, A).

Asset Surjection Proofs Since the asset tag is hidden, the legitimacy of the tag cannot be guaranteed. Malicious users can use illegal tags to attack the blockchain, especially tags whose discrete logarithms are known. For example, consider the asset tag $-H$ and its asset commitment $A' = -H + sG$. Any amount of asset A' would correspond to a negative amount of asset A, thereby increasing the supply of A.

To solve this problem, we introduce the ASP (Asset Surjection Proof) [8] technology. It generates a cryptographic proof for an asset commitment A and a set of asset commitments $\{A_i\}_{i=1}^{J}$, proving that A has the same asset tag H as a commitment A_K in the set, without exposing H and A_K to the verifier. In MimbleWimble, the proof can be generated for A and a subset of all issued asset tags $\{H_i\}_{i=1}^{J}$. In this way, it proves $H \in \{H_i\}_{i=1}^{J}$ without revealing the value of H, and thus demonstrates the legitimacy of A.

The ASP technology works as follows. The prover first computes a set of elliptic curve points $\{P_i\}_{i=1}^{J}$ by

$$P_i = A - H_i = \begin{cases} H + sG - H_i, & i \neq K \\ sG, & i = K \end{cases}$$

where only P_K is a point whose discrete logarithm to G is known due to $H = H_K$. Then the prover uses $\{P_i\}_{i=1}^{J}$, s, and G to generate an AOS (Abe-Ohkubo-Suzuki) ring signature [25]. The signature proves to the verifier that the discrete logarithm of one of the points in $\{P_i\}_{i=1}^{J}$ is known to the prover. This statement holds only when $H \in \{H_i\}_{i=1}^{J}$. Thus the ASP can be written as $(\mathcal{H}, \sigma)$, where $\mathcal{H}$ stands for $\{H_i\}_{i=1}^{J}$ and σ is the ring signature.

Fields in the output are expanded to $(C, \pi, A, \mathcal{H}, \sigma)$, significantly increasing the storage space of the output. The size of $\mathcal{H}$ is proportional to J, and the size of σ is proportional to $J + 1$ in AOS. If we choose a small J, although the sizes of $\mathcal{H}$ and σ can be reduced, the value of H is easy to be exposed due to being mixed with a small asset set.

To reduce the size of $\mathcal{H}$ while keeping the value of J unchanged, a compact representation of H can be adopted. First, we store all issued asset tags in an append-only list. Then, one can select tags within an interval of the list by specifying the start and end positions of the interval. Finally, $\mathcal{H}$ consists of the start and end positions of multiple intervals, representing a set of asset tags in these intervals.

We consider a more efficient ring signature algorithm for reducing the size of σ. Suppose in the elliptic curve, the size of a point is X, and the size of a scalar is Y. The size of σ is $(J+1)Y$ in AOS, which grows linearly with the size of the ring. A logarithmic size ring signature algorithm based on the Sigma-protocol is provided by [26]. Its size is $(4\log J)X + (3\log J + 1)Y$, which costs less storage space for large rings. The verification time is still linear. But when the same ring is used many times, or there is a significant overlap between different rings, the cost of verification can be further reduced by batching the verification of many signatures. This situation would be common when users select asset tags through the compact representation of $\mathcal{H}$.

4.4 Put It All Together

Overall, a transaction for Kepler Confidential Assets is shown as follows.

- From the senders, a set of inputs that reference and spend a set of previous outputs in the form of Pedersen Commitment with blinding asset tags: $C_{\text{in,1}}$..., $C_{\text{in,M}}$
- From both the senders and receivers, a set of new outputs $(C_{\text{out,1}}, \pi_1, A_1, \mathcal{H}_1, \sigma_1)$, ..., $(C_{\text{out,N}}, \pi_N, A_N, \mathcal{H}_N, \sigma_N)$that include:
  - A value and a blinding factor (which is just a new private key) multiplied on a curve and summed to be a Pedersen commitment with blinding asset tags: $C_{\text{out,n}}$
  - A range proof for corresponding Pedersen commitment shows that the value is non-negative: π_n
  - An asset tag with blinding factor: A_n
  - A asset surjection proof for corresponding Pedersen commitment shows that the asset tag is valid: $(\mathcal{H}_n, \sigma_n)$
- The transaction fee and corresponding asset tag included in the kernel, in cleartext.
- The Kernel:
  - For ordinary transactions, the kernel is a signature sig whose private key is computed by taking the excess value (the sum of all output values plus the fee, minus the input values).
  - For confidential assets issuance, the kernel is (sig, sig', m), where the m is issuance query. Except for the sig above, the user should also use the private key to generate a signature sig' for the issuance query m to demonstrate m is valid, and the issuance is authorized.

4.5 Kepler Confidential Assets Verification

When a verifier receives a Kepler multi-assets confidential transaction above, it needs to verify as following before sending it on to its peers:

- All inputs commitments come from the current UTXO set of different asset tags, which is kept track of by all full nodes.

– All outputs have a valid range proof by verifying against their corresponding commits.
– All outputs have valid assets surjection proof by verifying against their corresponding commits.
– The values balance by evaluating Eq.(20) to make sure that no coins are created or destroyed in the transaction for any confidential assets.
– The signature in the kernel is valid.
– Validate other consensus checks, such as the transaction fee value and corresponding asset types.

If all these checks pass, then the verifier node will forward the transaction on to its peers, and it will eventually be mined and be added to the blockchain, while the senders/receivers, transferred values, and asset types are kept confidential all through the processes.

4.6 Kepler Blocks and Chain State for Multiple Assets

All the transactions with different asset types also keep the same concepts of typical MimbleWimble transactions mentioned in Section 3.

Multi-assets Transaction Aggregation According to section 4, for a valid multi-asset transaction in Kepler, Eq.(14) always holds. i.e. the j-th transaction with i-th asset type in a block is validated by determining that the kernel excess is a valid public key. Suppose this transaction has I_j different asset types. For the i-th asset type, there are $M_{i,j}$ inputs $\{C_{\text{in},m}\}_{m=1}^{M_{i,j}}$ and $N_{i,j}$ outputs $\{C_{\text{out},n}\}_{n=1}^{N_{i,j}}$:

$$\sum_{n=1}^{N_{i,j}} C_{i,j,\text{out},n} - \sum_{m=1}^{M_{i,j}} C_{i,j,\text{in},m} \quad = e_{i,j}G \tag{22}$$

where $j \in \{1, ..., J\}$, $i \in \{1, ..., I_j\}$

If we sum the outputs, subtract the inputs from it, and equate the resulting Pedersen commitment to the sum of the kernel excesses, the transactions in a block can be aggregated. The following equation holds:

$$\begin{aligned} &\sum_{j=1}^{J}\sum_{i=1}^{I_j}\left(\sum_{n=1}^{N_{i,j}} C_{i,j,\text{out},n} - \sum_{m=1}^{M_{i,j}} C_{i,j,\text{in},m}\right) \\ &= \sum_{j=1}^{J}\sum_{i=1}^{I_j}(e_{i,j}G) \\ &= \left(\sum_{j=1}^{J}\sum_{i=1}^{I_j} e_{i,j}\right)G \end{aligned} \tag{23}$$

The kernel offset, cut-through, and pruning concept of Kepler multi-asset transactions are similar to typical MimbleWimble transactions.

Kepler Block Structure with Multiple Assets The basic structure of a Kepler block with multiple assets can be described as:

- A block header (includes the necessary information and hash root of the block).
- The list of all remaining inputs with different asset tags after transaction aggregation and cut-through.
- The list of all remaining outputs with different asset tags after transaction aggregation and cut-through.
- A kernel offset for the whole block
- The list of transaction kernels, which contains:
  - The public key obtained from the sum of excess commitment for the transaction
  - The signatures sig generated using the excess value for ordinary transaction, or the signatures (sig, sig', m) for asset issuance
- The mining fee in cleartext.

In such a block structure, all transactions of different asset types are in the same form. Since the structure of the transaction are removed, and transactions are only represented by their transaction kernels, it is impossible to tell different inputs and outputs or their asset types, which provides a very high level of privacy.

Kepler Chain State with Multiple Assets The chain state at any point in time can thus be described with only the following information:

- The asset tags set that includes all confidential asset types.
- The total amount of each kind of asset that has been mined(Native KMW) or issued(Other confidential assets).
- The complete set of UTXOs with different asset tags
- The transaction kernels for each transaction

5 Token Economics

5.1 Incentive

Kepler's native coin is KMW, and all KMWs are generated through block rewards using PoW as consensus. Kepler adopts the $C31$ mining algorithm. Miners are incentivized by all the block rewards, as well as transaction fees paid by users. As the native coin of the Kepler system, KMW can be used as fees for issuing, trading and destroying confidential assets. KMW can also be used as a confidential asset. Users need to lock a certain amount of their KMWs when issuing new confidential assets.

5.2 Monetary Policy

KMW adopts the deflation model with a max supply of $2,138,639,000$ and a genesis block reward (premine) of $42,000,000$ KMWs, which is about 2% of the total. The initial block reward is $1,000$ KMWs, and the average block interval is 60 seconds. The block reward is cut in half every 2 years, which is equivalent to halving every $1,048,320$ blocks. Detailed parameters are as follows:

Token Symbol	KMW
Max KMW Supply	2,138,639,000 KMW
Premine	2% as the genesis block reward
Block Interval	60 seconds in average
Initial Block Reward	1,000 KMWs/block
Block Reward Halving	every 2 years, or every 1,048,320 blocks

According to the halving rule, after the 40th halving, the block reward will be less than ($1nano$) KMW. At that time, the block reward will be directly set to $1nano$ KMW, and the total mining rewards for a whole century will be only 0.052416 KMW.

6 Future Work

Kepler's first version of the Confidential Assets solution has been released in January 2020, including features such as Confidential Transaction, essential confidential assets technologies(generating asset tags, blinding asset tags), and user-defined confidential assets issuance.

In the future, Asset Surjection Proof(ASP) will be implemented to improve the privacy of assets types. Moreover, the privacy of confidential assets issued by users will be improved from the following aspects:

- The asset types and total supply can be hidden to achieve complete privacy;
- Support confidential assets deconstruction or re-issuance;
- Support optional confidential assets. Users can choose whether to hide their asset types and transaction amounts;
- Support non-interactive transactions. When the user initiates a transaction, the recipient is allowed to be offline, reducing the difficulty of the transaction operation.

Smart contracts technology on top of Kepler confidential assets is also a meaningful direction. Both Taproot and scriptless script are possible solutions to achieve this.

References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

2. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.
3. Mark Friedenbach and Jorge Timon. Freimarkets: extending bitcoin protocol with user-specified bearer instruments, peer-to-peer exchange, off-chain accounting, auctions, derivatives and transitive transactions. `http://freico.in/docs/freimarkets.pdf`, August 2013.
4. Colored Coins. `https://en.bitcoin.it/wiki/Colored_Coins`.
5. Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014. Accessed: 2016-08-22.
6. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07), 2017. Accessed: 2018-01-03.
7. Benedikt Bnz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. Cryptology ePrint Archive, Report 2019/191, 2019. `https://eprint.iacr.org/2019/191`.
8. Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. In *International Conference on Financial Cryptography and Data Security*, pages 43–63. Springer, 2018.
9. D. Ermilov, M. Panov, and Y. Yanovich. Automatic bitcoin address clustering. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 461–466, 2017.
10. The Dash project. `https://github.com/dashpay/dash`.
11. The Menero project. `https://github.com/monero-project/monero`.
12. The ZCash project. `https://github.com/zcash/zcash`.
13. Introduction to MimbleWimble and Grin. `https://github.com/mimblewimble/grin/blob/master/doc/intro.md`.
14. The AZTEC project. `https://github.com/AztecProtocol/AZTEC`.
15. The Grin project. `https://github.com/mimblewimble/grin`.
16. The Beam project. `https://github.com/BeamMW/beam`.
17. Yi Zheng, Howard Ye, Patrick Dai, Tongcheng Sun, and Vladislav Gelfer. Confidential assets on mimblewimble. Cryptology ePrint Archive, Report 2019/1435, 2019. `https://eprint.iacr.org/2019/1435`.
18. Eduardo Morais, Tommy Koens, Cees van Wijk, and Aleksei Koren. A survey on zero knowledge range proofs and applications, 2019.
19. Tarilabs mimblewimble introduction. `https://tlu.tarilabs.com/protocols/mimblewimble-1/MainReport.html`.
20. Hashing to elliptic curves. `https://tools.ietf.org/html/draft-irtf-cfrg-hash-to-curve-04`.
21. Riad S Wahby and Dan Boneh. Fast and simple constant-time hashing to the bls12-381 elliptic curve. *IACR Cryptology ePrint Archive*, 2019:403, 2019.
22. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.
23. Andrew Shallue and Christiaan E van de Woestijne. Construction of rational points on elliptic curves over finite fields. In *International Algorithmic Number Theory Symposium*, pages 510–524. Springer, 2006.
24. Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable hashing to barreto–naehrig curves. In *International Conference on Cryptology and Information Security in Latin America*, pages 1–17. Springer, 2012.

25. Gregory Maxwell and Andrew Poelstra. Borromean ring signatures, 2015.
26. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.