Confidential Assets on MimbleWimble

Yi Zheng[1], Howard Ye[1], Patrick Dai[1],
Tongcheng Sun[2], and Vladislav Gelfer[3]

[1] Qtum Chain Foundation, zhengyi@qtum.info
[2] Peking University, suntongcheng@pku.edu.cn
[3] Beam Foundation, valdo@beam.mw

January 16, 2020

Abstract. This paper proposes a solution for implementing Confidential Assets on MimbleWimble, which allows users to issue and transfer multiple assets on a blockchain without showing transaction addresses, amounts, and asset types. We first introduce the basic principles of MimbleWimble and then describe the implementation in detail.

Keywords: Confidential Assets · MimbleWimble · Blockchain privacy.

1 Introduction

MimbleWimble [1] is a blockchain solution that enables Confidential Transactions, where the transaction addresses and amounts are hidden, providing a high level of privacy for blockchain users. Validators in MimbleWimble only store Unspent Transaction Outputs (UTXOs) instead of the entire transaction history for the blockchain, enabling space savings and faster sync. The design of MimbleWimble relies on Elliptic Curve Cryptography, which is easy to understand and audit. Currently this solution is used in Grin [2] and Beam [3] projects.

The concept of Confidential Assets is from [4], 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 and fungibility of all assets. With Confidential Assets, users can issue their own assets on the blockchain for privacy-related applications. Until now, only the Elements [5] project has implemented it.

In this paper, we propose an implementation of Confidential Assets based on MimbleWimble. It can hide transaction addresses, amounts, and asset types, achieving the highest privacy. We first introduce the basic principles of MimbleWimble and then describe the implementation in detail.

2 MimbleWimble

This section presents the main parts of MimbleWimble as an introduction to our work. Other parts, like the transaction fee, multi-signature, transaction aggregation, cut-through, and details of Bulletproofs, are beyond the scope of this

paper. You can refer to Grin [2] and Beam [3] projects for more information about them.

2.1 Pedersen commitments

In MimbleWimble, each input or output of a transaction is expressed by a Pedersen commitment:

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

where C denotes the commitment, and a is the amount. G and H are two points randomly selected from the elliptic curve E and keep constant for all transactions. r is a private key chosen by the owner of the input/output, also called the blinding factor; it stands for the ownership, and makes the value of a hard to be calculated from the commitment. Through the Pedersen commitment, the amount and address of each input and output are hidden.

2.2 Signature

We can find that for a transaction with M inputs $\{C_{\text{in},m}\}_{m=1}^{M}$ and N outputs $\{C_{\text{out},n}\}_{n=1}^{N}$, the following equation holds:

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

in which

$$\sum_{n=1}^{N} a_{\text{out},n} - \sum_{m=1}^{M} a_{\text{in},m} = 0 \tag{3}$$

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

since a transaction would not create new funds. e is called the excess value.

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 transaction creator can generate a signature sig using e obtained through Eq. (4) as the private key. Here, the transaction creator refers to all senders and receivers of the transaction 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 creator proves to the verifier that:

- **Balance of amounts**. 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. (3) holds.
- **Ownership of inputs**. The transaction creator knows the private key e. Combined with Eq. (4) and Bulletproofs introduced below, it can be further proved that the sum input blinding factors $\sum_{m=1}^{M} r_{\text{in},m}$ are known.

2.3 Bulletproofs

Bulletproofs [6] are introduced for the transaction creator to demonstrate that all hidden output amounts are positive. Besides, they prove the ownership of inputs together with the signature.

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

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

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. Combined with Eq. (4) and the fact that the creator knows e demonstrated by the signature, it proves that the sum of input blinding factors $\sum_{m=1}^{M} r_{\text{in},m}$ are known to the creator; that is, the creator owns the inputs.

2.4 Putting it all together

Overall, a transaction in MimbleWimble consists of the following three parts:

$$\begin{aligned} \text{Input}: &\ C_{\text{in},1}, ..., C_{\text{in},M} \\ \text{Output}: &\ (C_{\text{out},1}, \pi_1), ..., (C_{\text{out},N}, \pi_N) \\ \text{Kernel}: &\ sig \end{aligned}$$

3 Confidential Assets

This section introduces the technology of Confidential Assets to MimbleWimble, which further allows users to issue and transfer multiple assets on the blockchain.

3.1 Mutiple assets

For Eq. (1), different assets can be assigned with different Hs. Suppose there are total I assets, the Pedersen commitment for the i-th asset is expressed as:

$$C_i = rG + aH_i \tag{6}$$

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

We should ensure that each asset tag is selected randomly, whose discrete logarithm is not known with respect to G and any other asset tags. So 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, which is detailed in the next subsection.

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 $\{C_{\text{in},i,m}\}_{m=1}^{M_i}$ and N_i outputs $\{C_{\text{out},i,n}\}_{n=1}^{N_i}$. Then the following equation holds:

$$\begin{aligned}
&\sum_{i=1}^{I}\sum_{n=1}^{N_i} C_{\text{out},i,n} - \sum_{i=1}^{I}\sum_{m=1}^{M_i} C_{\text{in},i,m} \\
&= \sum_{i=1}^{I}\sum_{n=1}^{N_i} (r_{\text{out},i,n}G + a_{\text{out},i,n}H_i) - \sum_{i=1}^{I}\sum_{m=1}^{M_i} (r_{\text{in},i,m}G + a_{\text{in},i,m}H_i) \\
&= \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r_{\text{out},i,n} - \sum_{m=1}^{M_i} r_{\text{in},i,m}\right)G + \sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} a_{\text{out},i,n} - \sum_{m=1}^{M_i} a_{\text{in},i,m}\right)H_i \\
&= eG
\end{aligned} \tag{7}$$

in which

$$\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} a_{\text{out},i,n} - \sum_{m=1}^{M_i} a_{\text{in},i,m}\right) = 0 \text{ for } i \in \{1, ..., I\} \tag{8}$$

$$\sum_{i=1}^{I}\left(\sum_{n=1}^{N_i} r_{\text{out},i,n} - \sum_{m=1}^{M_i} r_{\text{in},i,m}\right) = e \tag{9}$$

since no new funds would be created for each asset. 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.

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).

3.2 Generating asset tags

There has been a lot of research [7] 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 [8], 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 [9] called MapToGroup. It simply 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 [10] describe a mapping that is computable in deterministic polynomial time. Fouque and Tibouchi [11] give a concrete set of parameters for this mapping geared toward BN curves.

We use an extension of [11] 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. At last, 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:

$$H(m) = f(h_1(m)) + f(h_2(m)) \tag{10}$$

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.

3.3 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.

Signature On this basis, the Pedersen commitment for the i-th asset in Eq. (6) can be derived as:

$$\begin{aligned} C_i &= rG + aA_i \\ &= rG + a(H_i + sG) \\ &= (r+as)G + aH_i \end{aligned}$$

where $i \in \{1, ..., I\}$. It is equivalent to Eq. (6), because only the blinding factor r is replaced by another secret value $(r + as)$, and other parameters remain unchanged. With this replacement, Eq. (7) (8) (9) 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 actually 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) [4] 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 [12]. 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.

Now, 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.

In order 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 [13]. 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 $\mathcal{H}$.

3.4 Putting it all together

Overall, a transaction for Confidential Assets on MimbleWimble is shown as follows. Note that there are two types of Kernel to choose from: sig for ordinary transactions and (sig, sig', m) for asset issuances.

$$\begin{aligned} \text{Input}: &\ C_{\text{in},1}, ..., C_{\text{in},M} \\ \text{Output}: &\ (C_{\text{out},1}, \pi_1, A_1, \mathcal{H}_1, \sigma_1), ..., (C_{\text{out},N}, \pi_N, A_N, \mathcal{H}_N, \sigma_N) \\ \text{Kernel}: &\ sig \text{ or } (sig, sig', m) \end{aligned}$$

4 Future Work

We would like to realize some parts of the proposed solution and conduct several experiments on them to evaluate the performance. These parts may include generating asset tags, blinding asset tags, and ASP. The issuance query needs to be further designed. On the one hand, it can be considered to charge a certain amount of fee or lock some funds as a condition for issuing new assets. On the other hand, re-issuance and destruction mechanisms can be added to increase the functionality of issued assets.

References

1. Introduction to MimbleWimble and Grin. `https://github.com/mimblewimble/grin/blob/master/doc/intro.md`.
2. The Grin project. `https://github.com/mimblewimble/grin`.
3. The Beam project. `https://github.com/BeamMW/beam`.
4. 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.
5. The Elements project. `https://github.com/ElementsProject/elements`.
6. 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.
7. Hashing to elliptic curves. `https://tools.ietf.org/html/draft-irtf-cfrg-hash-to-curve-04`.
8. 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.
9. 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.
10. 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.
11. 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.

12. Gregory Maxwell and Andrew Poelstra. Borromean ring signatures, 2015.
13. 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.