TU WIEN Informatics

Adaptor Signature Based Atomic Swaps Between Bitcoin and a Mimblewimble Based Cryptocurrency

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Internet Computing

by

Jakob Abfalter, BSc
Registration Number 01126889

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. Matteo Maffei
Assistance: Dr. Pedro Moreno Sanchez

Vienna, 6th April, 2020

Jakob Abfalter	Matteo Maffei

Erklärung zur Verfassung der Arbeit

Jakob Abfalter, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 6. April 2020

Jakob Abfalter

Acknowledgements

Enter your text here.

Abstract

Enter your text here.

Contents

CHAPTER 1

Introduction

TODO

CHAPTER 2

Motivation & Objectives

TODO

CHAPTER 3

Preliminaries

3.1 Bitcoin

3.1.1 Bitcoin Transaction Protocol

3.1.2 Bitcoin Scaling and Layer Two Solutions

3.2 Privacy-enhancing Cryptocurrencies

3.2.1 Zero Knowledge Proofs

3.2.2 Range Proofs

3.2.3 Mimblewimble

The Mimblewimble protocol was introduced in 2016 by an anonymous entity named Jedusor, Tom Elvis [Jed16]. The author's name, as well as the protocols name, are references to the Harry Potter franchise. [1] In Harry Potter, Mimblewimble is a tongue-typing curse which reflects the goal of the protocol's design,which is improving the user's privacy. Later, Andrew Poelstra took up the ideas from the original writing and published his understanding of the protocol in his paper [Poe16]. The protocol gained increasing interest in the community and was implemented in the Grin [2] and Beam [3] Cryptocurrencies, which both launched in early 2019. In the same year, two papers were published, which successfully defined and proved security properties for Mimblewimble [FOS19, BCL+19]. In this section, we will explain the fundamental properties of the protocols which are relevant for this thesis. The section is ba
Compared to Bitcoin, there are some differences in Mimblewimble:

[1] https://harrypotter.fandom.com/wiki/Tongue-Tying_Curse
[2] https://grin.mw/
[3] https://beam.mw/

- Use of Pedersen commitments instead of plaintext transaction values
- No addresses. Coin ownership is given by the knowledge of the opening of the coins Pedersen commitment.
- Spend outputs are purged from the ledger such that only unspent transaction outputs remain.
- No scripting features.

By utilizing Pedersen commitments in the transactions, we hide the amounts transferred in a transaction, improving the systems user privacy, but also requiring additional range proofs, attesting to the fact that actual amounts transferred are in between a valid range. Not having any addresses enables transaction merging and transaction cut through, which we will explain a bit later. However, this comes with the consequence that building transactions require active interaction between the sender and receiver, which is different than in constructions more similar to Bitcoin, where a sender can transfer funds to any address without requiring active participation by the receiver.
Through transaction merging and cut-through and some further protocol features, which we will see later in this section, we gain the third mentioned property of being able to delete transaction outputs from the Blockchain, which have already been spent before. This ongoing purging in the Blockchain makes it particularly space-efficient as the space required by the ledger only grows in the number of UTXOs, in contrast to Bitcoin, in which space requirement increases with the number of overall mined transactions. Saving space is especially relevant for Cryptocurrencies employing confidential transactions because the size of the range proofs attached to outputs can be significant. Another advantage of this property is that new nodes joining the system do not have to verify the whole history of the Blockchain to validate the current state, making it much easier to join the network.
Another limitation of Mimblewimble- based Cryptocurrencies is that at least the current construction does not allow scripts, such as they are available in Bitcoin or similar systems. Transaction validity is given solely by a single valid signature plus the balancedness of inputs and outputs. This shortcoming makes it challenging to realize concepts such as multi signatures or conditional transactions which are required for Atomic Swap protocols. However, as we will see in 3.3 there are ways we can still construct the necessary transactions by merely relying on cryptographic primitives [FOS19].

Transaction Structure

- For two adjacent elliptic curve generators g and h a coin in Mimblewimble is of the form $C := g^v + h^r$, π. C is a so called Pedersen Commitment [Ped91] to the value v with blinding factor r. π is a range proof attesting to the fact that v is in a valid range in zero-knowledge.

- As already pointed out, there are now addresses in Mimblewimble. Ownership of a coin is equivalent to the knowledge of its opening, so the blinding factor takes the role of the secret key.
- A transaction consists of $C_{inp} := (C_1, \ldots, C_n)$ input coins and $C_{out} := (C'_1, \ldots, C'_n)$ output coins.

A transaction is considered valid iff $\sum v'_i - \sum v_i = 0$ so the sum of all input values has to be 0. (Not taking transaction fees into account)
From that we can derive the following equation:

$$\sum C_{out} - \sum C_{inp} := \sum h^{v'_i} + g^{r'_i} - \sum h^{v_i} + g^{r_i}$$

So if we assume that a transaction is valid then we are left with the following so called excess value:

$$E := g^{(\sum r'_i - \sum r_i)}$$

Knowledge of the opening of all coins and the validity of the transaction implies knowledge of E. Directly revealing the opening to E would leak too much information, an adversary knowing the openings for input coins and all but one output coin, could easily calculate the unknown opening given E. Therefore knowledge of E instead is proven by providing a valid signature for E as public key. Coinbase transactions (transactions creating new money as part of a miners reward) additionally include the newly minted money as supply s in the excess equation:

$$E := g^{(\sum r'_i - \sum r_i)} - h^s$$

Finally a Mimblewimble transaction is of form:

$$tx := (s, C_{inp}, C_{out}, K) \text{ with } K := (l(\pi), l(E), l(\sigma))$$

where s is the transaction supply amount, C_{inp} is the list of input coins, C_{out} is the list of output coins and K is the transaction Kernel. The Kernel consists of $l(\pi)$ which is a list of all output coin range proofs, $l(E)$ a list of excess values and finally $l(\sigma)$ a list of signatures [FOS19].

Transaction Merging

An essential property of the Mimblewimble protocol is that two transactions can easily be merged into one, which is essentially a non-interactive version of the CoinJoin protocol on Bitcoin [Max13] Assume we have the following two transactions:

$$tx_0 := (s_0, C^0_{inp}, C^0_{out}, (l(\pi_0), l(E_0), l(\sigma_0)))$$

$$tx_1 := (s_1, C^1_{inp}, C^1_{out}, (l(\pi_1), l(E_1), l(\sigma_1)))$$

Then we can build a single merged transaction:

$$tx_m \;:=\; (s_0 + s_1,\; C^0_{inp} \,||\, C^1_{inp},\; C^0_{out} \,||\, C^1_{out},\; (l(\pi_0) \,||\, l(\pi_1)),\; l(E_0) \,||\, l(E_1),\; l(\sigma_0) \,||\, l(\sigma_1))$$

We can easily deduce that if tx_0 and tx_1 are valid, it follows that tx_m also has to be valid: If tx_0 and tx_1 are valid that means $C^0_{inp} - C^0_{out} - h^{s_0} := E_0$, $l(\pi_0)$ contains valid range proofs for the outputs C^0_{out} and $l(\sigma_0)$ contains a valid signature to $E_0 - h^{s_0}$ as public key, the same must hold for tx_1.
By the rules of arithmetic it then must also hold that

$$C^0_{inp} \,||\, C^1_{inp} \;-\; C^0_{out} \,||\, C^1_{out} \;-\; h^{s_0 + s_1} \;:=\; E_0 \;+\; E_1,\; l(\pi_0) \,||\, l(\pi_1)$$

must contain valid range proofs for the output coins and $l(\sigma_0) \,||\, l(\sigma_1)$ must contain valid signatures to the respective Excess points, which makes tx_m a valid transaction.

Subset Problem

A subtle problem arises with the way transactions are merged in Mimblewimble. From the shown construction, it is possible to reconstruct the original separate transactions from the merged one, which can be a privacy issue. Given a set of inputs, outputs, and kernels, a subset of these will recombine to reconstruct one of the valid transaction which were aggregated since Kernel Excess values are not combined. (which would invalidate the signatures and therefore break the security of the system) This problem has been mitigated in Cryptocurrencies implementing the protocol by including an additional variable in the Kernel, called offset value. The offset is randomly chosen and needs to be added back to the Excess values to verify the sum of the commitments to zero.

$$\sum C_{out} \;-\; \sum C_{inp} \;-\; h^s \;:=\; E \;+\; o$$

Every time two transactions are merged, the offset values are combined into a single value. If offsets are picked truly randomly, and the possible range of values is broad enough, the probability of recovering the uncombined offsets from a merged one becomes negligible, making it infeasable to recover original transactions from a merged one [Poe16].

Cut Through

From the way transactions are merged together, we can now learn how to purge spent outputs securely. Let's assume C_i appears as an output in tx_0 and as an input in tx_1, which are being merged. Remembering the equation for transaction balancedness, $C_{inp} - C_{out} := E$ if C_i appears both in the inputs and outputs, and we erase it on both sides, the equation will still hold. Therefore every time a transaction spends an output, it can be virtually forgotten to improve transaction unlinkability as well as yielding saving space.

The Ledger

The ledger of the Mimblewimble protocol itself is a transaction of the already discussed form. Initially, the ledger starts empty, and transactions are added and aggregated recursively.

- Only transactions in which input coins are contained in the output coins of the ledger will be valid.
- The supply of the ledger is the sum of the supplies of all transactions added so far. Therefore we can easily read the total circulating supply from the ledger state.
- Due to cut through, the input coin list of the ledger is always empty, and the output list is the set of UTXOs.

Transaction Building

As already pointed out, building transactions in Mimblewimble is an interactive process between the sender and receiver of funds. Jedusor, Tom Elvis originally envisioned the following two-step process to build a transaction: [Jed16]
Assume Alice wants to transfer coins of value p to Bob.

1. Alice first selects input coins C_{inp} of total value $v \geq p$ that she controls. She than creates change coin outputs C^A_{out} (could be multiple) of total value $v - p$ and then sends $C_{inp},\ C^A_{out}$, a valid range proofs for C^A_{out}, plus the opening $(-p,\ k)$ of $\sum C^A_{out} - \sum C_{inp}$ to Bob.
2. Bob creates himself additional output coins C^B_{out} plus range proofs of total value p with keys (k'_i) and computes a signature σ with the combined secret key $k + \sum k'_i$ and and finalizes the transaction as

   $$tx := (0,\ C_{inp},\ C^A_{out} \mid\mid C^B_{out},\ (\pi,\ E := \sum C^A_{out} + \sum C^B_{out} - \sum C_{inp},\ \sigma))$$

   and publishes it to the network.

Figure shows this original transaction flow.
This protocol however turned out to be vulnerable. The receiver can spend the change coins C^A_{out} by reverting the transaction. Doing this would give the sender his coins back, however as the sender might not have the keys for his spent outputs anymore, the coins could then be lost.
In detail this reverting transaction would look like:

$$tx_{rv} := (0,\ C^A_{out} \mid\mid C^B_{out},\ C_{inp},\ (\pi_{rv},\ E_{rv},\ \sigma_{rv}))$$

Again remembering the construction of the Excess value of this construction would look like this:

$$E_{rv} := \sum C^A_{out} \mid\mid C^B_{out} - C_{inp}$$

The key k originally sent by Alice to Bob is a valid opening to $\sum C_{inp} - \sum C^A_{out}$. With the inverse of this key k_{inv} we get the opening to $\sum C^A_{out} - C_{inp}$. Now all Bob has to do is add his keys $\sum k'_i$ to get:

$$k_{rv} := -k + \sum k'_i$$

which is the opening to E_{rv}. Furthermore obtaining a valid range proofs is trivial, as it once was a valid output the ledger will cointain a valid proof for this coin already.
This means Bob spends the newly created outputs and sends them back to the original input coins, chosen by Alice. It might at first seem unclear why Bob would do that. An example situation could be if Alice pays Bob for some good which Bob is selling. Alice decides to pay in advance, but then Bob discovers that he is already out of stock of the good that Alice ordered. To return the funds to Alice, he reverses the transaction instead of participating in another interactive process to build a new transaction with new outputs. If Alice already deleted the keys to her initial coins, the funds are now lost. The problem was solved in the Grin Cryptocurrency by making the signing process itself a two-party process which will be explained in more detail in chapter 4.

Fuchsbauer et al. [FOS19] proposed the following alternative way to build transactions which would not be vulnerable to this problem.

1. Alice constructs a full-fledged transaction tx_A spending her input coins C_{inp} and creates her change coins C^A_{out}, plus a special output coin $C^{sp}_{out} := h^p + g^{k_{sp}}$, where p is the desired value which should be transferred to Bob and k_{sp} is a randomly choosen key. She proceeds by sending tx_A as well as $(p,\ k_{sp})$ and the necessary range proofs to Bob.

2. Bob now creates a second transaction tx_B spending the special coin C^{sp}_{out} to create an output only he controls C^B_{out} and merges tx_A with tx_B into tx_m. He then broadcasts tx_m to the network. Note that when the two transactions are merged the intermediate special coin C^{sp}_{out} will be both in the coin output and input list of the transaction and therfore will be discarded.

The only drawback of this approach is that we have two transaction kernels instead of just one because of the merging step, making the transaction slightly bigger.

3.3 Scriptless Scripts

3.4 Adaptor Signatures

3.4.1 Schnorr Signature Construction

3.4.2 ECDSA Signature Construction

CHAPTER 4

Mulitparty Fixed Witness Adaptor Signatures

4.1 General Notation

4.2 Cryptographic Primitives

4.3 Mulitparty Fixed Witness Adaptor Signature Scheme

We define a Generalized Multiparty Adaptor Signature Scheme from the standard construction of multiparty Schnorr signatures which are defined as follows:

$\underline{GEN()}$

1 : $k \leftarrow\$ \mathbb{Z}_q$
2 : $r \leftarrow\$ \mathbb{Z}_q$
3 : **return** $(k,\ r)$

$\underline{GEN\_PT\_SIG(m,\ k,\ r,\ g^{k'},\ g^{r'})}$

1 : $e\ :=\ h((m\ ||\ g^k\ +\ g^{k'}\ ||\ g^r\ +\ g^{r'}))$
2 : $\sigma_{prt}\ :=\ k\ +\ e\ +\ r$
3 : **return** $(\sigma_{prt},\ g^k,\ g^r)$

$\underline{VRF\_PT\_SIG(m,\ k,\ r,\ g^{k'},\ g^{r'},\ \sigma_{prt})}$

1 : $e\ :=\ h(((m\ ||\ g^k\ +\ g^{k'}\ ||\ g^r\ +\ g^{r'}))$
2 : **return** $g^{\sigma_{prt}}\ =\ g^{k'}\ +\ g^{e\ *\ r}$

$\underline{FIN\_SIG(\sigma_{prt},\ \sigma'_{prt},\ g^k,\ g^{k'},\ g^r,\ g^{r'})}$

1 : **return** $(\sigma_{prt}\ +\ \sigma'_{prt},\ g^k\ +\ g^{k'},\ g^r\ +\ g^{r'})$

In order to have adaptable partial signature we add the following procedures

$\underline{APT\_PT\_SIG(\sigma_{prt},\ x)}$

1 : $\sigma_{prt_apt} := \sigma_{prt} + x$
2 : **return** $(\sigma_{prt_apt},\ g^x)$

$\underline{EXT\_WIT(\sigma_{fin},\ \sigma_{prt},\ \sigma'_{prt_apt})}$

1 : $\sigma'_{prt} := \sigma_{fin} - \sigma_{prt}$
2 : $x := \sigma'_{prt_apt} - \sigma'_{prt}$
3 : **return** (x)

$\underline{VRF\_APT\_SIG(m,\ k,\ r,\ g^{k'},\ g^{r'},\ g^x,\ \sigma'_{prt_apt})}$

1 : $e := h(m \mid\mid g^k + g^{k'} \mid\mid g^r + g^{r'})$
2 : **return** $g^{\sigma'_{prt_apt}} = g^{k'} + g^{e * r} + g^x$

CHAPTER 5

Adaptor Signature Based Atomic Swaps Between Bitcoin and a Mimblewimble Based Cryptocurrency

CHAPTER 6

Implementation

CHAPTER 7

Implementation Security and Privacy Evaluation

CHAPTER 8

Related and Future Work

CHAPTER 9

Conclusion

List of Figures

List of Tables

List of Algorithms

Bibliography

[BCL+19] Gustavo Betarte, Maximiliano Cristiá, Carlos Luna, Adrián Silveira, and Dante Zanarini. Towards a formally verified implementation of the mimblewimble cryptocurrency protocol. *arXiv preprint arXiv:1907.01688*, 2019.

[FOS19] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: a cryptographic investigation of mimblewimble. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 657–689. Springer, 2019.

[Jed16] Tom Elvis Jedusor. Mimblewimble, 2016.

[Max13] Greg Maxwell. Coinjoin: Bitcoin privacy for the real world. In *Post on Bitcoin forum*, 2013.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[Poe16] Andrew Poelstra. Mimblewimble, 2016.