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

Abstract **vii**

Contents **ix**

1 Introduction **1**

2 Motivation & Objectives **5**

3 Preliminaries **7**
3.1 General Notation and Definitions 7
3.2 Bitcoin 10
3.3 Privacy-enhancing Cryptocurrencies 10
3.4 Scriptless Scripts 15
3.5 Adaptor Signatures 15

4 Two Party Fixed Witness Adaptor Signatures **17**
4.1 Definitions 17
4.2 Schnorr-based instantiation 21
4.3 Security 22

List of Figures **27**

List of Tables **29**

List of Algorithms **31**

Bibliography **33**

CHAPTER 1

Introduction

Pedro: We need to discuss a structure for the introduction. Proposal:
- Introduce why coin exchanges are interesting
- Explain why atomic swaps protocols (e.g., one could use a trusted server for this and problem solved, right?)
- Why coin exchanges between Bitcoin and Mimblewimble?
- Why what you are proposing in this thesis is challenging?
- What are the main contributions of these thesis?
- What do you think is an interesting future research direction?

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].

Pedro: I would not add a line break at the end of each paragraph. The template should do that

Pedro: If you are going to compare to Bitcoin, you need to introduce Bitcoin before

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

Compared to Bitcoin, there are some differences in Mimblewimble:

- Use of Pedersen commitments instead of plaintext transaction values

  Pedro: The reader does not know what Pedersen commitments are at this point. Perhaps say transaction values are hidden from a blockchain observer while this is not the case in Bitcoin

- No addresses. Coin ownership is given by the knowledge of the opening of the coins Pedersen commitment.

  Pedro: This is also unclear. Could one see the commitment as the "address" in Mimblewimble? Perhaps you want to say that there is no scripting language supported?

- Spend outputs are purged from the ledger such that only unspent transaction outputs remain.

- No scripting features.

Pedro: Use "we" for contributions that you do in the thesis and "they" for parts that are borrowed from other works

Pedro: An intuition of these two terms is required here

Pedro: another sentence that shows that you need to explain before how Bitcoin works (the basics)

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 in section 3.3.3. 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.

Pedro: What comes next is hard to read. It requires better organization: Advantages of Mimblewimble are: (i) .., (ii)...; Disadvantages are: (i)..., (ii),...).

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 3.4 there are ways we can still construct the necessary transactions by merely relying on cryptographic primitives [FOS19].

CHAPTER 2

Motivation & Objectives

TODO

CHAPTER 3

Preliminaries

Pedro: Although not strictly required, IMO it is nice to have some text here introducing what the reader should expect in the rest of the section. For instance:
In this section, we first introduce the notation and definitions used hereby in this thesis. Then, we ..... Finally, we introduce.....

3.1 General Notation and Definitions

Notation We first define the general notation used in the following chapters to formalize procedures and protocols. Let $\mathbb{G}$ denote a cyclic group of prime order p and $\mathbb{Z}_p$ the ring of integers modulo p. $\mathbb{Z}_p^*$ is $\mathbb{Z}_p \setminus \{0\}$. g, h are adjacent generators in $\mathbb{G}$, where adjacent means the discrete logarithm of h in regards to g is not known. Exponentiation stands for repeated application of the group operation. We define the group operation between two curve points as $g^a \cdot g^{g^b} = g^{a+b}$.

Pedro: I think macro \ was broken here. I have updated to use \ instead. Please check that this is what you expected

Definition 3.1 (Hard Relation[AEE+20])**.** Given a language $L_R := \{A \mid \exists a \text{ s.t. } (A, a) \in R\}$ then the relation R is considered hard if the following three properties hold:

1. $\mathsf{genRel}((1^n))$ is a PPT sampling algorithm which outputs a statement/witness of the form $(A, a) \in R$.
2. Relation R is poly-time decidable.
3. For all PPT adversaries $\mathcal{A}$ the probability of finding a given A is negligible.

Pedro: We normally do not use the tilde to add spaces in math mode

Pedro: I would include these two relations below as your own definitions because I imagine that you would like to refer to them afterwards in the thesis

In this thesis we find two types of hard relations:

1. The output of a secure hash function (as defined in 3.3) and it's input $(I, \mathsf{H}(I))$.
2. The discrete logarithm x of g^x in the group $\mathbb{G}$.

Pedro: Link to paper/book where you got this definition from is missing

Pedro: What is valid here? You have not defined it before

Definition 3.2 (Signature Scheme)**.** A valid Signature Scheme must provide three procedures:

Pedro: I would write this sentence as: A signature scheme Φ is a tuple of algorithms (setup, sign, verf) defined as follows:

$$\Phi = (\mathsf{setup},\ \mathsf{sign},\ \mathsf{verf})$$

Pedro: write the API of the algorithms in bullet points

$(sk, pk) \leftarrow \mathsf{setup}(1^n)$ takes as input a security parameter 1^n and outputs a keypair (sk, pk), consisting of a secret key sk and a public key pk, whereas the secret key has to be kept private and the public key is shared with other parties. sk can be used together with a message m to call the $\mathsf{sign}(sk,\ m)$ procedure to create a signature σ over the message m. Parties knowing pk can then test the validity of the signature by calling $\mathsf{verf}(pk,\ \sigma,\ m)$ with the same message m. The procedure will only output 1 if the message was indeed signed with the correct secret key sk of pk and therefore proves the possesion of sk by the signer . A valid signature scheme have to fulfill two security properties

Pedro: "proving that the sender had the sk" is a property that no all signature schemes may have

Pedro: Choose one unforgeability form, the one that you require later in the thesis

- Correctness: For all messages m and valid keypairs (sk, pk) the following must hold $\mathsf{verf}(pk,\ \mathsf{sign}(sk,\ m),\ m) = 1$
- Unforgeability: Note that there are different levels of Unforgability: [GMR88]
  - Universal Forgery: The ability to forge signatures for any message.
  - Selective Forgery: The ability to fogre signatures for messages of the adversary's choice.
  - Existential Forgery: The ability to forge a valid signature / message pair not previously known to the adversary.

Pedro: Except with negligible probability

Pedro: nice that you have defined the three properties. I would keep the one that you need later

minor
we normally
ital H

Definition 3.3 (Cryptographic Hash Function)**.** A cryptographic hash function H is defined as $\mathsf{H}(I) \rightarrow \{0,1\}^n$ for some fixed number n and some input I. A secure hashing function has to fullfill the following security properties: [AKDB11]

- Collision-Resistence (CR): Collision-Resistance means that it is computationally infeasible to find two inputs I_1 and I_2 such that $\mathsf{H}(I_1) := \mathsf{H}(I_2)$ with $I_1 \neq I_2$.
- Pre-image Resistence (Pre): In a hash function H that fulfills Pre-image Resistance it is infeasible to recover the original input I from its hash output $\mathsf{H}(I)$. If this security property is achieved, the hash function is said to be non-invertible.
- 2nd Pre-image Resistence (Sec): This property is similar to Collision-Resistance and is sometimes referred to as *Weak Collision-Resistance*. Given such a hash function H and an input I, it should be infeasible to find a different input I' such that $I \neq I'$ and $\mathsf{H}(I) = \mathsf{H}(I')$.

Pedro: I think here the Open operation of the commitment is missing, which you need later for the binding property

Definition 3.4 (Commitment Scheme [BBB+18])**.** A cryptographic Commitment is defined by a pair of functions $(\mathsf{setup}(1^n), \mathsf{commit}(I,k))$. setup is the setup procedure, it takes as input a security parameter 1^n and outputs public parameters PP. Depending on PP we define a input space $\mathbb{I}_{PP}$, a randomness space $\mathbb{K}_{PP}$ and a commitment space $\mathbb{C}_{PP}$.

The function commit takes an arbitrary input $I \in \mathbb{I}_{PP}$, and a random value $k \in \mathbb{K}_{PP}$ and generates an output $C \in \mathbb{C}_{PP}$.

Secure commitments must fullfill the *Binding* and *Hiding* security properties:

- *Binding:* If a Commitment Scheme is binding it must hold that for all PPT adversaries $\mathcal{A}$ given a valid input $I \in \mathbb{I}_{PP}$ and randomness $k \in \mathbb{K}_{PP}$ the probabilty of finding a $I' \neq I$ and a k' with $\mathsf{commit}(I,k) = \mathsf{commit}(I',k')$ is negligible.
- *Hiding:* For a PPT adversary $\mathcal{A}$, commitment inputs $I \in \mathbb{I}_{PP}$, $k \in \mathbb{K}_{PP}$ and a commitment output $C := \mathsf{commit}(I,k,)$ the probabilty of the adversary choosing the correct input $\{I, I'\}$ must not be higher then $\frac{1}{2} + \mathsf{negl}(P)$.

Pedro: this definition seems wrong to me? Where does I' come from?

Pedro: Add reference from where you took this definition. You may want to add that it is an "Additive" Homomorphic Commitment

Definition 3.5 (Homomorphic Commitment)**.** If a Commitment Scheme as defined in 3.4 is homomorphic then the following must hold

$$\mathsf{commit}(I_1,k_1) \cdot \mathsf{commit}(I_2,k_2) = \mathsf{commit}(I_1 + I_2, k_1 + k_2)$$

First, a Pedersen Commitment is an instance of Commitment Scheme as in Def 3.4; Second it has the homomorphic property as in 3.5. Clarify that, for instance, by explaining exactly how the algorithms are implemented, as you did below.

Definition 3.6 (Pedersen Commitment)**.** A Pedersen Commitment is an instantiation of a Homomoprhic Commitment Scheme as defined in 3.5:

$$\mathbb{C}_{PP} := \mathbb{G}$$

of order p, $\mathbb{I}_{PP}$, $\mathbb{K}_{PP} := \mathbb{Z}_p$. the procedures $(\mathsf{setup}, \mathsf{commit})$ are then instantiated as:

$$\mathsf{setup}(1^n) := g, h \leftarrow_\$ \mathbb{G}$$

$$\mathsf{commit}(I, k) := g^k h^I$$

3.2 Bitcoin

3.2.1 Bitcoin Transaction Protocol

3.2.2 Bitcoin Scaling and Layer Two Solutions

3.3 Privacy-enhancing Cryptocurrencies

3.3.1 Zero Knowledge Proofs

3.3.2 Range Proofs

3.3.3 Mimblewimble

In this section we will outline the fundamental properties of the protocols employed in Mimblewimble which are relevant for the thesis and particularily the construction of the Atomic Swap protocol defined in **??**.

Transaction Structure

Pedro: I think that throughout this section, you have nice explanations of the different parts of the transaction. It would be also possible to add definitions for the different things that you use

- For two adjacent elliptic curve generators g and h a coin in Mimblewimble is a tuple of the form $(\mathcal{C}, \pi)$, where $\mathcal{C} := g^v \cdot h^k$ a Pedersen Commitment [Ped91] to the value v with blinding factor k. π is a range proof attesting to the fact that v is in a valid range in zero-knowledge.

  Pedro: not sure whether the point below is required

Pedro: you might want to specify what range is used here. Also I rewrote some part, so please check.

- 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 $\mathcal{C}_{inp} := (\mathcal{C}_1, \ldots, \mathcal{C}_n)$ input coins and $\mathcal{C}_{out} := (\mathcal{C}'_1, \ldots, \mathcal{C}'_n)$ 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)

Pedro: doesn't need to check the range proofs as well?

From that we can derive the following equation:

$$\sum \mathcal{C}_{out} - \sum \mathcal{C}_{inp} := \sum (h^{v'_i} \cdot g^{k'_i}) - \sum (h^{v_i} \cdot g^{k_i})$$

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

$$\mathcal{E} := g^{(\sum k'_i - \sum k_i)}$$

Knowledge of the opening of all coins and the validity of the transaction implies knowledge of $\mathcal{E}$. Directly revealing the opening to $\mathcal{E}$ 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 $\mathcal{E}$. Therefore knowledge of $\mathcal{E}$ instead is proven by providing a valid signature for $\mathcal{E}$ 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:

Pedro: You mean knowledge of the exponent of $\mathcal{E}$?

$$\mathcal{E} := g^{(\sum k'_i - \sum k_i)} - h^s$$

Finally a Mimblewimble transaction is of form:

$$tx := (s, \mathcal{C}_{inp}, \mathcal{C}_{out}, K) \text{ with } K := (\{\pi\}, \{\mathcal{E}\}, \{\sigma\})$$

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

Pedro: the $\mathcal{E}$ is a single value? or a set?

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:

$$tx_0 := (s_0, \mathcal{C}^0_{inp}, \mathcal{C}^0_{out}, (\{\pi_0\}, \{\mathcal{E}_0\}, \{\sigma_0\}))$$

$$tx_1 := (s_1,\ \mathcal{C}^1_{inp},\ \mathcal{C}^1_{out},\ (\{\pi_1\},\ \{\mathcal{E}_1\},\ \{\sigma_1\}))$$

Then we can build a single merged transaction:

$$tx_m := (s_0 + s_1,\ \mathcal{C}^0_{inp} \,||\, \mathcal{C}^1_{inp},\ \mathcal{C}^0_{out} \,||\, \mathcal{C}^1_{out},\ (\{\pi_0\} \,||\, \{\pi_1\}),\ \{\mathcal{E}_0\} \,||\, \{\mathcal{E}_1\},\ \{\sigma_0\} \,||\, \{\sigma_1\})$$

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 $\mathcal{C}^0_{inp} - \mathcal{C}^0_{out} - h^{s_0} := \mathcal{E}_0$, $\{\pi_0\}$ contains valid range proofs for the outputs $\mathcal{C}^0_{out}$ and $\{\sigma_0\}$ contains a valid signature to $\mathcal{E}_0 - h^{s_0}$ as public key, the same must hold for tx_1.

By the rules of arithmetic it then must also hold that

$$\mathcal{C}^0_{inp} \,||\, \mathcal{C}^1_{inp} - \mathcal{C}^0_{out} \,||\, \mathcal{C}^1_{out} - h^{s_0 + s_1} := \mathcal{E}_0 \cdot \mathcal{E}_1,\ \{\pi_0\} \,||\, \{\pi_1\}$$

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

Subset Problem

Pedro: I think the content below is not fully clear yet. If needed for the rest, we need to clarify (e.g., add an example?)

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.

$$\sum \mathcal{C}_{out} - \sum \mathcal{C}_{inp} - h^s := \mathcal{E}^o$$

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

Pedro: This requires further explanation and maybe an example?

From the way transactions are merged together, we can now learn how to purge spent outputs securely. Let's assume $\mathcal{C}_i$ appears as an output in tx_0 and as an input in tx_1, which are being merged. Remembering the equation for transaction balancedness, $\mathcal{C}_{inp} - \mathcal{C}_{out} := \mathcal{E}$ if $\mathcal{C}_i$ 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

Pedro: Do we need this subsection?

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 $\mathcal{C}_{inp}$ of total value $v \geq p$ that she controls. She then creates change coin outputs $\mathcal{C}^A_{out}$ (could be multiple) of total value $v - p$ and then sends $\mathcal{C}_{inp}$, $\mathcal{C}^A_{out}$, a valid range proofs for $\mathcal{C}^A_{out}$, plus the opening $(-p,\ x)$ of $\sum \mathcal{C}^A_{out} - \sum \mathcal{C}_{inp}$ to Bob.

   Pedro: we need a way to express this clearer

2. Bob creates himself additional output coins $\mathcal{C}^B_{out}$ plus range proofs of total value p with keys (x'_i) and computes a signature σ with the combined secret key $x + \sum x'_i$ and finalizes the transaction as

   $$tx := (0,\ \mathcal{C}_{inp},\ \mathcal{C}^A_{out} \,||\, \mathcal{C}^B_{out},\ (\pi,\ \mathcal{E} := \sum \mathcal{C}^A_{out} \cdot \sum \mathcal{C}^B_{out} - \sum \mathcal{C}_{inp},\ \sigma))$$

   and publishes it to the network.

Figure 3.1 depicts the original transaction flow.
This protocol however turned out to be vulnerable. The receiver can spend the change coins $\mathcal{C}^A_{out}$ 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.

Pedro: For security, privacy or both?

In detail this reverting transaction would look like:

$$tx_{rv} := (0,\ \mathcal{C}^A_{out} \,||\, \mathcal{C}^B_{out},\ \mathcal{C}_{inp},\ (\pi_{rv},\ \mathcal{E}_{rv},\ \sigma_{rv}))$$

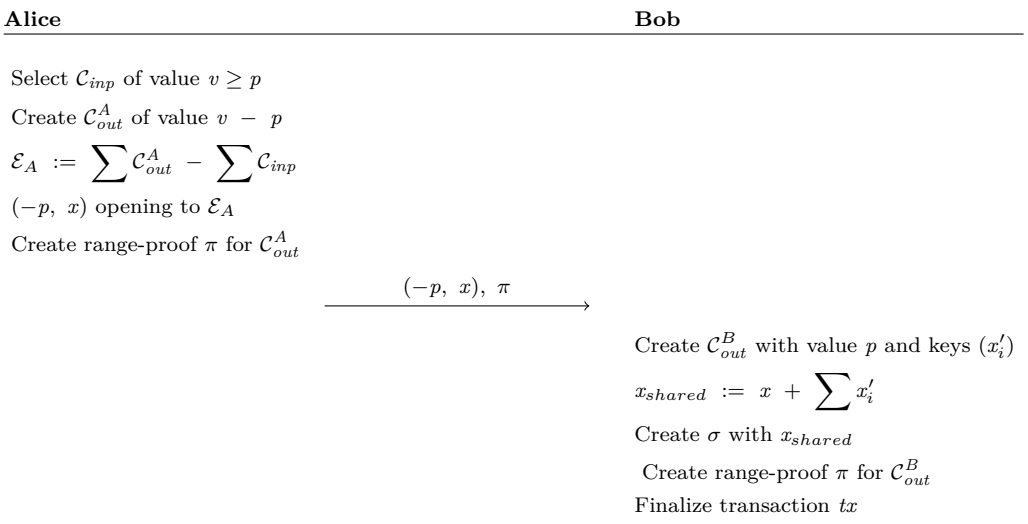

Figure 3.1: Original transaction building process

Pedro: Really nice that you created this protocol :)

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

$$\mathcal{E}_{rv} := \sum \mathcal{C}^A_{out} \,||\, \mathcal{C}^B_{out} - \mathcal{C}_{inp}$$

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

$$x_{rv} := -x + \sum x'_i$$

Pedro: Why range proof is not correct here in the first place?

which is the opening to $\mathcal{E}_{rv}$. 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.

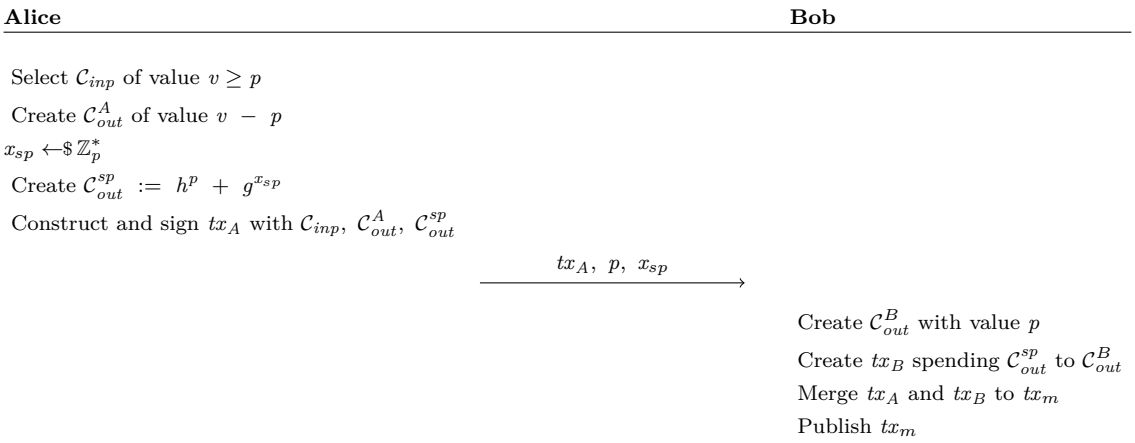

Figure 3.2: Salvaged tranction protocol by Fuchsbauer et al. [FOS19]

Pedro: put labels inside captions

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

2. Bob now creates a second transaction tx_B spending the special coin $\mathcal{C}^{sp}_{out}$ to create an output only he controls $\mathcal{C}^B_{out}$ 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 $\mathcal{C}^{sp}_{out}$ 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. A figure showing the protocol flow is depicted in Figure 3.2.

3.4 Scriptless Scripts

3.5 Adaptor Signatures

3.5.1 Schnorr Signature Construction

3.5.2 ECDSA Signature Construction

CHAPTER 4

Two Party Fixed Witness Adaptor Signatures

In this chapter, we will define a variant of the adaptor signature scheme as explained in 3.5, which is specifically tailored for the use in an Atomic Swap scenario in which (at least one side of the swap) uses a two-party protocol to generate transaction signatures and keypairs are fixed beforehand.

First we will define the general two-party signature creation protocol as it is currently implemented for example in the Grin cryptocurrency. We reduce the generated signatures to the general case [Sch89] and thereby prove its correctness. From this protocol, we then derive the adapted variant, which allows hiding a fixed witness value in the signature, which can be revealed only by the other party after attaining the final signature. We start by defining our extended signature scheme in section 4.1, proceed by providing a schnorr-based instantiation of the protocol in section 4.2 and finally prove its security in section 4.3.

4.1 Definitions

Definition 4.1 (Two Party Signature Scheme)**.** A two-party signature scheme wrt. a hard relation R is an extension of a signature scheme as defined in definition 3.2, which allows us to distribute signature generation for a composite public key shared between two parties Alice and Bob. Alice and Bob want to collaborate to generate a signature valid under the composite public key $pk_C := pk_A + pk_B$ without having to reveal their secret keys to each other. For this we add three procedures to our signature scheme definition:

$$\Phi_{MP} = (\Phi \mid\mid \mathsf{signPt},\ \mathsf{vrfPt},\ \mathsf{finSig})$$

- signPt is defined as a two-party procedure which requires both Alice and Bob to provide their secret key and a message which should be signed. The parties collaborate to both calculate their own partial signatures. These partial signatures do not need to be valid signatures yet, but the other participating parties need to be able to verify them using the vrfPt procedure.

- vrfPt Is a function which lets one party verify the partial signature of the other party. As input it requires the partial signature, signed message and the participants public keys. The output will be either 1 if the verification was successfull or 0 otherwise.

- finSig will take the two partial signatures and combine them into a final valid signature under the participants composite public key.

Definition 4.2 (Two Party Fixed Witness Adaptor Schnorr Signature Scheme)**.** From the definition 4.1, we now derive an adapted signature scheme Φ_{Apt}, which allows one of the participants to hide the discrete logarithm x of a statement X chosen at the beginning of the protocol. Again we extend our previously defined signature scheme with new functions:

$$\Phi_{Apt} \ := \ (\Phi_{MP} \ || \ \mathsf{adaptSig} \ || \ \mathsf{verifyAptSig} \ || \ \mathsf{extWit})$$

- adaptSig takes as input a partial signature $\tilde{\sigma}$ and a secret witness value x. The procedure will output an adapted partial signature $\hat{\sigma}$ which can be verified to contain x using the verifyAptSig function, without immidiately revealing x.

- verifyAptSig takes as input an adapted partial signature $\hat{\sigma}$, the participants public keys and a statement X. The function will verify the partial signature's validity as well that it contains the secret witness x.

- extWit lets Alice extract the secret witness x from the final composite signature. Note that to extract the witness x the partial signatures shared between the participants beforehand and the statement X needs to be provided as inputs. This makes sure that only participants of the procotol will be able to perform the extraction.

Definition 4.3 (Secure Adaptor Signature Scheme)**.** As defined by Aumayr et al. in [AEE+20], a secure adaptor signature scheme needs four security properties to be fulfilled:

1. Adaptor Signature Correctness
2. aEUF − CMA
3. Witness Extractability

We proceed by redefining these properties for our adapted two-party fixed witness signature scheme defined in definition 4.2:

Definition 4.4 (Adaptor Signature Correctness)**.** Similar to how it is defined in [AEE+20] additionally to *correctness* we require our signature scheme to satisfy Adaptor Signature Correctness . This property is given when every adapted partial signature generated by adaptSig can be completed into a final signature for all pairs $(x,\ X) \in R$, from which it will be possible to extract the witness computing extWit with the required parameters.

More formally Adaptor Signature Correctness is given if for every security parameter $n \in \mathbb{N}$, message $m \in \{0,1\}^*$, keypairs $(sk_A, pk_A),\ (sk_B, pk_B)$ with their composite public key $pk_C\ :=\ pk_A\ \cdot\ pk_B$ and every statement/witness pair $(X,\ x)$ in a relation R it must hold that:

$$\Pr\left[\begin{array}{c} 1.\ \mathsf{verf}(m,\ \sigma_{fin},\ pk_C)\ =\ 1 \\ \wedge \\ 2.\ \mathsf{verifyAptSig}(\hat{\sigma_B},\ m,\ pk_A,\ pk_B,\ X)\ =\ 1 \\ \wedge \\ 3.\ (X,\ x' \in R) \end{array} \middle| \begin{array}{l} (x,\ X)\ \leftarrow\ \mathsf{genRel}(1^n) \\ (\tilde{\sigma_A},\ \tilde{\sigma_B})\ \leftarrow\ \mathsf{signPt} < (m,\ sk_A)(m,\ sk_B) > \\ \hat{\sigma_B}\ \leftarrow\ \mathsf{adaptSig}(\tilde{\sigma_B},\ x) \\ \sigma_{fin}\ \leftarrow\ \mathsf{finSig}(\tilde{\sigma_A}, \tilde{\sigma_B}) \\ x'\ \leftarrow\ \mathsf{extWit}(\sigma_{fin},\ \tilde{\sigma_A},\ \hat{\sigma_B}) \end{array}\right] = 1.$$

Definition 4.5 ($\mathsf{aEUF-CMA}$)**.** Additionally to the regular definition of *existential unforgeability under chosen message attacks* as defined for example in [Vau06] we require that it is hard to produce a forged partial signature $\tilde{\sigma}$ if the adversary $\mathcal{A}$ gets to know a valid adapted signature $\hat{\sigma}$ w.r.t. some message m and a statement X.

For the definition of $\mathsf{aEUF-CMA}$ -security we define the experiment $\mathsf{forgeAptSig}_{\mathcal{A}}$ for a PPT adversary $\mathcal{A}$ with a fixed keypair $(sk_{\mathcal{A}},\ pk_{\mathcal{A}})$ and an adapted signature scheme Φ_{Apt} as follows:

$\mathsf{forgeAptSig}_{\mathcal{A}}(n)$

$\mathbb{S} := \emptyset$
$m \leftarrow \mathcal{A}^{\mathcal{O}_{sa}(\cdot)}(pk_{\mathcal{A}})$
$(x,\ X) \leftarrow \mathsf{genRel}(1^n)$
$\tilde{\sigma_B} \leftarrow \mathsf{signPt}(m,\ sk_B)$
$\hat{\sigma_B} \leftarrow \mathsf{adaptSig}(\tilde{\sigma_B},\ x)$
$\tilde{\sigma}_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{sa}(\cdot)}(\hat{\sigma})$
$\sigma_{fin} \leftarrow \mathsf{finSig}(\tilde{\sigma}_{\mathcal{A}}, \tilde{\sigma_B})$
return $(m \notin \mathbb{S} \ \wedge\ \mathsf{verf}(m,\ \sigma_{fin},\ pk_{\mathcal{A}} \cdot pk))$

$\mathcal{O}_{sa}(m)$

$\mathbb{S} := \mathbb{S} \cup m$
$(x,\ X) \leftarrow \mathsf{genRel}(1^n)$
$\tilde{\sigma} \leftarrow \mathsf{signPt}(m,\ sk)$
$\hat{\sigma} \leftarrow \mathsf{adaptSig}(\tilde{\sigma},\ x)$
return $(\hat{\sigma},\ X)$

The adapted signature scheme Φ_{Apt} is called $\mathsf{aEUF-CMA}$ -secure if

$$\Pr[\mathsf{forgeAptSig}_{\mathcal{A}}(n) = 1] \leq \mathsf{negl}(n)$$

Definition 4.6 (Witness Extractability)**.** Informally the Witness Extractability property holds for an adapted signature scheme Φ_{Apt} computed for the statement X when we can always extract the witness $(x,\ X)$ from the final signature σ_{fin}, given the partial signatures of the participants. To formalize this statement we describe an experiment $\mathsf{aExtrWit}_{\mathcal{A}}$ for a PPT adversary $\mathcal{A}$ with a fixed keypair $(sk_{\mathcal{A}},\ pk_{\mathcal{A}})$ and the fixed keypair (sk_B, pk_B) of a second party.

$\mathsf{aExtrWit}_{\mathcal{A}}(n)$

1 : $\mathbb{S} := \emptyset$
2 : $(m,\ X) \leftarrow \mathcal{A}^{\mathcal{O}_{sp}(\cdot)}(pk_{\mathcal{A}})$
3 : $(\tilde{\sigma_{\mathcal{A}}},\ \tilde{\sigma_B}) \leftarrow \mathsf{signPt} < (m,\ sk_{\mathcal{A}})(m,\ sk_B) >$
4 : $\hat{\sigma_{\mathcal{A}}} \leftarrow \mathcal{A}^{\mathcal{O}_{sp}(\cdot)}(pk_{\mathcal{A}},\ \tilde{\sigma_{\mathcal{A}}})$
5 : $\sigma_{fin} \leftarrow \mathsf{finSig}(\tilde{\sigma_{\mathcal{A}}}, \tilde{\sigma_B})$
6 : $x' \leftarrow \mathsf{extWit}(\sigma_{fin},\ \tilde{\sigma_B},\ \hat{\sigma_{\mathcal{A}}})$
7 : **return** $(m \notin \mathbb{S} \ \wedge\ (X,\ x') \notin R \ \wedge\ \mathsf{verf}(m,\ \sigma_{fin},\ pk_{\mathcal{A}} \cdot pk_B))$

$\mathcal{O}_{sp}(m)$

1 : $\mathbb{S} := \mathbb{S} \cup m$
2 : $\tilde{\sigma} \leftarrow \mathsf{signPt}(m,\ sk)$
3 : **return** $\tilde{\sigma}$

In order to satisfy witness extractability the following must hold:

$$\Pr[\mathsf{aExtrWit}_{\mathcal{A}}(n) = 1] \leq \mathsf{negl}(n)$$

4.2 Schnorr-based instantiation

We start by providing a general instantiation of a signature scheme (see definition 3.2): We assume we have a group $\mathbb{G}$ with prime p, H is a secure hash function as defined in definition 3.3 and $m \in \{0,1\}^*$ is a message.

- setup creates a keypair (sk, pk), the public key can be distributed to the verifier(s) and the secret key has to be kept private.

- sign creates a signature consisting of a variable s and R which is a commitment to the secret nonce k used during the signing process.
- verf allows a verifier knowing the signature σ, message m and the provers public key pk to verify the signatures validity.

A concrete implementation can be seen in figure 4.1. The signature scheme is called schnorr signature scheme, first defined in [Sch89] and is widely employed in many cryptography systems. Correctness of the scheme is easy to derive. As s is calculated as $k + e \cdot sk$, when generator g is raised to s, we get $g^{k + e \cdot sk}$ which we can transform into $g^k \cdot g^{sk \cdot e}$, and finally into $R \cdot pk^e$ which is the same as the right side of the equation.

$\mathsf{setup}(1^n)$	$\mathsf{sign}(m,\ sk)$	$\mathsf{verf}(m,\ \sigma,\ pk)$
1 : $x \leftarrow\!\$\ \mathbb{Z}_p^*$	1 : $k \leftarrow\!\$\ \mathbb{Z}_p^*$	1 : $(s,\ R) \leftarrow \sigma$
2 : **return** $(sk\ :=\ x,\ pk\ :=\ g^x)$	2 : $R\ :=\ g^k$	2 : $e\ :=\ \mathsf{H}(m \mid\mid R \mid\mid pk)$
	3 : $e\ :=\ \mathsf{H}(m \mid\mid R \mid\mid pk)$	3 : **return** $g^s\ =\ R^e\ \cdot\ pk$
	4 : $s\ :=\ k\ +\ e\ \cdot\ sk$	
	5 : **return** $\sigma\ :=\ (s, R)$	

Figure 4.1: Schnorr Signature Scheme as first defined in [Sch89]

From the regular schnorr signature we now provide an instantiation in figure 4.2 for the two-party case defined in definition 4.1. Note that this two-party variant of the scheme is what is currently implemented in the Grin mimblewimble cryptocurrency and will provide a basis from which we will build our adapted scheme.

We further show in figure 4.3 how Alice and Bob can cooperate to produce a final signature which fulfills Correctness as defined in definition 3.2.

The final signature is a valid signature to the message m with the composite public key $pk_C\ :=\ pk_A\ \cdot\ pk_B$. A verifier knowing the signed message m, the final signature σ_{fin} and the composite public key pk_C can now verify the signature using the regular verf procedure. The challenge e will be the same because

$$\mathsf{H}(m \mid\mid R \mid\mid pk_C)\ =\ \mathsf{H}(m \mid\mid R_A\ \cdot\ R_B \mid\mid pk_A\ \cdot\ pk_B)$$

In figure 4.4 we further provide a schnorr-based instantiation for the fixed witness adapted signature scheme as defined in definition 4.2:

Again in figure 4.5 we show an example interaction between Alice and Bob creating a signature σ_{fin} for the composite public key $pk_B\ :=\ pk_A\ \cdot\ pk_B$ while Bob will hide his secret x which Alice can extract after the signing process has completed.

4.3 Security

We now show that the outlined instantiation is secure with regards to the regular signature scheme definition 3.2 and the adaptor signature scheme definition 4.2. We start by proving Correctness of the scheme by showing that for two partial signatures $\tilde{\sigma_A}$ and $\tilde{\sigma_B}$:

$$\mathsf{verf}(m,\ \mathsf{finSig}(\tilde{\sigma_A}, \tilde{\sigma_B}),\ pk_A\ \cdot\ pk_B)\ =\ 1$$

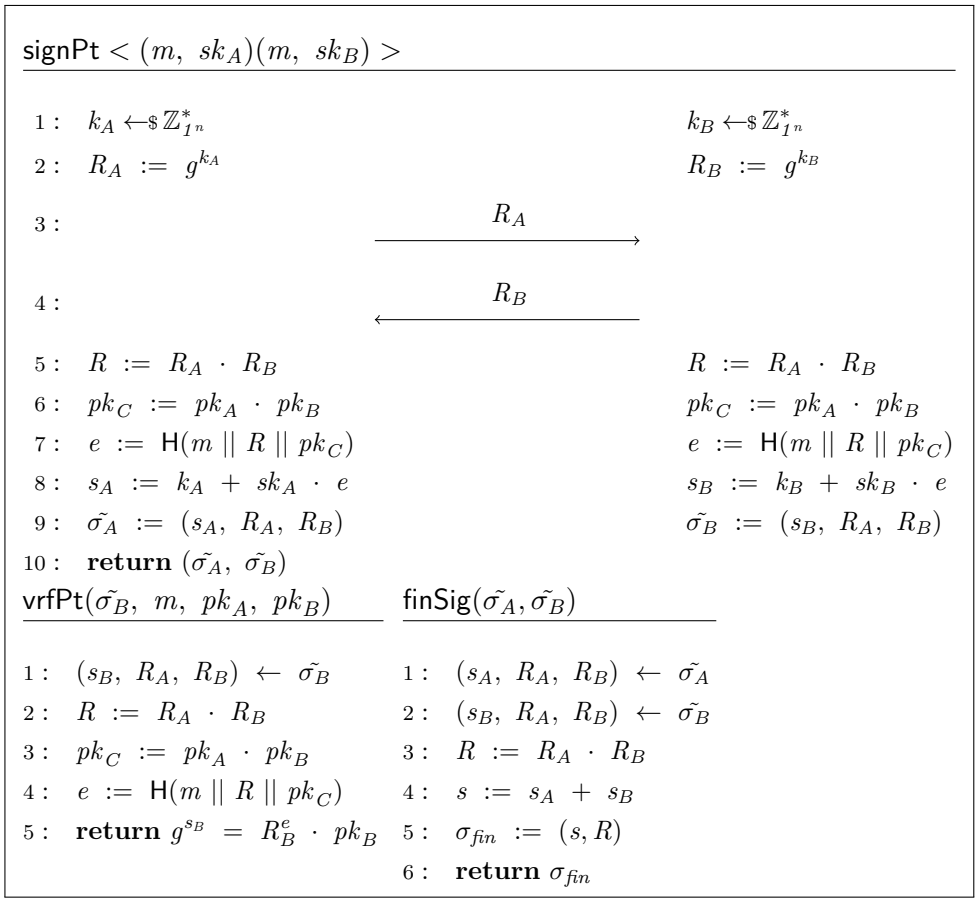

Figure 4.2: Two Party Schnorr Signature Scheme

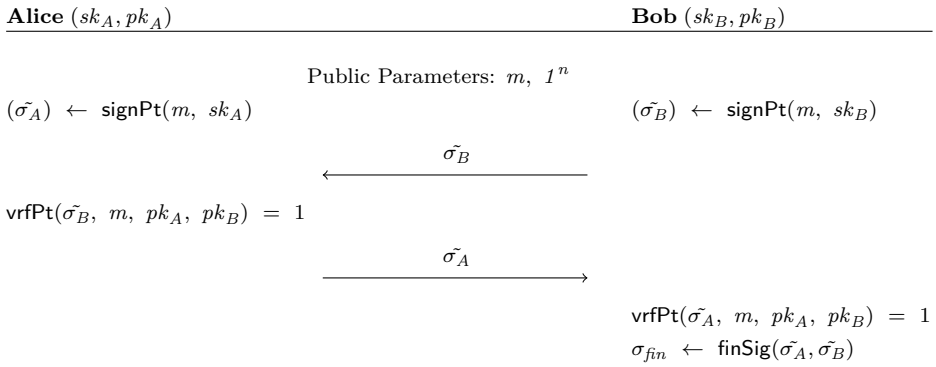

Figure 4.3: Two Party Schnorr Signature Scheme Interaction

$\mathsf{adaptSig}(\tilde{\sigma},\ x)$

1 : $(s,\ R_A,\ R_B) \leftarrow \tilde{\sigma}$
2 : $s' := s + x$
3 : **return** $\hat{\sigma} := (s',\ R_A,\ R_B)$

$\mathsf{verifyAptSig}(\hat{\sigma_B},\ m,\ pk_A,\ pk_B,\ X)$

1 : $(s_B,\ R_A,\ R_B) \leftarrow \hat{\sigma_B}$
2 : $R := R_A \cdot R_B$
3 : $pk_C := pk_A \cdot pk_B$
4 : $e := \mathsf{H}(m \mid\mid R \mid\mid pk_C)$
5 : **return** $g^{s_B} = R_B^e \cdot pk_B \cdot X$

$\mathsf{extWit}(\sigma_{fin},\ \tilde{\sigma_A},\ \hat{\sigma_B})$

1 : $(s,\ R) \leftarrow \sigma_{fin}$
2 : $(s_A,\ R_A,\ R_B) \leftarrow \tilde{\sigma_A}$
3 : $(s'_B,\ R_A,\ R_B) \leftarrow \hat{\sigma_B}$
4 : $s_B := s - s_A$
5 : $x := s'_B - s_B$
6 : **return** (x)

Figure 4.4: Fixed Witness Adaptor Schnorr Signature Scheme

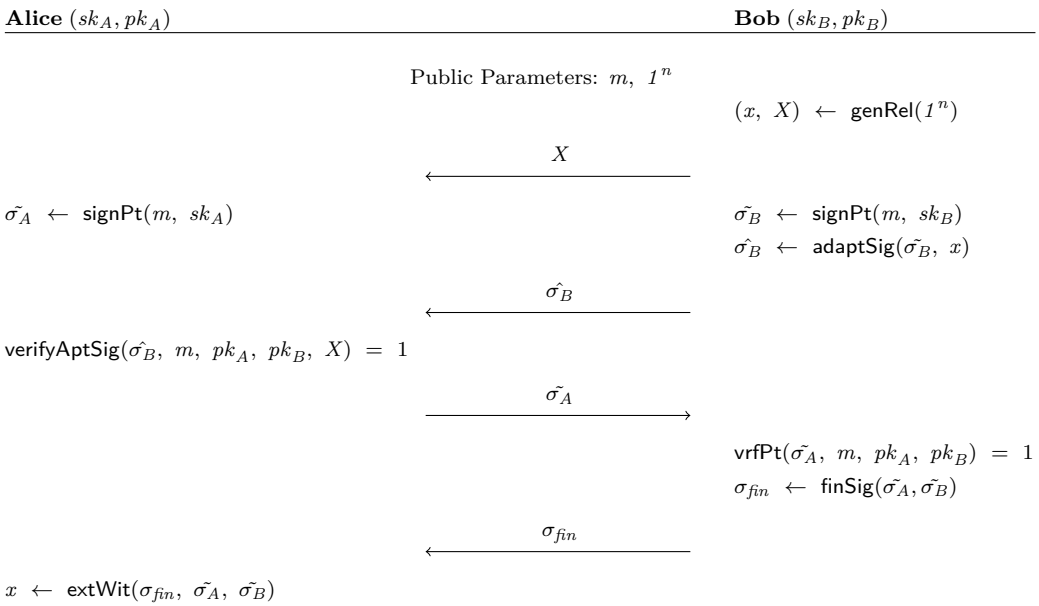

Figure 4.5: Fixed Witness Adaptor Schnorr Signature Interaction

Proof. The correctness proof is by showing equality of the equation checked by the verifier by continous substitutions in the left side of equation:

$$g^{s} = R^{e} \cdot pk_C \tag{4.1}$$
$$g^{s_A} \cdot g^{s_B} \tag{4.2}$$
$$g^{k_A + e \cdot sk_A} \cdot g^{k_B + e \cdot sk_B} \tag{4.3}$$
$$g^{k_A \cdot e} \cdot g^{sk_A} \cdot g^{k_B \cdot e} \cdot g^{sk_B} \tag{4.4}$$
$$R_A^e \cdot pk_A \cdot R_B^e \cdot pk_B \tag{4.5}$$
$$R^e \cdot pk_C = R^e \cdot pk_C \tag{4.6}$$
$$1 = 1 \tag{4.7}$$

□

Next we provide a proof that in addition to regular Correctness also Adaptor Signature Correctness holds. Note that we have 3 statements to prove, we have already proven that $\mathsf{verf}(m,\ \sigma_{fin},\ pk_A \cdot pk_B) = 1$ holds in our instantiation of the signature scheme in the correctness proof 4.3. It remains to prove that with the same setup $\mathsf{verifyAptSig}(\hat{\sigma_B},\ m,\ pk_A,\ pk_B,\ X) = 1$ and $(X,\ x') \in R$ hold.

Proof. For this prove we assume the setup already specified in definition 4.4. First we prove that the following statement:

$$\mathsf{verifyAptSig}(\hat{\sigma_B},\ m,\ pk_A,\ pk_B,\ X) = 1$$

The proof is by continuous substitutions in the equation checked by the verifier:

$$g^{\hat{\sigma_B}} = R_B^e \cdot pk_B \cdot X \tag{4.8}$$
$$g^{\tilde{\sigma_B} + x} \tag{4.9}$$
$$g^{k_B + e \cdot sk_B + x} \tag{4.10}$$
$$g^{k_B \cdot e} \cdot g^{sk_B} + g^{x} \tag{4.11}$$
$$R_B^e \cdot pk_B \cdot X = R_B^e \cdot pk_B \cdot X \tag{4.12}$$
$$1 = 1 \tag{4.13}$$

We now continue to prove the last equation required:

$$(X,\ x' \in R)$$

To prove correctness we show that x is calculated correctly in extWit:

$$x := s'_B - (s - s_A) \tag{4.14}$$
$$s'_B - ((s_A + s_B) - s_A) \tag{4.15}$$
$$s_B + x - (s_B) \tag{4.16}$$
$$x := x \tag{4.17}$$
$$\tag{4.18}$$

□

TODO Proof for pre-signature adaptability, $\mathsf{aEUF-CMA}$ and witness extractability.

List of Figures

3.1 Original transaction building process 14
3.2 Salvaged tranction protocol by Fuchsbauer et al. [FOS19] 15

4.1 Schnorr Signature Scheme as first defined in [Sch89] 22
4.2 Two Party Schnorr Signature Scheme 23
4.3 Two Party Schnorr Signature Scheme Interaction 23
4.4 Fixed Witness Adaptor Schnorr Signature Scheme 24
4.5 Fixed Witness Adaptor Schnorr Signature Interaction 24

List of Tables

List of Algorithms

Bibliography

[AEE+20] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized bitcoin-compatible channels. Cryptology ePrint Archive, Report 2020/476, 2020. `https://eprint.iacr.org/2020/476`.

[AKDB11] Saif Al-Kuwari, James H Davenport, and Russell J Bradford. Cryptographic hash functions: recent design trends and security notions. *IACR Cryptology ePrint Archive*, 2011:565, 2011.

[BBB+18] 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.

[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.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. Meta-multisignature schemes based on the discrete logarithm problem. In *Information Security—the Next Decade*, pages 128–142. Springer, 1995.

[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.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer, 1989.

[Vau06] Serge Vaudenay. *A classical introduction to cryptography: Applications for communications security*. Springer Science & Business Media, 2006.