Grin-Bitcoin Atomic Swap

Lucas Soriano del Pino, Lloyd Fournier

November 2019

1 Introduction

This document sketches an atomic swap protocol between Grin and Bitcoin with a Secp256k1 elliptic curve point as escrow lock. The protocol is semi-scriptless: On Bitcoin we use one 2-of-2 `OP_CHECKMULTISIG` but no other opcodes. The primary goal for this investigation was to develop a *symmetric* atomic swap protocol with Grin. By symmetric we mean the steps and messages of the protocol are arranged in the same way whether Alice is selling Grin or Bitcoin. Symmetric atomic swap protocols are much easier to specify because the logic of the protocol can be expressed independently from the cryptographic algorithms that generate the "smart contract" logic on each particular ledger. Note that prior to this document there existed only a folklore asymmetric protocol[1] where Grin was always the "alpha" ledger.

1.1 Protocol Overview

We create the point/time lock contract (PTLC) on with the following pattern on both ledgers. Prior to broadcasting the fund transaction, the *funder* (The party selling the asset) has a time-locked refund transaction and the *redeemer* (the party buying the asset) has a *one-time verifiably encrypted signature* (one-time VES) on the redeem transaction. Both redeem transactions on each ledger are encrypted under the same encryption key Y for which Alice knows the decryption key y. Once Alice redeems the asset on β, due to the one-timeness of the signature encryption, Bob can recover y and decrypt the signature on his redeem transaction to claim the asset on α. It is easier to see the common logic between this PTLC and a typical *hash/time lock contract* (HTLC) based atomic swap.

To realise this on Grin, the fund transaction has an output with a jointly controlled public key. The two parties cooperatively create the refund signature and the redeem one-time VES through a typical two-party Schnorr signing protocol. Note that the parties must also cooperatively sign the fund transaction since even the owners of an output must sign the transaction (in our case the joint output owned by both parties so both must participate). Additionally, the parties jointly create a range proof on the joint output and share "kernel offsets".

On Bitcoin things are much more straightforward. The fund transaction is a 2-of-2 `OP_CHECKMULTISIG` where each party owns one key. Thus, the refund signature and redeem one-time VES can just use simple single-signer ECDSA algorithms for each.

At the end of the signing phase, each party can produce signatures on the fund transaction, the refund transaction and a one-time VES for the redeem transaction on the other ledger. Thus, the parties can the execute the protocol by broadcasting the transactions with their signatures at the appropriate time. Note that none of the private or public keys need to be remembered once the signing phase is complete.

We attempted with moderate success to describe the protocol within the COMIT paradigm where the cryptocurrency wallet is a separate entity that consumes "actions" (denoted by A) produced by the protocol. This was particularly difficult to do with Grin as the redeem and refund identities controlled by the wallet need to be involved in singing the protocol transactions. It may be possible to address this issue by using the double kernel trick from [2].

1.2 Security

We do not formally model security for this protocol here. Our approach is to ensure that joint transaction public keys and joint nonces are generated uniformly at random through a coin tossing protocol. This protocol can be proved to simulate an incorruptible trusted party choosing the keys and handing them to the parties as described in [3]. We do not attempt (yet) to make the signing phase simulatable in the same way. We conjecture the signing phase is secure because half signatures (and half-bullet proofs) do not provide an adversary any useful information for doing anything other than following the protocol.

2 Protocol Sketch

We now sketch our protocol idea. It is very much in a state of development with some important details left out that only exist in the minds of the authors for now. Its main purpose has been to draw out the missing pieces in our understanding.

2.1 Setup

We assume that before this protocol is executed the parties have come to an agreement regarding the following list of parameters:

1. $v_{\mathsf{grin}}, v_{\mathsf{bitcoin}}$: Amounts to be swapped
2. $E_{\mathsf{grin}}, E_{\mathsf{bitcoin}}$: Expiries - block height for Grin and timestamp for Bitcoin
3. $P^{\mathsf{redeem}}_{\mathsf{grin}}, P^{\mathsf{redeem}}_{\mathsf{bitcoin}}$: Redeem identities
4. $P^{\mathsf{refund}}_{\mathsf{grin}}, P^{\mathsf{refund}}_{\mathsf{bitcoin}}$: Refund identities
5. $\mathbf{I}_{\mathsf{grin}}, \mathbf{I}_{\mathsf{bitcoin}}$: Sets of funding inputs
6. $C_{\mathsf{grin}}, C_{\mathsf{bitcoin}}$: Change outputs
7. $\mathsf{fee}_{\mathsf{grin}}, \mathsf{fee}_{\mathsf{bitcoin}}$: Transaction fees

We will refer to the whole list of setup parameters as ρ, and use a subscript to refer to a subset of it based on the ledger.

2.2 Key generation

The parties participate in the 3-round key generation protocol depicted in Figure 1. Alice generates a secret scalar y and its corresponding curve point yG, where G is an elliptic curve generator point of secp256k1. The encryption key-pair (y, yG) is analogous to the (secret, secret hash) pair used in HTLC-based atomic swaps. The rest of the keys generated during the protocol will later be used in a multi-signature scheme.

Π_{KGen}

Alice		**Bob**
$y \leftarrow\!\!\$\ \mathbb{Z}_q; Y \leftarrow yG$		
$(\mathbf{k}^A_\alpha, \mathbf{K}^A_\alpha) \leftarrow\!\!\$\ \mathsf{KGen}_\alpha$		
$(\mathbf{k}^A_\beta, \mathbf{K}^A_\beta) \leftarrow\!\!\$\ \mathsf{KGen}_\beta$		
$(\mathsf{com}, \mathsf{open}) \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{COM}}(\mathbf{K}^A_\alpha, \mathbf{K}^A_\beta, Y)$		
	$\xrightarrow{\mathsf{com}}$	
		$(\mathbf{k}^B_\alpha, \mathbf{K}^B_\alpha) \leftarrow\!\!\$\ \mathsf{KGen}_\alpha$
		$(\mathbf{k}^B_\beta, \mathbf{K}^B_\beta) \leftarrow\!\!\$\ \mathsf{KGen}_\beta$
	$\xleftarrow{\mathbf{K}^B_\alpha, \mathbf{K}^B_\beta}$	
	$\xrightarrow{(\mathsf{open}, \mathbf{K}^A_\alpha, \mathbf{K}^A_\beta, Y)}$	
		$\mathsf{V}_{\mathsf{COM}}(\mathsf{open}, (\mathbf{K}^A_\alpha, \mathbf{K}^A_\beta, Y)) \stackrel{?}{=} 1$
return $(\mathbf{k}^A_\alpha, \mathbf{k}^A_\beta, \mathbf{K}^A_\alpha, \mathbf{K}^A_\beta, \mathbf{K}^B_\alpha, \mathbf{K}^B_\beta, Y)$		**return** $(\mathbf{k}^B_\alpha, \mathbf{k}^B_\beta, \mathbf{K}^A_\alpha, \mathbf{K}^A_\beta, \mathbf{K}^B_\alpha, \mathbf{K}^B_\beta, Y)$

Figure 1: Key Generation Protocol. $(\mathsf{P}_{\mathsf{COM}}, \mathsf{V}_{\mathsf{COM}})$ are the commitment and opening algorithms for a secure commitment scheme.

The specific key generation algorithms are defined in Figure 2. Private keys go on the left element of the tuple returned by the algorithm; their corresponding public keys, on the right one.

$\mathsf{KGen}_{\mathsf{grin}}$

$x_{\mathsf{fund}} \leftarrow\!\!\$\ \mathbb{Z}_q$; $X_{\mathsf{fund}} \leftarrow x_{\mathsf{fund}} G$
$\mathbf{r} \leftarrow\!\!\$\ \mathbb{Z}_q^3$; $\mathbf{R} \leftarrow \mathbf{r}G$
$\boldsymbol{\phi} \leftarrow\!\!\$\ \mathbb{Z}_q^3$
$\boldsymbol{\tau} \leftarrow\!\!\$\ \mathbb{Z}_q^2$; $\mathbf{T} \leftarrow \boldsymbol{\tau} H$
return $((x_{\mathsf{fund}}, \mathbf{r}, \boldsymbol{\tau}), (X_{\mathsf{fund}}, \mathbf{R}, \mathbf{T}, \boldsymbol{\phi}))$

$\mathsf{KGen}_{\mathsf{bitcoin}}$

$x_{\mathsf{fund}} \leftarrow \mathbb{Z}_q$; $X_{\mathsf{fund}} \leftarrow x_{\mathsf{fund}} G$
return $(x_{\mathsf{fund}}, X_{\mathsf{fund}})$

Figure 2: Key Generation Algorithms

2.3 Signing protocol to build wallet actions

After the key generation phase, both parties have all they need to start signing transactions, with which they will be able to construct wallet actions to be performed during the execution phase of the atomic swap.

$\Pi_{\mathsf{Sig}}(\rho_\alpha, \rho_\beta, K_\alpha^A, K_\beta^A, K_\alpha^B, K_\beta^B, Y, y)$

Alice(k_α^A, k_β^A)		**Bob**(k_α^B, k_β^B)
$\sigma_\beta^R \leftarrow \mathsf{Sig}_\beta^R(\rho_\beta, k_\beta^A, K_\beta^A, K_\beta^B, Y)$		
	$\xrightarrow{\sigma_\beta^R}$	
		$\sigma_\alpha^R \leftarrow \mathsf{Sig}_\alpha^R(\rho_\alpha, k_\alpha^B, K_\alpha^B, K_\alpha^A, Y)$
		$(\mathrm{A}_\beta^{\mathsf{fund}}, \mathrm{A}_\beta^{\mathsf{refund}}, \hat{\sigma}_\beta^{\mathsf{redeem}}) \leftarrow \mathsf{Sig}_\beta^F(\rho_\beta, k_\beta^B, K_\beta^B, K_\beta^A, Y, \sigma_\beta^R)$
	$\xleftarrow{\hat{\sigma}_\beta^{\mathsf{redeem}}, \sigma_\alpha^R}$	
$(\mathrm{A}_\alpha^{\mathsf{fund}}, \mathrm{A}_\alpha^{\mathsf{refund}}, \hat{\sigma}_\alpha^{\mathsf{redeem}}) \leftarrow \mathsf{Sig}_\alpha^F(\rho_\alpha, k_\alpha^A, K_\alpha^A, K_\alpha^B, Y, \sigma_\alpha^R)$		
	$\xrightarrow{\hat{\sigma}_\alpha^{\mathsf{redeem}}}$	
// Alice knows y so she can decrypt $\hat{\sigma}_\beta^{\mathsf{redeem}}$ immediately		
return $(\mathrm{A}_\alpha^{\mathsf{fund}}, \mathrm{A}_\alpha^{\mathsf{refund}}, \hat{\sigma}_\beta^{\mathsf{redeem}})$		**return** $(\mathrm{A}_\beta^{\mathsf{fund}}, \mathrm{A}_\beta^{\mathsf{refund}}, \hat{\sigma}_\alpha^{\mathsf{redeem}})$

Figure 3: Signing Protocol. F and R denote funder and redeemer respectively such that Sig_β^R means the signing algorithm for the redeemer for the β ledger. Note that both parties must verify each message they are sent.

As can be seen on Figure 3, this protocol is generic over the identities of alpha and beta ledger too. The only limitation (imposed by Grin) is that the *redeemer* goes first. We choose Alice here, but it could just as easily be Bob. The public arguments to the protocol are the setup parameters and all the public keys exchanged during key generation. Each party also uses the private keys also generated during key generation. After 3 rounds, Alice has actions to both fund and refund her asset on alpha ledger, and to redeem Bob's asset on beta ledger. Conversely, Bob has actions to fund and refund his asset on beta ledger, and to redeem Alice's asset on alpha ledger.

$\underline{\mathsf{Sig}^R_{\mathsf{grin}}((v, \mathbf{I}, C, \mathbf{P}, \mathsf{fee}, E), (x^R_{\mathsf{fund}}, \mathbf{r}^R, \boldsymbol{\tau}^R), (X^R_{\mathsf{fund}}, \mathbf{R}^R, \boldsymbol{\phi}^R, \mathbf{T}^R), (X^F_{\mathsf{fund}}, \mathbf{R}^F, \boldsymbol{\phi}^F, \mathbf{T}^F), Y)}$

// Generating half-bulletproof for fund transaction

$(T_1, T_2) \leftarrow \mathbf{T}^F + \mathbf{T}^R$

$\tau^R_x \leftarrow \mathsf{bp}^{\frac{1}{2}}(v, (T_1, T_2), X^R_{\mathsf{fund}}, \boldsymbol{\tau}^R, x^R_{\mathsf{fund}})$

// Generating half-signature for fund transaction

$X_{\mathsf{fund}} \leftarrow (X^F_{\mathsf{fund}} + C - \Sigma\mathbf{I} - \boldsymbol{\phi}^F_{\mathsf{fund}} G) + (X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{fund}} G)$

$R_{\mathsf{fund}} \leftarrow \mathbf{R}^F_{\mathsf{fund}} + \mathbf{R}^R_{\mathsf{fund}}$

$s^R_{\mathsf{fund}} \leftarrow \mathbf{r}^R_{\mathsf{fund}} + \mathsf{H}(R_{\mathsf{fund}} || X_{\mathsf{fund}} || \mathsf{fee} H)(x^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{fund}})$

// Generating half-signature for refund transaction

$X_{\mathsf{refund}} \leftarrow (\mathbf{P}_{\mathsf{refund}} - X^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{refund}} G) + (-X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{refund}} G)$

$R_{\mathsf{refund}} \leftarrow \mathbf{R}^F_{\mathsf{refund}} + \mathbf{R}^R_{\mathsf{refund}}$

$s^R_{\mathsf{refund}} \leftarrow \mathbf{r}^R_{\mathsf{refund}} + \mathsf{H}(R_{\mathsf{refund}} || X_{\mathsf{refund}} || \mathsf{fee} H || E)(-x^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{refund}})$

// Generating half-signature for redeem transaction

$X_{\mathsf{redeem}} \leftarrow (-X^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{redeem}} G) + (\mathbf{P}_{\mathsf{redeem}} - X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{redeem}} G)$

$R_{\mathsf{redeem}} \leftarrow \mathbf{R}^F_{\mathsf{redeem}} + \mathbf{R}^R_{\mathsf{redeem}} + Y$

$p_{\mathsf{redeem}} \leftarrow$ $\mathsf{wallet}_{\mathsf{grin}}.\mathsf{getPrivateKeys}(\mathbf{P}_{\mathsf{redeem}})$

$p \leftarrow \mathsf{dumpprivkey}(\mathbf{P}_{\mathsf{redeem}})$

return p

$\hat{s}^R_{\mathsf{redeem}} \leftarrow \mathbf{r}^R_{\mathsf{redeem}} + \mathsf{H}(R_{\mathsf{redeem}} || X_{\mathsf{redeem}} || \mathsf{fee} H)(p_{\mathsf{redeem}} - x^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{redeem}})$

return $((s^R_{\mathsf{fund}}, \tau^R_x), s^R_{\mathsf{refund}}, \hat{s}^R_{\mathsf{redeem}})$

Figure 4: Signing Algorithm for redeemer of Grin.

$\underline{\mathsf{Sig}^F_{\mathsf{grin}}((v, \mathbf{I}, C, \mathbf{P}, \mathsf{fee}, E), (x^F_{\mathsf{fund}}, \mathbf{r}^F, \boldsymbol{\tau}^F), (X^F_{\mathsf{fund}}, \mathbf{R}^F, \boldsymbol{\phi}^F, \mathbf{T}^F), (X^R_{\mathsf{fund}}, \mathbf{R}^R, \boldsymbol{\phi}^R, \mathbf{T}^R), Y, ((s^R_{\mathsf{fund}}, \tau^R_x), s^R_{\mathsf{refund}}, \hat{s}^R_{\mathsf{redeem}}))}$

$/\!/$ Generating bulletproof for fund transaction

$(T_1, T_2) \leftarrow \mathbf{T}^F + \mathbf{T}^R$

$\pi \leftarrow \mathsf{bp}(v, (T_1, T_2), X^F_{\mathsf{fund}}, \boldsymbol{\tau}^F, x^F_{\mathsf{fund}}, \tau^R_x)$

$/\!/$ Constructing fund action

$X_{\mathsf{fund}} \leftarrow (X^F_{\mathsf{fund}} + C - \Sigma\mathbf{I} - \boldsymbol{\phi}^F_{\mathsf{fund}} G) + (X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{fund}} G)$

$R_{\mathsf{fund}} \leftarrow \mathbf{R}^F_{\mathsf{fund}} + \mathbf{R}^R_{\mathsf{fund}}$

$\tilde{s}^F_{\mathsf{fund}} \leftarrow \mathbf{r}^F_{\mathsf{fund}} + \mathsf{H}(R_{\mathsf{fund}} || X_{\mathsf{fund}} || \mathsf{fee} H)(x^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{fund}})$

$/\!/$ Funder to complete $\tilde{s}^F_{\mathsf{fund}}$ using wallet to provide c and $\mathbf{i}$

$\mathrm{A}_{\mathsf{fund}} := (((R_{\mathsf{fund}}, \tilde{s}^F_{\mathsf{fund}} + s^R_{\mathsf{fund}}), \pi), C, \mathbf{I})$

$/\!/$ Constructing refund action

$X_{\mathsf{refund}} \leftarrow (\mathbf{P}_{\mathsf{refund}} - X^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{refund}} G) + (-X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{refund}} G)$

$R_{\mathsf{refund}} \leftarrow \mathbf{R}^F_{\mathsf{refund}} + \mathbf{R}^R_{\mathsf{refund}}$

$\tilde{s}^F_{\mathsf{refund}} \leftarrow \mathbf{r}^F_{\mathsf{refund}} + \mathsf{H}(R_{\mathsf{refund}} || X_{\mathsf{refund}} || \mathsf{fee} H || E)(-x^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{refund}})$

$/\!/$ Funder to complete $\tilde{s}^F_{\mathsf{refund}}$ using wallet to provide p_{refund}

$\mathrm{A}_{\mathsf{refund}} := ((R_{\mathsf{refund}}, \tilde{s}^F_{\mathsf{refund}} + s^R_{\mathsf{refund}}), \mathbf{P}_{\mathsf{refund}})$

$/\!/$ Generating redeem encrypted signature

$X_{\mathsf{redeem}} \leftarrow (-X^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{redeem}} G) + (\mathbf{P}_{\mathsf{redeem}} - X^R_{\mathsf{fund}} - \boldsymbol{\phi}^R_{\mathsf{redeem}} G)$

$R_{\mathsf{redeem}} \leftarrow \mathbf{R}^F_{\mathsf{redeem}} + \mathbf{R}^R_{\mathsf{redeem}} + Y$

$\hat{s}^F_{\mathsf{redeem}} \leftarrow \mathbf{r}^F_{\mathsf{redeem}} + \mathsf{H}(R_{\mathsf{redeem}} || X_{\mathsf{redeem}} || \mathsf{fee} H)(-x^F_{\mathsf{fund}} - \boldsymbol{\phi}^F_{\mathsf{redeem}})$

$/\!/$ It's important that the funder can produce the full encrypted signature

$/\!/$ here to be able to recover y once the decrypted signature is published

$\hat{\sigma}_{\mathsf{redeem}} \leftarrow ((R_{\mathsf{redeem}}, \hat{s}^F_{\mathsf{redeem}} + \hat{s}^R_{\mathsf{redeem}}))$

return $(\mathrm{A}_{\mathsf{fund}}, \mathrm{A}_{\mathsf{refund}}, \hat{\sigma}_{\mathsf{redeem}})$

Figure 5: Signing Algorithm for funder of Grin

$\underline{\mathsf{Sig}^R_{\mathsf{bitcoin}}((v, \mathbf{I}, C, \mathbf{P}, \mathsf{fee}, E), x^R_{\mathsf{fund}}, X^R_{\mathsf{fund}}, X^F_{\mathsf{fund}}, Y)}$

$/\!/$ Generating one signature to refund 2-of-2 multisig

$tx_{\mathsf{fund}} \leftarrow \mathsf{Tx}(v, \mathbf{I}, (\mathtt{OPCMS}(X^F_{\mathsf{fund}}, X^R_{\mathsf{fund}}), C), \mathsf{fee}, \cdot)$

$tx_{\mathsf{refund}} \leftarrow \mathsf{Tx}(v - \mathsf{fee}, (tx_{\mathsf{fund}}), (\mathbf{P}_{\mathsf{refund}}), \mathsf{fee}, E)$

$\sigma^R_{\mathsf{refund}} \leftarrow \mathsf{ECDSA.Sig}(x^R_{\mathsf{fund}}, tx_{\mathsf{refund}})$

$/\!/$ Generating one signature to redeem 2-of-2 multisig.

$/\!/$ This is only sent over for symmetry with Grin; in

$/\!/$ practice it is not needed

$tx_{\mathsf{redeem}} \leftarrow \mathsf{Tx}(v - \mathsf{fee}, (tx_{\mathsf{fund}}), (\mathbf{P}_{\mathsf{redeem}}), \mathsf{fee}, \cdot)$

$\sigma^R_{\mathsf{redeem}} \leftarrow \mathsf{ECDSA.Sig}(x^R_{\mathsf{fund}}, tx_{\mathsf{redeem}})$

return $(\sigma^R_{\mathsf{refund}}, \sigma^R_{\mathsf{redeem}})$

$\underline{\mathsf{Sig}^F_{\mathsf{bitcoin}}((v, \mathbf{I}, C, \mathbf{P}, \mathsf{fee}, E), x^F_{\mathsf{fund}}, X^F_{\mathsf{fund}}, X^R_{\mathsf{fund}}, Y, (\sigma^R_{\mathsf{refund}}, \sigma^R_{\mathsf{redeem}}))}$

$/\!/$ Constructing fund action

$tx_{\mathsf{fund}} \leftarrow \mathsf{Tx}(v, \mathbf{I}, (\mathtt{OPCMS}(X^F_{\mathsf{fund}}, X^R_{\mathsf{fund}}), C), \mathsf{fee}, \cdot)$

$/\!/$ Funder to sign tx_{fund} using wallet to provide $\mathbf{i}$

$\mathrm{A}_{\mathsf{fund}} := (tx_{\mathsf{fund}}, \mathbf{I})$

$/\!/$ Constructing refund action

$tx_{\mathsf{refund}} \leftarrow \mathsf{Tx}(v - \mathsf{fee}, (tx_{\mathsf{fund}}), (\mathbf{P}_{\mathsf{refund}}), \mathsf{fee}, E)$

$\sigma^F_{\mathsf{refund}} \leftarrow \mathsf{ECDSA.Sig}(x^F_{\mathsf{fund}}, tx_{\mathsf{refund}})$

$\mathrm{A}_{\mathsf{refund}} := (\sigma^F_{\mathsf{refund}}, \sigma^R_{\mathsf{refund}})$

$/\!/$ Generating redeem encrypted signature

$tx_{\mathsf{redeem}} \leftarrow \mathsf{Tx}(v - \mathsf{fee}, (tx_{\mathsf{fund}}), (\mathbf{P}_{\mathsf{redeem}}), \mathsf{fee}, \cdot)$

$/\!/$ Funder to keep $\hat{\sigma}^F_{\mathsf{redeem}}$ to be able to recover

$/\!/$ y once the decrypted signature is published

$\hat{\sigma}^F_{\mathsf{redeem}} \leftarrow \mathsf{ECDSA.EncSig}(x^F_{\mathsf{fund}}, Y, tx_{\mathsf{redeem}})$

$\hat{\sigma}_{\mathsf{redeem}} := (tx_{\mathsf{redeem}}, (\hat{\sigma}^F_{\mathsf{redeem}}, \sigma^R_{\mathsf{redeem}}))$

return $(\mathrm{A}_{\mathsf{fund}}, \mathrm{A}_{\mathsf{refund}}, \hat{\sigma}_{\mathsf{redeem}})$

Figure 6: Signing on Bitcoin

3 Future Work

While discussing with the team we came up with the following points that require further exploration:

1. Wallet interaction: How do we design the protocol so that typical wallets can be used to complete the actions?
2. Two-party Bullet Proof Implementation: How do we implement this? Existing code (`https://github.com/jaspervdm/rust-secp256k1-zkp`) is not structured in a way that lets us easily specify a protocol.

References

[1] Grin Developers. Contracts: Atomic Swap. `https://github.com/mimblewimble/grin/blob/master/doc/contracts.md#atomic-swap`. Accessed 2019-11-28.

[2] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. Cryptology ePrint Archive, Report 2018/1039, 2018. `https://eprint.iacr.org/2018/1039`.

[3] Yehuda Lindell. Fast Secure Two-Party ECDSA Signing. Cryptology ePrint Archive, Report 2017/552, 2017. `https://eprint.iacr.org/2017/552`.