komodo

Sia Atomic Swaps with Adaptor Signatures Feasibility Study

By Bud “Alright” Bokanovsky

Introduction

Who we are:

Komodo Platform is an open source blockchain technology project that focuses on providing advanced solutions for blockchain development and interoperability. It aims to address the scalability, security, and interoperability challenges faced by various blockchain networks. Started in 2016, it further developed and pioneered atomic swap technology enabling 500+ coins and tokens to be exchanged trustlessly.

UTXO BV is a software development company set up by Komodo community members and team to enable interaction with other entities on a legitimate real world basis on behalf of the Komodo Platform Project.

Introduction

Who I am:

- “Alright” or “Alrighttt” across various blockchain development communities
- 5+ years experience software developer
- C, C++, rust, python
- Specialize in identifying vulnerabilities in blockchain consensus models and p2p networking stacks

Scope

- The original stated scope of this grant was to produce “a feasibility study to estimate development of the technology to allow Sia adaptor signatures to be able to be used in atomic swaps, specifically in the Atomicdex protocol.”

- After our resource gathering and research phase for the original scope we asked to expand the scope to simply to identify “the simplest or most cost effective path towards SIA integration into existing DEXs”.

Komodo DeFi Framework

- Multi-coin wallet with various DeFi functionality
- Ordermatching engine for p2p atomic swaps
- Allows for integration of essentially any coin's network protocols and consensus rules over a set of generic interfaces
- To put it simply, any new coin protocol can integrate into the framework and automatically have compatible atomic swaps with all other integrated protocols.
- supports BTC-like coins, EVM-based coins and tokens, tendermint, solana and lightning network

Atomic Swaps

Atomic swaps are a way for two parties to trustlessly exchange two sets of funds.

- eg, Alice can exchange her Litecoin for Bob's Bitcoin without trusting him or any third party

This is generally accomplished via what is known as a Hash Locked Time Contract ("HLTC").

- a very simple smart contract supported by bitcoin-like blockchain networks.
- Pioneered in 2013-2014 by Tier Nolan
- Many different variations
- Allows one party to compel the other to reveal a secret needed to spend a set of funds
- All current integrations into the Komodo DeFI Framework implement HLTCs or emulate HLTC behavior via smart contracts

Sia Blockchain

Sia is a unique blockchain requiring a custom integration into the Komodo DeFi Framework

- Sia is written in Go. It is not forked from Bitcoin or any previous project.
- Sia shares the Unspent Transaction Output(or "UTXO") model first established by Bitcoin.
- Sia uses ed25519, unlike Bitcoin and Ethereum which both use secp256k1.
- Sia does not implement any equivalent to Bitcoin's "script" interpreter. This is Bitcoin's basic scripting language which allows for simple conditional statements. Sia has no scripting language to allow similar functionality.
- Sia supports locking UTXOs with a “timelock” or specifying “this UTXO cannot be spent until a specified timestamp has passed.
- Sia supports creating signatures that are not considered valid until a specified timestamp has passed.

Adaptor Signatures

Adaptor Signatures are a cryptographic concept that can be used in place of an HLTC within atomic swap protocols.

- Alice can share an incomplete signature to Bob. Bob is able to complete this signature by adapting it with a secret that Alice would like to know. By completing the signature and sharing it to Alice, Alice receives knowledge of an agreed upon secret.

- Same basic premise of HLTCs; one party compels the other to reveal a secret

Motivations for the Study

Given the limitations of Sia's current consensus model, namely the inability to make conditional statements while sending funds, we believed adaptor signatures to be the best possible way to integrate Sia into the Komodo DeFi Framework.

We believed the Sia team and community would find it unacceptable to make changes to Sia's consensus model to support HLTC-like behavior.

During the course of our research, we made the realization that the needed changes to the consensus model are not nearly as invasive as we had first anticipated. We found Sia's current "UnlockConditions" and planned "Spend Policies" to be a perfectly viable ways of introducing HLTC-like behavior.

Please keep in mind that the scope of the grant was not expanded until the third week of our four week deadline. Because of this, the work towards the adaptor signature route is much more polished.

Adaptor Signature Atomic Swap Protocol

We have designed a protocol that will allow atomic swaps between Sia and various other blockchains including BTC-like blockchains and EVM-based blockchains.

We demonstrate this protocol using BTC-like coins. This logic can also be easily implemented into a EVM smart contract.

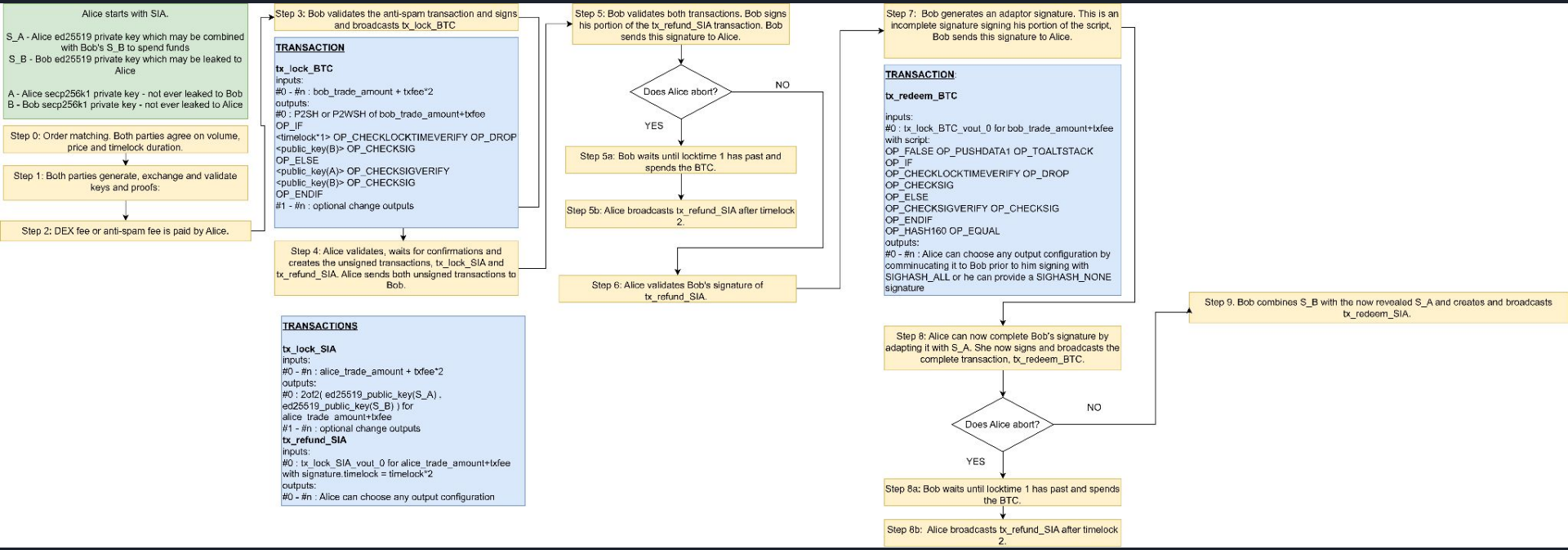
Any blockchain network using ed25519 or secp256k1 signatures with the necessary timelock functionality will be able to use this protocol or an easily adapted version of this protocol.

Adaptor Signature Atomic Swap Protocol

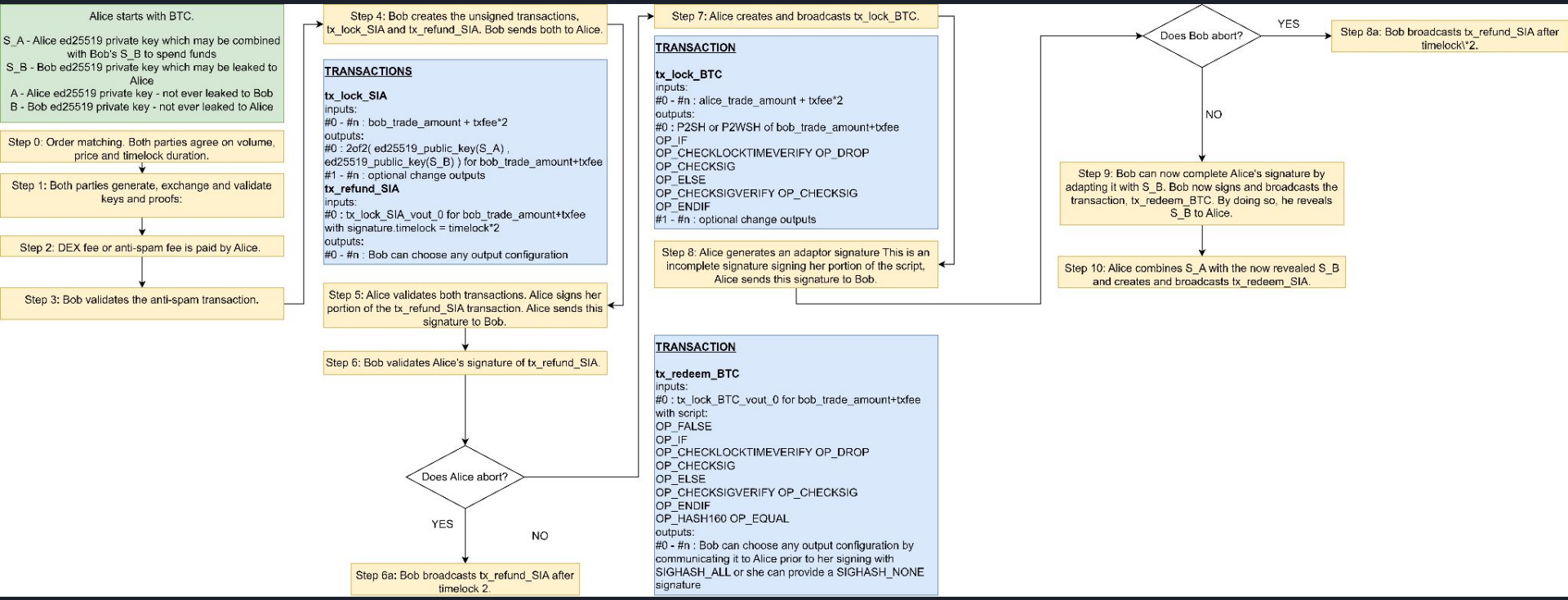
This protocol is based heavily on prior research funded by the Monero community.

- We are reusing concepts developed by “COMIT”[1]
- COMIT developed a zero knowledge proof algorithm [2] allowing a proof to be made that a secp256k1 public key and a ed25519 public key were both generated using the same private key.
- This proof enables us to use secp256k1 adaptor signatures to compel one party to reveal a ed25519 private key to the other party. [3]

Adaptor Signature Atomic Swap Protocol (SIA)



Adaptor Signature Atomic Swap Protocol (BTC)



Introducing HLTC Logic into Sia Consensus

We have identified two viable options for introduce HLTC-like behavior into Sia's consensus model

- Option 1: Introducing New Signature Schemes to Emulate HLTC Behavior
- Option 2: Introducing Spend Policies to Emulate HLTC Behavior.

Introducing HLTC Logic into Sia Consensus
Option 1: Introducing New Signature Schemes to Emulate HLTC Behavior

Sia supports adding additional signature schemes via soft forks.

We believe it possible to emulate HLTC-like behavior via an additional algorithm.

We need to introduce logic that can impose the following validation rules onto a UTXO:

- Alice can spend if she includes the correct secret
- Bob can spend if a timestamp has passed

With these rules in mind, we propose two additional signature schemes to be introduced, one for the "success" path of the HLTC and another for the "refund" path of the HLTC.

Introducing HLTC Logic into Sia Consensus
Option 1: Introducing New Signature Schemes to Emulate HLTC Behavior

For demonstration purposes, a normal Sia address is generally a hashed representation of the following:

```
{
  "unlockConditions": {
    "timelock": 0,
    "publicKeys": [
      "ed25519:2fac394bd26ac986d05b54b8ab052fca209376e1955d2e19ae8297bde1d0e83b"
    ],
    "signaturesRequired": 1
  }
}
```

We propose adding functionality to support the following type of address:

```
{
  "unlockConditions": {
    "timelock": 0,
    "publicKeys": [
      "success:<alice_pubkey><hashed_secret>",
      "refund:<bob_pubkey><timestamp>"
    ],
    "signaturesRequired": 1
  }
}
```

Introducing HLTC Logic into Sia Consensus Option 1:

Success path

The "signature" provided must be the ed25519 signature for `alice_pubkey` with a 32 byte `secret` appended to it.

The validation logic for the the "success" signature scheme must:
1. Validate a provided signature against `alice_pubkey`
2. Validate a provided secret against the `hash_secret` within the unlockCondition. ie, `sha256(secret) == hash_secret`
3. Validate that this provided secret is 32 bytes long.

Refund path

The "signature" provided must be simply the ed25519 signature for `bob_pubkey`

The validation logic for the the "success" signature scheme must:
1. Validate a provided signature against `bob_pubkey`
2. Validate that `timestamp` has passed.

Introducing HLTC Logic into Sia Consensus
Option 2: Introducing Spend Policies to Emulate HLTC Behavior.

Sia is actively developing a new feature called "Spend Policies".

These policies are intended to replace the current "UnlockCondition" functionality within Sia's consensus model.

These policies are an ideal candidate for introducing HLTC-like behavior.

As mentioned in option 1, we need to introduce logic that can impose the following validation rules onto a UTXO:

1. Alice can spend if she includes the correct secret
2. Bob can spend if a timestamp has passed

Introducing HLTC Logic into Sia Consensus
Option 2: Introducing Spend Policies to Emulate HLTC Behavior.

We propose introducing two additional policy types, PolicyTypeHashLock and PolicyTypeTimeLock.

PolicyTypeHashLock must:

1. Validate a provided secret against the hash_secret within the policy. ie, sha256(secret) == hash_secret
2. Validate that this provided secret is 32 bytes long.

PolicyTypeTimeLock must:

1. Validate that the timestamp within the policy has passed

Introducing HLTC Logic into Sia Consensus
Option 2: Introducing Spend Policies to Emulate HLTC Behavior.

We propose using this threshold mechanism to introduce HLTC-like behavior in the following way:

```
{
  "PolicyTypeThreshold": {
    "N": 1,
    "Of": [
      "PolicyTypeThreshold": {
        "N": 2,
        "Of": {
          "PolicyTypePublicKey": "<alice_pubkey>",
          "PolicyTypeHashLock": "<secret_hash>"
        }
      },
      "PolicyTypeThreshold": {
        "N": 2,
        "Of": {
          "PolicyTypePublicKey": "<bob_pubkey>",
          "PolicyTypeTimeLock": "<timestamp>"
        }
      }
    ]
  }
}
```

Resources

[1] COMIT “Monero-Bitcoin Atomic Swap - https://comit.network/blog/2020/10/06/monero-bitcoin/#our-approach-to-the-cross-curve-dleq-proof

[2] COMIT’s “Cross curve proof of Discrete Log equality between secp256k1 and ed25519” - https://github.com/LLFourn/secp256kfun/blob/master/sigma_fun/src/ext/dl_secp256k1_ed25519_eq.rs

[3] Our proof of concept implementation of the needed cryptography for SIA<->BTC atomic swaps via adaptor signatures - https://github.com/Alrighttt/sia_adaptor_sig_study/blob/6dc5fa9a295011009083e3d1156022f4c90203d5/src/main.rs

Thank you!

We hope to be working with you again soon!