Atomic swap

Jump to bottom

alex.starun edited this page on 28 Jun 2019 · 6 revisions

Atomic swap is performed between Beam and other cryptocurrency. Denote it BTC, but actually it can denote other network that supports the needed functionality.

The swap is performed in a single transaction, which effectively transfers the ownership of the Beam UTXO in exchange for some secret, which is needed to claim the being-exchanged coin on the BTC network.

High-level design

There are two parties:

- **A**lice. Owns the Beam, interested to trade it for the BTC
- **B**ob. Owns the BTC, interested to trade it for the Beam.

There are 3 major phases of the atomic swap

1. Prerequesites
    - The being-exchanged UTXOs are locked on both networks for a specific time period (in terms of blocks).
    - Parties monitor both networks to ensure the source UTXOs are indeed locked.
2. Exchange
    - Parties collaborate to create a transaction that transfers the locked Beam to **B** in exchange for the secret.
    - **B** substitutes the secret to the transaction and finally broadcasts it to the network.
    - Once transaction is visible - **A** learns the secret.
    - **A** creates a BTC transaction to claim the BTC UTXO, and broadcasts it to the BTC network.
3. Rollback. In case the swap didn't take place (for whatever reason)
    - Locked UTXOs on both network should remain intact until their lock timeout expires.
    - After the timeout expiration parties broadcast transactions that transfers the locked UTXOs back to them.

Note that **A** actually claims the BTC UTXO after sending the Beam UTXO. This means that the lock timeout of the BTC UTXO must be significantly bigger than that of the Beam UTXO, because **B** may broadcast the Beam transaction just before the timeout expires, the **A** should still have enough time to build and broadcast the BTC transaction.

Technical design

The *unlock secret* is the SHA-256 hash preimage (i.e. a 256-bit value which, after hashing, should be equal to a known 256-bit value). Supported on Beam, BTC, and, probably, many other networks.

Note: The drawback of this scheme is a considerable privacy compromise. As we'll see, it's possible for the attacker to detect such an atomic swap, by looking for the matching hash preimages in both networks.

There is a better option, where the *unlock secret* is an EC scalar, i.e. private key. It's based on the so-called aggregate signatures. It's less generic, since it assumes both networks use the same EC equation and the Generator.

So we'll review the scheme with the Hash-lock, but keep in mind that it can easily be modified to the one with the aggregate signature, and the rest of the argument, and the general flow are exactly the same.

There are no scripts in Beam blockchain. However the following is supported in the transaction kernel:

- Timelock - minimum blockchain height for the kernel to be valid
- HashPreimage

UTXO lock on BTC network

B broadcasts a BTC transaction, which creates an UTXO which can be spent under the following conditions:

1. Timeout + secret key chosen by **B**
2. Hash preimage chosen by **B** + secret key chosen by **A**

UTXO lock on Beam network

This part is somewhat more complex, since there are no scripts in Beam. **A** and **B** collaborate to create a *shared* UTXO, whose blinding factor is shared between them.

1. **A** and **B** randomly choose their parts of the blinding factor for the shared UTXO
2. **A** and **B** first create the "rollback" transaction, which would transfer the shared UTXO back to **A** if the exchange didn't take place.
    - This transaction kernel is Time-locked, i.e. transaction can only be broadcasted starting from specific blockchain height.
    - **A** saves this transaction, but doesn't broadcast yet.

3. **A** and **B** create the transaction that spends **A**'s UTXO and creates the shared one.
    - The tricky part is creating the Bulletproof for the shared UTXO. Takes 3 iterations.
4. They broadcast this transaction to the Beam network, to create the shared UTXO.

Exchange transaction

At this point both Beam and BTC UTXOs are locked, both parties get confirmations for this. In addition **A** knows the *Image* of the locked BTC UTXO. Now they collaborate to build a Beam transaction that transfers the shared (locked) Beam UTXO to **B** in exchange for revealing the *Hash Preimage*, which, after hashing, equals to the known *Image*.

- **B** randomly chooses the blinding factor for the new UTXO
- **A** creates the Transaction Kernel, which is supposed to contain the *Hash Preimage* (but doesn't contain yet), which corresponds to the knwo *Image*.
- **A** puts her part of the kernel multisig, which assumes the correct *Image*.
- Incomplete transaction is passed to **B**.
- **B** puts his part of the kernel multisig, which assumes the correct *Image* (same as Alice).
- **B** substitutes the correct *Hash Preimage* to make the transaction kernel valid.
    - This is where the atomic swap actually occurs!
- **B** broadcasts the transaction to the Beam network.

A monitors the Beam network. Once the exchange transaction kernel is visible:

- **A** realized that the exchange occurred.
- **A** learns the *Hash Preimage*.
- **A** creates and broadcasts the BTC transaction to claim her BTC UTXO.

In-depth flow diagram

Bob

- Collaborate to lock the BTC UTXO
    - Get `Pka` from **A** - her generated pubkey for some secret key `ska`.
    - Generate the `hpi` - the *Hash Preimage*.
    - `hi = Hash(hpi)` - the *Hash Image*.
    - Generate and broadcast the transaction that creates a locked BTC UTXO.
        - The created UTXO can be spent iff both `hpi` and `ska` are known.
        - `hi` and `Pka` are revealed.
    - Send **A** this transaction (so she'll be able to identify it in the BTC network).
- Collaborate to lock the Beam UTXO

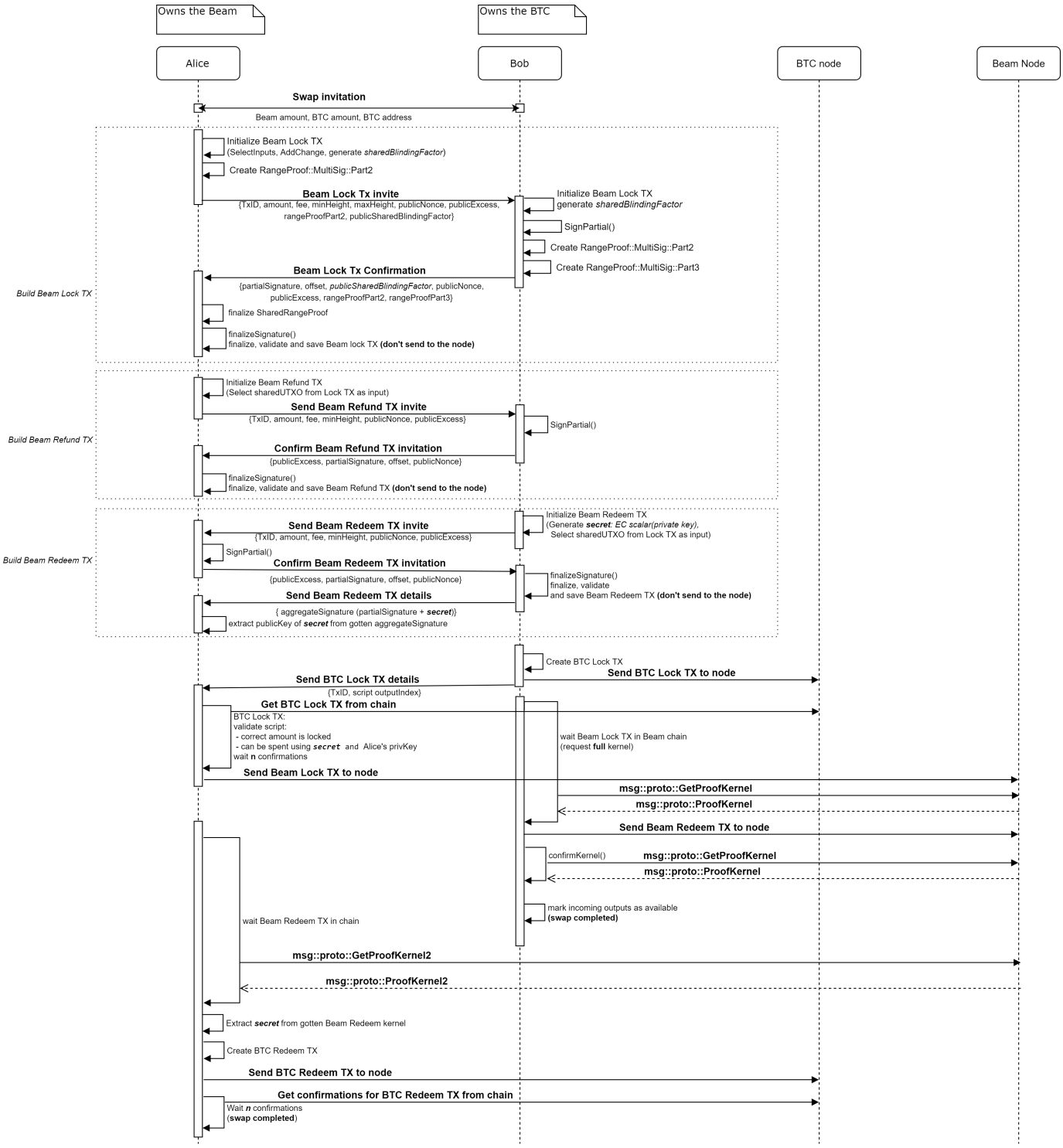
    - Generate `sfb` - the part of the blinding factor of the shared UTXO.
    - Get `Pfa` - **A**'s part of the public blinding factor (of the shared UTXO).
    - Collaborate to sign the kernel for the transaction that transfers the shared (non-existing yet) UTXO back to **A**
        - The kernel must be time-locked
        - **B**'s part of the signature accounts for the `sfb`.
    - Collaborate to create the Bulletproof of the shared UTXO.
        - requires 3 iteration cycles with **A**
    - Collaborate to sign the kernel for the transaction that creates the shared UTXO
        - **B**'s part of the signature accounts for the `sfb`.
- Wait until the mutual UTXO becomes visible
- Collaborate to build the exchange transaction
    - Transaction kernel is supposed to (but doesn't yet) contain the `hpi`.
        - Means - its signature signs the kernel contents, including `Hash(hpi) == hi`, whereas `hi` is known to both parties.
    - Generate the new UTXO
    - Pass it to **A** so that she creates such a kernel, and substitutes her part of the signature.
    - Get the half-signed kernel from **A**, and verify it.
    - **Important:** Ensure there is enough time left until the timelock of the shared UTXO expires!
    - Finish the kernel signature.
        - Substitute the `hpi` as the *Hash Preimage*.
        - Complete the signature.
    - Broadcast the transaction to the Beam network.
- Wait until the exchange transaction becomes visible, or until the BTC UTXO timelock expires.
    - If the transaction is visible (and enough new blocks are generated above) - Congratulations! It's done.
    - If the BTC UTXO timelock expired and it's still unspent - take it back (create and broadcast another BTC transaction).

Alice

- Collaborate to lock the BTC UTXO
    - Generate `ska` - private key on the BTC network.
    - Send `Pka = G * ska` to **B**.
    - Receive the BTC transaction details from **B**, and verify it
        - Correct amount is locked.
        - Can be spent using `ska`, and a *Preimage* of `hi` (the preimage is not known yet).
- Collaborate to lock the Beam UTXO

    - Select input(s) to build the transaction that creates the shared UTXO
    - Generate `sfa` - the part of the blinding factor of the shared UTXO.
    - Get `Pfb` - **B**'s part of the public blinding factor (of the shared UTXO).
    - Collaborate with **B** to build:
        - A time-locked rollback transaction that transfers the shared (non-existing yet) UTXO back to **A**.
            - It must be created before the shared UTXO is actually created.
        - Bulletproof for the shared UTXO
        - Transaction that spends the selected inputs and creates the shared UTXO
    - Send the transaction to create the shared UTXO to the network
- Wait until the locked BTC UTXO becomes visible in the BTC network
- Ensure the BTC timelock is significantly bigger than the Beam timelock
- Collaborate to build the exchange transaction
    - Transaction kernel is supposed to (but doesn't yet) contain the `hpi`.
    - **A** creates such a kernel, puts her part of the signature, which
        - Corresponds to her part of the shared UTXO (compensates for `ska`).
        - Would become valid only when the correct *Hash Preimage* (`hpi`) will be substituted.
    - Half-signed transaction is passed to **B**
- Wait until the exchange transaction becomes visible, or until the Beam UTXO timelock expires.
    - Once the transaction is visible:
        - **A** learns the `hpi` from the visible transaction kernel
        - **A** Creates and broadcasts the BTC transaction to claim the BTC UTXO
    - If the Beam UTXO timelock expired and it's still unspent
        - Broadcast the rollback transaction to the network
    - Wait until the transaction (either post-exchange or the rollback) becomes visible.

Swap diagram



Clone this wiki locally

```
https://github.com/BeamMW/beam.wiki.git
```