Non-linkable CoinSwap backouts using weak blinded ECDSA signatures

version 0.1

1 Introduction

In this scheme, there are Two participants, **Signer** and **Blinder**. Blinder will own the *secret*, a signature for a backout transaction "somewhat" blindly signed by Signer. The setup is supposed to ensure that if Blinder ever uses Signer's signature for the backout, then Signer will learn a secret value, allowing him to backout from Blinder's own funding transaction. Also, the setup must ensure that Signer can not learn the unblinded hash being signed before the backout is used. The coinswap scheme tries to achieve a succesful atomic swap such as the one described in [1], with the added benefit for unlinkability for the atomic backouts, in case the swap fails. The blinding scheme used here is described in [2], with modifications in favor of the Signer.

2 Setup

The setup begins the same way as outlined in [1], whereby the two participants negotiate pubkeys to be used for the initial funding **TX0** and **TX1**, and final successful transactions **TX4** and **TX5**. Additionally, pubkeys and *locktimes* for backout transactions are exchanged. Specifically, Blinder will send one pubkey to Signer, noted as BLN1, and Signer will send two pubkeys SGN1 and SGN2.
Once the funding transactions have been determined and pubkeys exchanged, the next stage can begin.

3 Backout setup

Signer first chooses at random two secrets p and q, and computes the points $P = p^{-1} * G$ and $Q = \left(q * p^{-1}\right) * G$, which he then sends to **Blinder**
They then proceed with constructing the blinded signature :

Blinder
Receive P, Q from signer
Choose a, b, c, d at random

Compute R, the public nonce to be used in validation
$R = (a * c)^{-1} * P$; $r = R_x$

Compute T, the pubkey to be used in validation
$T = (a * r)^{-1} * \left(b * G + Q + \left(d * c^{-1}\right) * P\right)$

Prepare script $scr1$

```
IF
   2 <SGN1> <SGN2 + T> 2 CHECKMULTISIG
ELSE
   [L0] CLTV DROP <BLN1> CHECKSIG
ENDIF
```

pay to one of signer's pubkeys, and a second signer's pubkey tweaked by the secret, or blinder's pubkey after `L0` *has passed*

Prepare script $scr2$

```
IF
   2 <BLN2> <T> 2 CHECKMULTISIG
ELSE
   [L1] CLTV DROP <SGN3> CHECKSIG
ENDIF
```

pay to one of blinder's pubkeys and the secret, or to signer's pubkey after `L1` *has passed*

Compute the *sighash h1*.
The input is **TX1** and the output is set to *scr2*. This makes **TX3**.

Compute the blinded sighash *h2*
$h2 = a * h1 + b$

Compute points B, D
$B = b * G$; $D = d * G$

Send to **Signer**
$\{a, c, h2, B, D\}$
Signatures for spending **TX0**, paying to **TX2**

Signer

Compute k
$k = (c * a * p)^{-1}$
secret nonce, corresponding to R

Compute T from own values
$T = \left(k * r^{-1}\right) * ((c * p) * B + (q * c) * G + D)$
same T that blinder is supposed to be using

Prepare *scr1* and *scr2* from own values , checking correct use of T in them by blinder

Compute a blinded signature `s` value *s1*
$s1 = p * h2 + q$

Compute the point $h1 * G$, a point where the discreet log is $h1$
$h1 * G = a^{-1} * (h2 * G - B)$

Check that the blinded signature verifies $h1 * G$ by checking
$k * ((c * s1) * G + D) == h1 * G + r * T$

Send to **Blinder**
Blinded `s` value *s1*
Signatures paying from **TX0** to **TX2**, and from **TX1** to **TX3**

Blinder

Unblind *s1*, producing a valid signature *sig1* over *h1*, signed by pubkey T with the public nonce *R*, and verify
$s2 = c * s1 + d$
$s2 * R == h1 * G + r * T$

Send to **Signer** a signature for payment from **TX1** to **TX3**

Assuming everything verifies up until now, both broadcast **TX0** and **TX1**, and wait for them to confirm.

4 Post funding transactions confirming

Once **TX0** and **TX1** both confirmed, the protocol continues with the same semantics as in [1]. **Blinder** should send **Signer** the *secret* in the form of the unblinded signature *sig1*. Signer is then able to extract the secret *t* from it, enabling his immediate backout using **TX2**.
With all backout paths covered for both, the parties are able to complete the swap successfuly through **TX4** and **TX5**.
However, Blinder can also choose to abort the swap by using *sig1* for an immediate backout from **TX3**. If this happens, Blinder is still able to learn the secret by monitoring the chain.

Upon being relayed *sig1* in one way or another, **Signer** can extract the secret t by solving :
$t = r^{-1} * (s2 * k - h1)$
$T == t * G$

Alternatively, Signer might choose to halt the protocol after receiving *sig1*, or after **TX4** being broadcast by Blinder.
In this case, Blinder should still be able to successfully use *sig1* for redeeming **TX3**.

References

[1] Coinswap - with privacy
`https://joinmarket.me/blog/blog/coinswaps/`

[2] Bitcoin Blind Signatures
`https://github.com/oleganza/bitcoin-papers/blob/master/BitcoinBlindSignatures.md`