Blind Signatures in Scriptless Scripts

Jonas Nick
jonasd.nick@gmail.com
@n1ckler

February 17, 2019

Blockstream

Schnorr Signatures

G is generator of a DLog hard group

```
def keygen():
  x := rand, P := x*G
  return (x, P)

def sign(x, m):
  k := rand, R := k*G
  s := k + hash(R,P,m)*x
return (R, s)

def verify(P, m, R, s):
  return s*G ?= R + hash(R,P,m)*P
```

Blind Signatures

- Blind signer does not know the message being signed
- Blind signature gets *unblinded*, s.t. unblinded and blind sig are not linkable
- 2 party protocol between Client and Server
  1. Client knows message, Server does not
  2. Client creates blind challenge
  3. Server signs blind challenge and gives to Client
  4. Client unblinds signature

Blind Schnorr Signatures

```
def server_nonce():
  k := rand, R := k*G
  return (k, R)


def client_blind(R, m):
  alpha := rand, beta := rand
  R' := R + alpha*G + beta*P
  c' := hash(R',P,m), c := c'+beta
  return (alpha, beta, R', c', c)
```

Blind Schnorr Signatures

```
def server_sign (k, x, c):
  s := k + c*x
  return s


def client_unblind(alpha, s):
  s' := s + alpha
  return s'


def verify(P, m, R, s):
  return s*G ?= R + hash(R,P,m)*P
```

EDIT: Blind Schnorr Signatures are not secure per se

- EDIT: At this point I should have stressed that implementations of above Blind Schnorr signatures are not secure against forgery
- They are vulnerable to Wagner's attack. With 65536 parallel signing sessions can forge a signature with only $O(2^{32})$ work.
- This could be somewhat mitigated by having the signer abort in $p < 1/2$ cases during the signing step.

EDIT: Blind Schnorr Signatures are not secure per se

- Moreover, they can't be proven secure in the ROM (Baldimtsi, Lysyanskaya)
- Use them only when the signatures need to pass normal Schnorr verification.

Ecash

- Trusted server maintains a database to prevent double spending
- Server database consists of serial numbers that have been spent
- Ecash token is tuple
  - `(serial number, server sig(serial number))`

Ecash

Reissuance protocol to exchange token for fresh unlinkable token:

1. Client chooses new random serial number and blind challenge
2. Client shows token and blind challenge to Server
3. Server aborts if token serial number is in DB
4. Server signs blind challenge and inserts serial number in database
5. Client unblinds signature to get fresh token

(reissuance proocol is also used for payments)

Exchange Discrete Logs for Bitcoin

Buyer wants to buy discrete log of point `T = t*G` from Seller

1. Buyer creates multisig output with Seller
2. Seller sends transaction and *adaptor signature* with `T` to Buyer
3. Buyer gives Seller her signature over the transaction
4. Seller spends the output
5. Buyer computes the discrete log `t` from Seller's tx signature and adaptor signature

Blockchain footprint: single boring transaction

Exchange Discrete Logs for LN Payments

Requires Lightning with *Multi-Hop Locks* in place of HTLCs.

- HTLCs
  1. Buyer $\xrightarrow{\texttt{hash(p)}}$ Hop $\xrightarrow{\texttt{hash(p)}}$ Seller
  2. Seller claims with preimage p

Exchange Discrete Logs for LN Payments

Multi-Hop Locks use curve points instead of hashes

1. Buyer $\xrightarrow{\text{L1}}$ Hop $\xrightarrow{\text{L2}}$ Seller
   - Buyer set up route to buy discrete log of `T = t*G`
   - `L1 = T + l1*G`
   - `L2 = L1 + l2*G`
2. Buyer gives `l2` to Hop and `l1 + l2` to Seller
3. Seller claims `L2` with `c2 = (t + l1 + l2)`
4. Hop claims `L1` with `c1 = c2 - l2`
5. Buyer computes `t = c1 - l1`

Exchange Blind Signatures for Bitcoin

Building block: Committed R-point signatures

1. Seller creates public nonce `R`
2. Buyer can compute Seller's signature `s` times `G` (without being able to compute `s` of course) as `R + hash(R,P,m)*P = s*G`

Exchange Blind Signatures for Bitcoin

- Committed R-point signatures work for blind signatures too! So Client computes Server's blind signature `s` times `G`
- Then buys discrete log of `s*G` with bitcoins using above techniques

Blind Coinswaps

- Coinswap where Server can not link coins
  - similar to Tumblebit
  - but with scriptless scripts
- Key idea: Client buys blind signature from Server where the message is a transaction that gives Server's coins to the Client

Blind Coinswaps

1. Create blind challenge from unsigned tx
2. Create blind sig times G
3. Pay for its DLog (the actual blind sig)
4. Unblind and broadcast tx

Exchanging Ecash Tokens for Bitcoin

- Want Clients buy tokens from each other without trust
- Server needs to make sure that buyer gets token and seller bitcoins
  1. Payment uses multisig with Server
  2. Or Server is part of lightning payment route
  3. **Or issue token with locktime**

Brands credentials

- think ecash tokens that encode more attributes
- essentially Pedersen multicommitments of the attributes
  - `a1*G1 + a2*G2 + ....  + r*G`
- allows proving properties of token attributes in zero knowledge

Brands credentials

Reissuance of example ecash token `(type, amount, serial number, server signature)`

1. Client chooses new random serial number and blind challenge
2. Client shows token and blind challenge to Server
3. **Client proves that type and amount in token and in challenge is the same**
4. [...]

Exchanging Ecash Tokens for Bitcoin

- Token seller runs version of reissuance protocol with Server and receives two tokens with the same serial number
- Buyer token `(BUYERSECRET, SELLERSECRET, type_buyer, amount, serial number, server signature)`
  - reissued only when providing `BUYERSECRET`, `SELLERSECRET`
- Seller token `(LOCKTIME, type_seller, amount, serial number, server signature)`
  - reissued only when `LOCKTIME` is over

Exchanging Ecash Tokens for Bitcoin

1. Seller gives buyer token without `SELLERSECRET` to buyer and proves that `LOCKTIME` of seller token is sufficiently far in the future
2. Buyer buys `SELLERSECRET` from seller either on-chain or off-chain using above techniques
3. EITHER Buyer runs reissuance protocol with Server
   OR Buyer becomes unresponsive and Seller runs reissuance protocol after `LOCKTIME`

Conclusion

- Blind signatures are useful in Bitcoin protocol designs (blind coinswaps)
- Can build trustless off-chain or on-chain ecash token exchange protocols using scriptless scripts
- Next steps
  - Schnorr soft fork
  - Lightning v1.x

Further Reading

- Schnorr, C.. Security of Blind Discrete Log Signatures Against Interactive Attacks
- Scriptless Scripts
- Scriptless Scripts in Lightning
- Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., & Maffei, M. Multi-Hop Locks for Secure, Privacy-Preserving and Interoperable Payment-Channel Networks.
- Blind Coinswaps
- Brands, S. (2002). A technical overview of digital credentials.

Q&A

- Slides: https://nickler.ninja/slides/2018-bob.pdf
- Questions?