Schnorr and Taproot in Lightning

2018-09-01 Jonas Nick jonasd.nick@gmail.com https://nickler.ninja @n1ckler

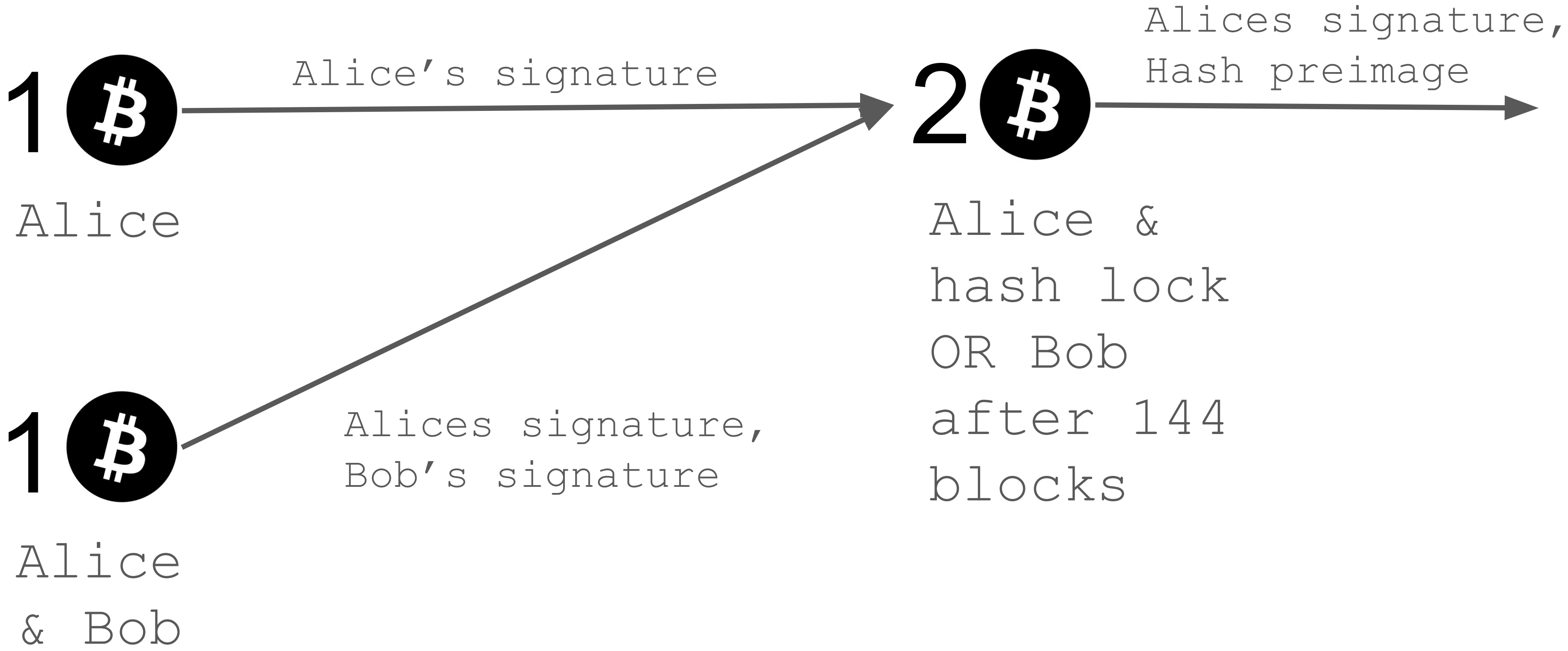
Objective: Increase Robustness

- Privacy
- Scalability
- Consensus

Scriptless Scripts approach: different payment types (multisig, lightning channels, etc) should look like normal payments.

1. Participants communicate directly
2. That results in a simple transaction (“Alice pays Bob”)

Introduction: bitcoins



1 ₿

Alice

Alice’s signature

1 ₿

Alice
& Bob

Alices signature,
Bob’s signature

2 ₿

Alice &
hash lock
OR Bob
after 144
blocks

Alices signature,
Hash preimage

Bitcoin Scripts

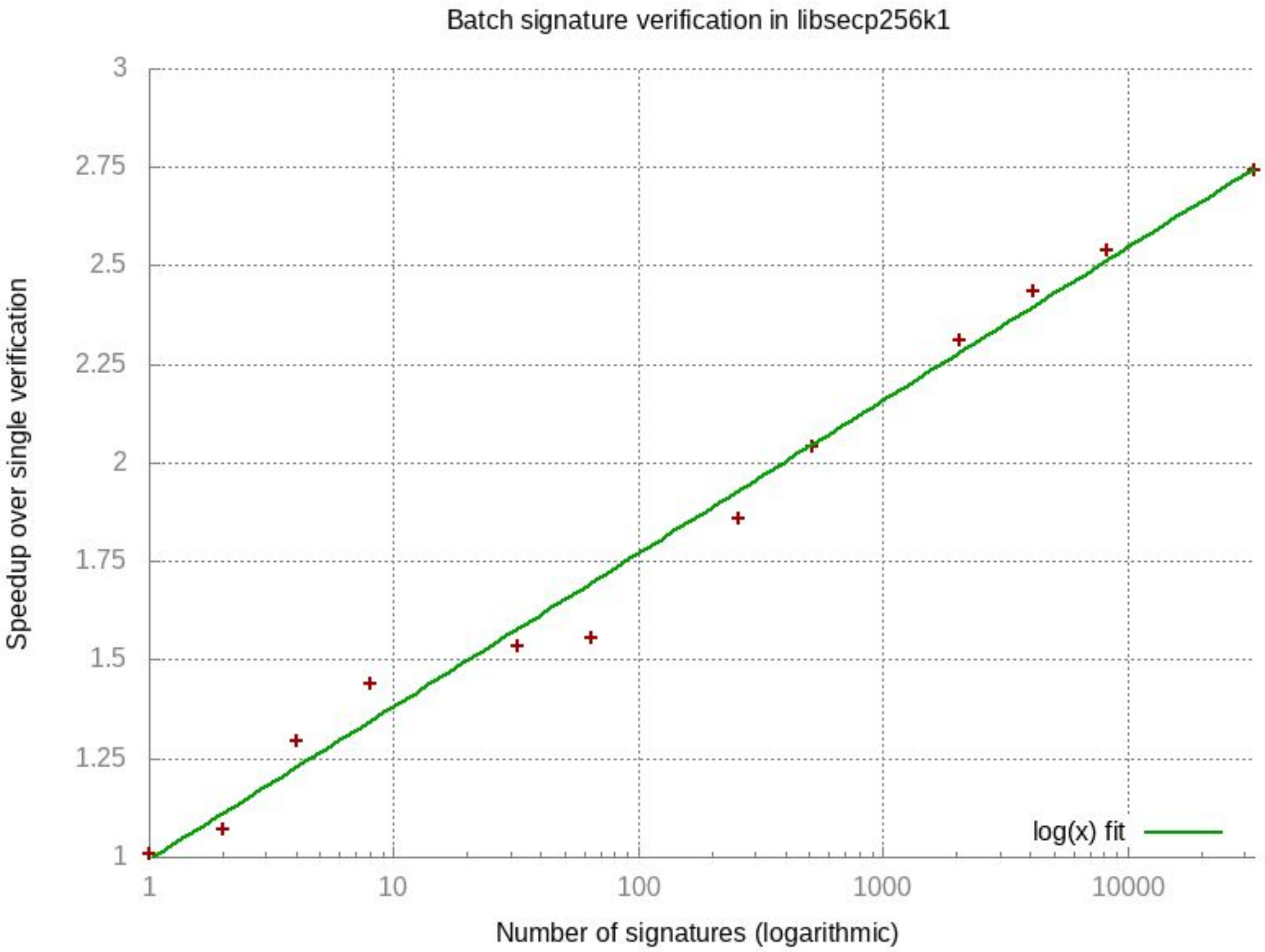
Script	**Witness**

`<pubkey>`
`OP_CHECKSIGVERIFY`
	`<signature>`

`2 <pubkey1> <pubkey2> 2`
`OP_CHECKMULTISIGVERIFY`
	`<signature1>`
	`<signature2>`

Schnorr Signatures

- Currently: Elliptic Curve Digital Signature Algorithm (ECDSA)
- Schnorr signatures is a different signature scheme that could be used instead
- BIP recently was proposed to standardize them for Bitcoin
- No new crypto assumptions, stronger security proof
- Efficiently batch verifiable: multiple signature verifications at once are faster than individually


Batch signature verification in libsecp256k1
Speedup over single verification
3
2.75
2.5
2.25
2
1.75
1.5
1.25
1
1
10
100
1000
10000
Number of signatures (logarithmic)
log(x) fit

Schnorr Signatures

Add new consensus rule to add Schnorr signature validation to Script

Script	**Witness**	**Meaning**

`<pubkey>`
`OP_SCHNORR`
	`<signature>`	● Normal payment?
		● k-of-n multisig?
		● Lightning cooperative close?
		● Hash lock?

Size: 32 bytes public key + 64 bytes signature

Schnorr Signatures: 2-of-2 MuSig

1. Create combined public key `P` from Alice’s key `A` and Bob’s key `B`

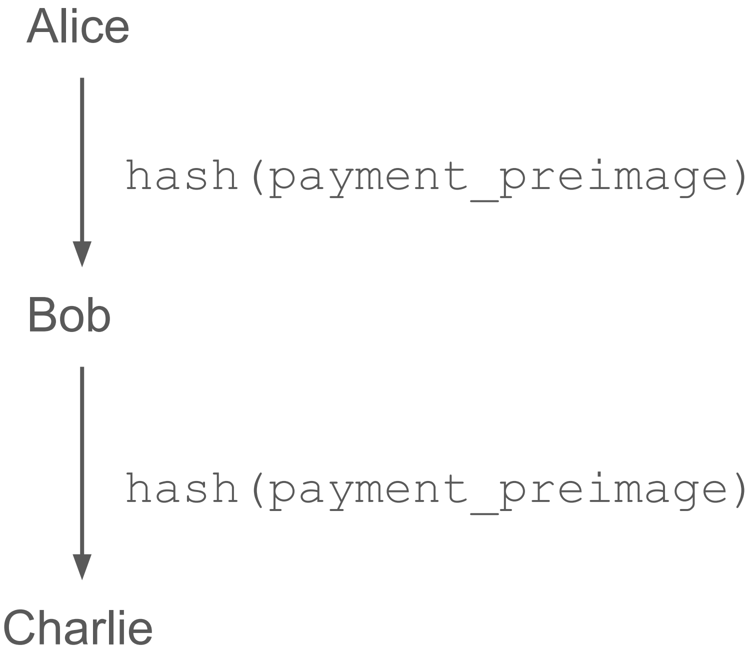
   `P = hash(A,B,0)*A + hash(A,B,1)*B`

2. Interactively sign transaction

```
Alice:           Bob:

nonce commitment ->
                  <- nonce commitment
           nonce ->
                  <- nonce
     partial sig ->
                  <- partial sig
    combine               combine
```

Payment Forwarding with Hash Locks



Alice

hash(payment_preimage)

Bob

hash(payment_preimage)

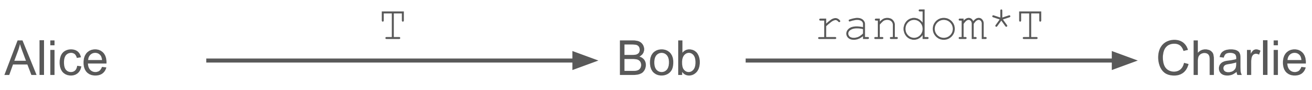
Charlie

Hash Locks

Script	Witness	Meaning
... <payment_hash> ... <pubkey> OP_CHECKSIG	<payment_preimage> <signature>	Forces spender to reveal the payment preimage which can be used to atomically swap payments.

Locks with Schnorr & Adaptor Signatures

Hash locks	**Discrete Log based locks**
`hash(payment_preimage)`	`payment_preimage*G`
“On-chain”: payment_preimage explicit in tx	“Off-chain”: Payment_preimage computable from normal tx signature & adaptor signature
	Routing privacy
	Allows proof of payment and buying discrete logarithms



Locks with Schnorr & Adaptor Signatures

Script	**Witness**	**Meaning**

`<pubkey>`
`OP_SCHNORR`
	`<signature>`	● Normal payment?
		● k-of-n multisig?
		● Lightning cooperative close?
		● Hash lock?

Size: 32 bytes public key + 64 bytes signature

Locks with Schnorr & Adaptor Signatures

- Bob knows some secret, Alice wants to know it
- They have a 2-of-2 MuSig output
- Alice signs a transaction only when it in turn learns the secret

1₿

Alice
& Bob

Main idea: Bob sends Alice *adaptor signature* before Alice sends partial signature.

```
secret = adaptor_sig + Alice_partial_sig - combined_sig
```

Locks with Schnorr & Adaptor Signatures

- Bob knows some secret, Alice wants to know it
- They have a 2-of-2 MuSig output

1₿

Alice
& Bob

```
Alice:                      Bob:
            … exchange nonces …
                     <- adaptor sig
 verify adaptor sig
       partial sig ->
                        partial sign
                        combine
```

Bob spends coin, Alice computes lock secret as

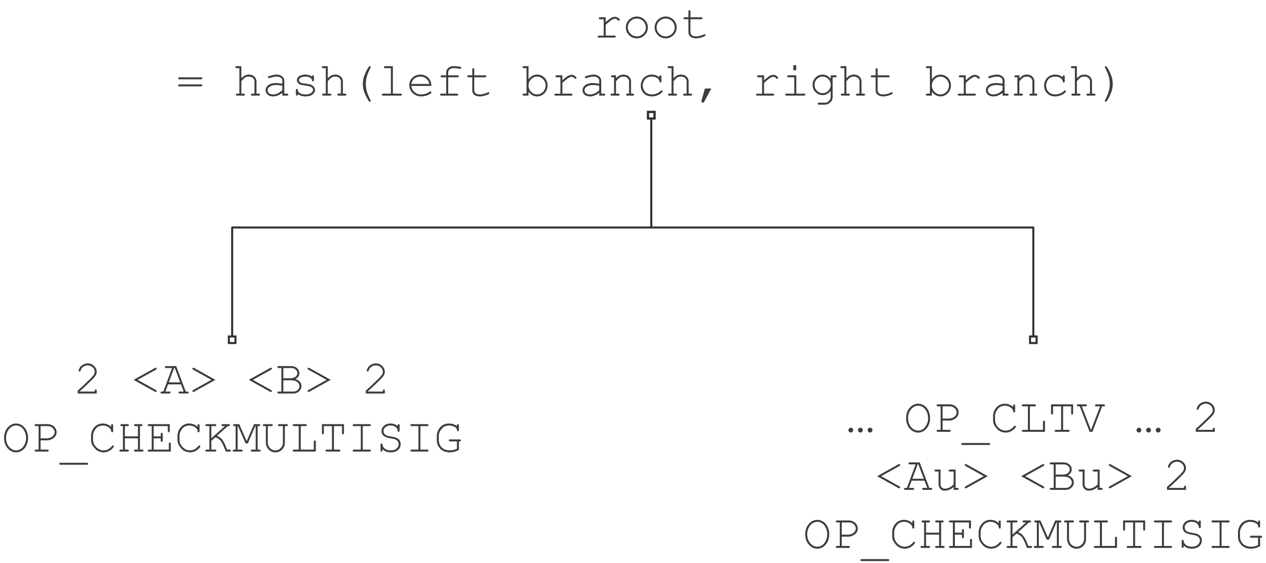
```
secret = adaptor_sig + Alice_partial_sig - combined_sig
```

Example: eltoo updates

Script	**Meaning**

`OP_IF`
`  2 <A> <B> 2 OP_CHECKMULTISIG`
`OP_ELSE`
`  ... OP_CLTV ...`
`  2 <Au> <Bu> 2 OP_CHECKMULTISIG`
`OP_ENDIF`
	Can be spent either by 2-of-2 of pubkeys A and B or by attaching another update transaction

Merkleized Abstract Syntax Trees (MAST)



root
= hash(left branch, right branch)

2 <A> <B> 2
OP_CHECKMULTISIG

… OP_CLTV … 2
<Au> <Bu> 2
OP_CHECKMULTISIG

Merkleized Abstract Syntax Trees (MAST)

Script	Witness

`root OP_MAST(?)`	`<script>`
	`<merkle proof>`
	`<witness>`

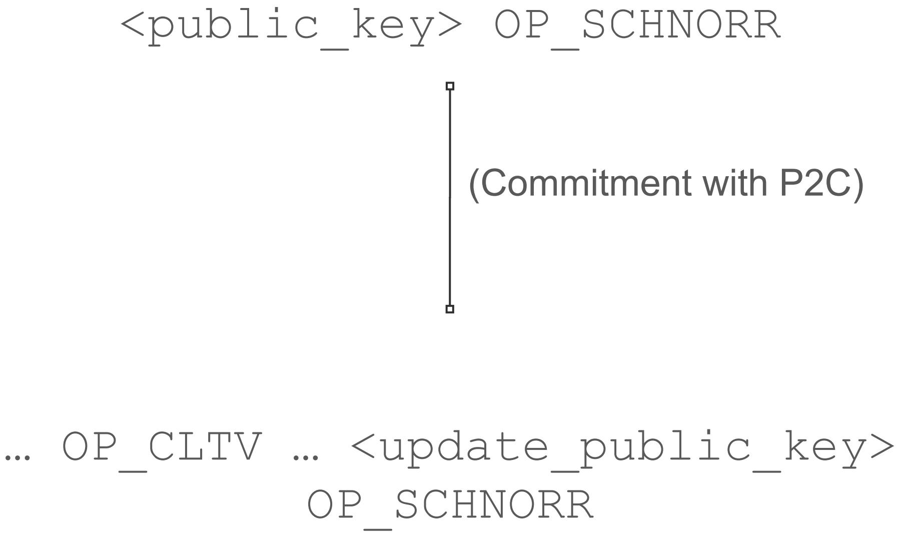
- MAST usage is revealed to blockchain observers
- data overhead because there’s no default branch

Pay-To-Contract (P2C)

- Idea: put commitment to data into a public key
- Original use case: allow sender to prove in private what purpose of payment was
    - F.e. address commits to data “this public key is used to buy a hat”

1. Generate normal public key `P = x*G`
2. Create new public key `Q` from `P` and `C` as `Q = P + hash(P,C)*G`
3. Commit to `C` by putting `Q` in the blockchain
4. Now can
    a. Sign for `Q` because know private key `x + hash(P,C)`
    b. Reveal `P` and `C` to prove that `Q` commits to `C`

Taproot & Schnorr



```
<public_key> OP_SCHNORR
```

(Commitment with P2C)

```
… OP_CLTV … <update_public_key>
          OP_SCHNORR
```

Taproot Assumption:
Interesting scripts have almost always a logical top level branch that allows satisfaction of the contract with nothing other than a signature by all parties

Taproot & Schnorr

Taproot: Add a new consensus rule that **additionally** allows spending a coin by proving that the input **public key committed to a script** and providing the witness for that script.

Taproot & Schnorr

Script	Witness	Meaning

`<pubkey>`
`OP_SCHNORR`
	`<signature>`	… (as before) …

	`<… OP_CLTV …`
	`<update_public_key>`
	`OP_SCHNORR>`
	`<P>`
	`<signature>`
		Uncooperative close

Conclusion

- Adding Schnorr Signatures to Bitcoin allows cheaper and more private Lightning channels
    - With adaptor signatures cheaper and more private uncooperative closings, routing privacy, proof of payment
- Adding Taproot to Bitcoin allows cheaper and more private uncooperative channel closings
- Status
    - Schnorr standardization BIP in review stage
    - Schnorr softfork BIP work-in-progress
    - Schnorr/taproot code WIP

References

- Schnorr BIP https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki
- MuSig https://eprint.iacr.org/2018/068.pdf
- Adaptor Sigs https://eprint.iacr.org/2018/472.pdf
- Blind Signatures in Scriptless Scripts https://nickler.ninja/slides/2018-bob.pdf
- Eltoo https://blockstream.com/eltoo.pdf
- Taproot https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-January/015614.html

Q&A

- slides: https://nickler.ninja/slides/2018-hackday.pdf
- questions?