An overview of Schnorr / Taproot

Kostas Karasavvas
@kkarasavvas

What?

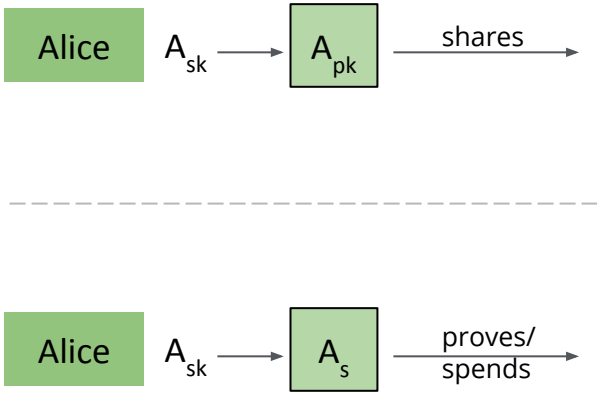
- Schnorr Signatures
  - new signature algorithm supported
- Musig
  - multi-signature orchestration of Schnorr signatures
- Taproot (P2TR)
  - Tapscript
  - Taptweak
  - Taptree

Why?

- Scalability / Efficiency / Bandwidth
- Privacy
- Extensibility

Signatures - Funds locked/unlocked

- Public Key Cryptography
  - sk -> pk
- Public key (address) is shared
- People send funds
- Alice proves ownership

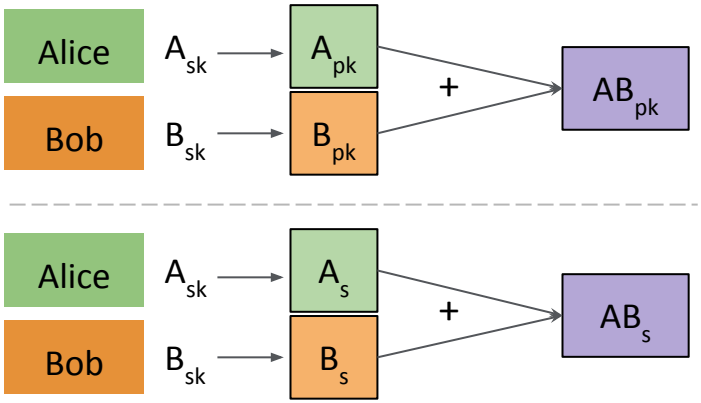


Schnorr Signatures - Basic

- New Signature Scheme
  - Compatible with ECDSA's secp256k1
- Deployed and used with Taproot upgrade
  - Uses Segwit v1 output scripts
  - Does not affect existing output types
- 11% smaller signatures
  - 64 bytes instead of ~71 bytes
  - Saves space in the block
    - Scalability / Disk space / Network bandwidth

Schnorr Signatures - Aggregation

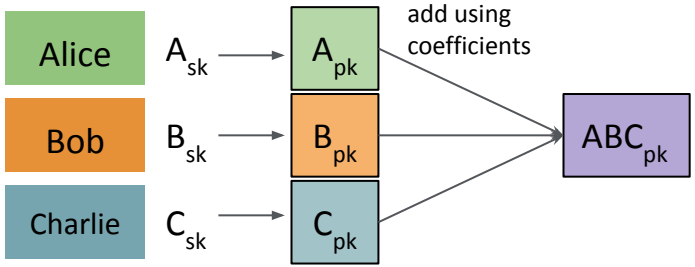
- Signature Aggregation
  - $A_{sk} \Rightarrow A_{pk}$ and $B_{sk} \Rightarrow B_{pk}$
  - $A_{pk} + B_{pk} = \mathbf{AB_{pk}}$
  - $A_{s} + B_{s} = \mathbf{AB_{s}}$



  - *Privacy* - multi-signature outputs could masquerade as simple Schnorr signatures
  - *Scalability* - saves significant space in the block
  - *Efficiency* - validates a single signature instead of several
  - Signature aggregation needs orchestration
    - Musig

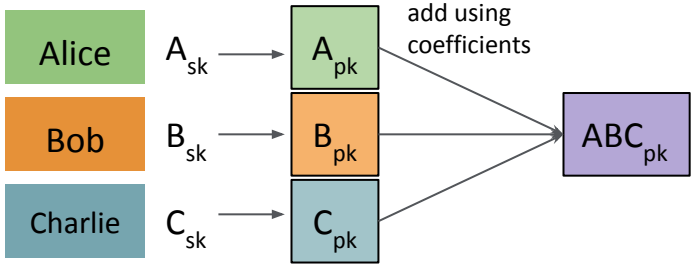
Schnorr Signatures - Musig

- Public keys are combined to create a new public key
- The new public key is shared for payments

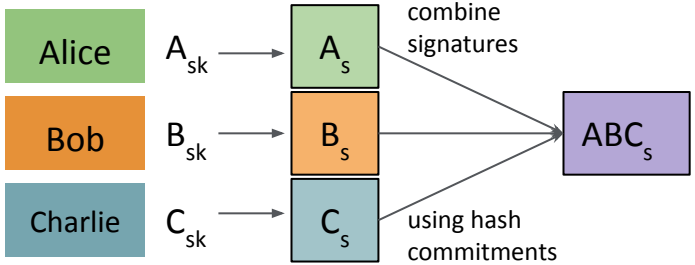


Schnorr Signatures - Musig

- Public keys are combined to create a new public key
- The new public key is shared for payments



- To spend the funds each participant signs with their own key
- The combined signatures will correspond to the aggregated public key



Schnorr Signatures - Future potential

- Cross-input signature aggregation
  - several complexities
- Cross-transaction signature aggregation
  - several complexities

Taproot (P2TR)

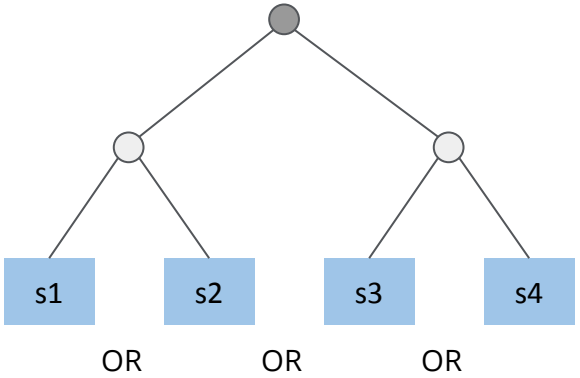
- Default spending path
  - Single key
  - Or Musig (*privacy*)

Default spending path

Taproot public key

- One or more hidden alternative scripts
  - Cannot know if there are alternatives (*privacy*)
  - Only reveal specific script when spending (*privacy*)
  - Very large scripts possible (*scaling*)

Alternative spending path(s)
(OR conditions)



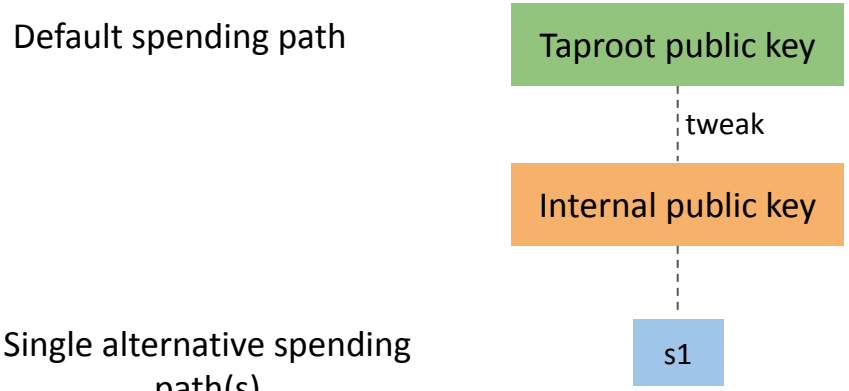
Taproot - Tapscript

- Script allowed in taproot alternative spending paths (Segwit v1)
  - Signature opcodes now validate Schnorr signatures
  - Multisig opcodes replaced with CHECKSIGADD opcode

    Has several reserved opcodes to add new functionality later (NOP)
    - *extensibility*
  - Allows versioning (alternative path scripts could be of different version)
    - *extensibility*

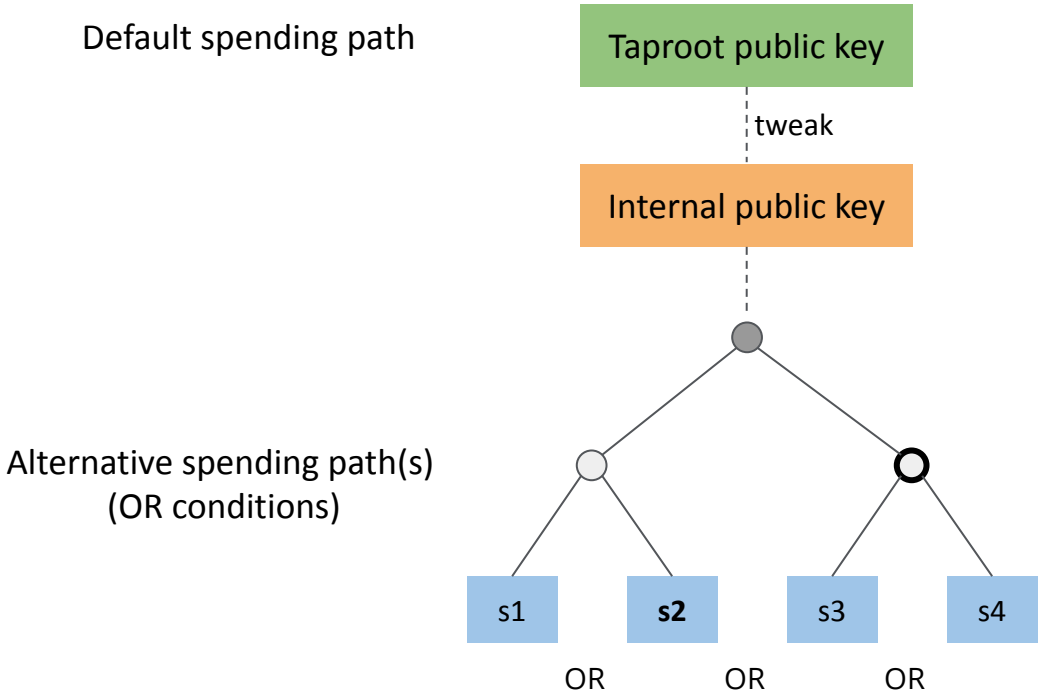
Taproot - Taptweak

- Public key is tweaked to produce the Taproot public key
  - EC adding a commitment
  - Commitment is the tapscript hash
- Private key is also tweaked
  - We use the tweaked private key to sign the transaction
- To spend alternative script
  - Unlocking tapscript
  - Tapscript
  - Internal public key



Taproot - Taptree

- To spend one of several alternative paths
    - Unlocking tapscript
    - Tapscript itself
    - Internal public key
    - Inclusion proof
        - Hashes of s2 and P(s3-s4)



Conclusion

- Scalability / Efficiency / Bandwidth
- Privacy
- Extensibility

- Taproot Software Upgrade proposals
  - Schnorr Signatures (BIP-340)
  - A new SegWit v1 output type based on Taproot, Schnorr and Merkle branches (BIP-341)
  - Taproot script validation rules (BIP-342)

- UX Disruption

Thank You

@kkarasavvas

Thessaloniki's Bitcoin and Blockchain Tech Meetup

https://www.meetup.com/BlockchainGreece-1/
@Thess_Bitcoin

Python Bitcoin Library (FOSS)

https://github.com/karask/python-bitcoin-utils

Bitcoin Programming Book (CC)

https://github.com/karask/bitcoin-textbook