Blockstream

Research
by Blockstream

Mass Adoption, But Decentralized

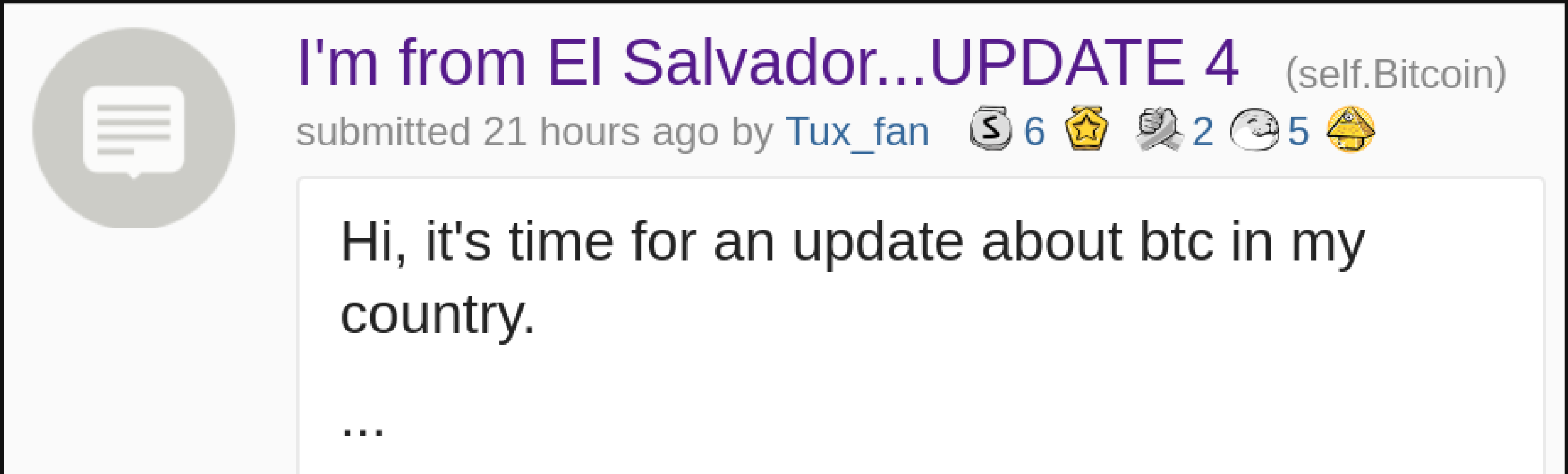
Jonas Nick
nickler.ninja

Blockstream Research

- research focus:
  - signature schemes
  - scripting languages (Miniscript, Simplicity)
- for the Bitcoin protocol, wallets, Elements sidechain, Lightning Network, etc...
- and contributions to open source projects like Bitcoin Core, libsecp, rust-bitcoin, minimint and many more

Blockstream

Research
by Blockstream

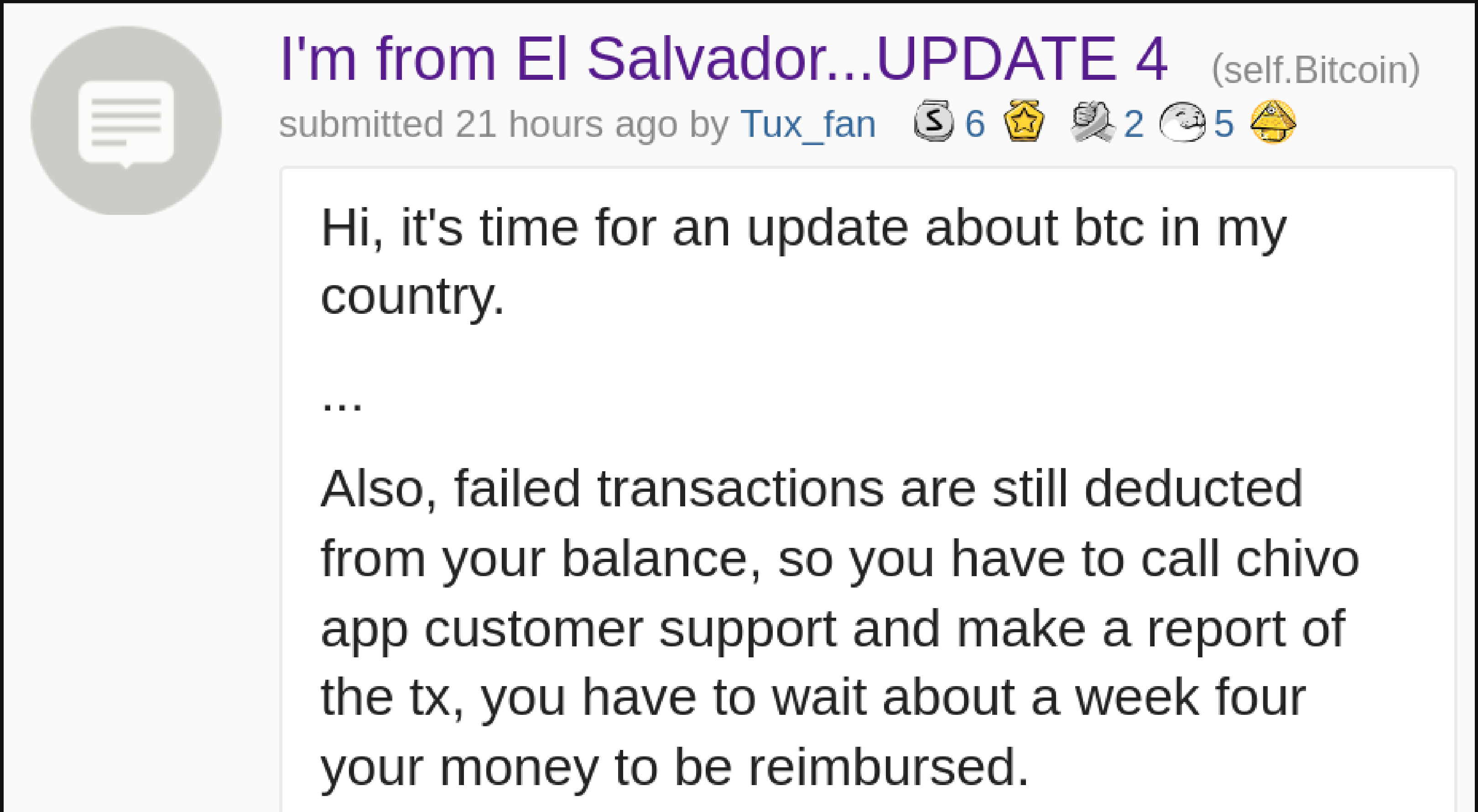


I'm from El Salvador...UPDATE 4 (self.Bitcoin)

submitted 21 hours ago by Tux_fan 6 2 5

Hi, it's time for an update about btc in my country.

...



I'm from El Salvador...UPDATE 4 (self.Bitcoin)

submitted 21 hours ago by Tux_fan 6 2 5

Hi, it's time for an update about btc in my country.

...

Also, failed transactions are still deducted from your balance, so you have to call chivo app customer support and make a report of the tx, you have to wait about a week four your money to be reimbursed.

Is this Bitcoin?

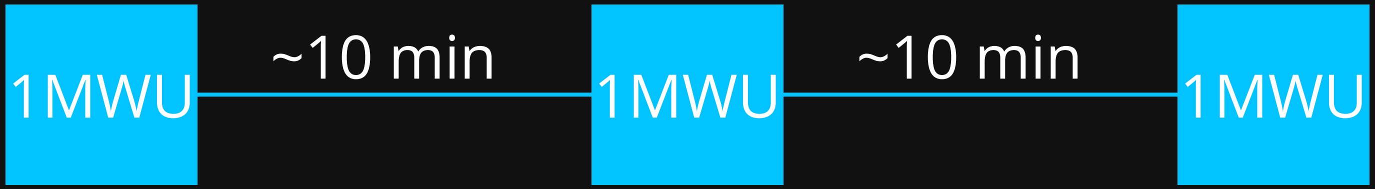
What is Bitcoin?

> *I hope it's obvious it was only the centrally controlled nature of those systems that doomed them. I think this is the first time we're trying a decentralized, non-trust-based system.*
>
> \- Satoshi

Resists powerful actors through decentralization

Running a Bitcoin Node

- Bitcoin is a protocol that is verified by a network of *full nodes*
- nodes are in consensus about the history of transactions determined by protocol rules and proof-of-work
- BUT: creating transactions doesn't work for everyone, since the throughput is limited



1MWU ~10 min 1MWU ~10 min 1MWU

MWU = Mega Weight Unit

Case Study: Blocksize Increase

- Bitcoin governance emerges through the software users run on their computers
- hence, could increase size limit of blocks
- but this increases cost to run full node

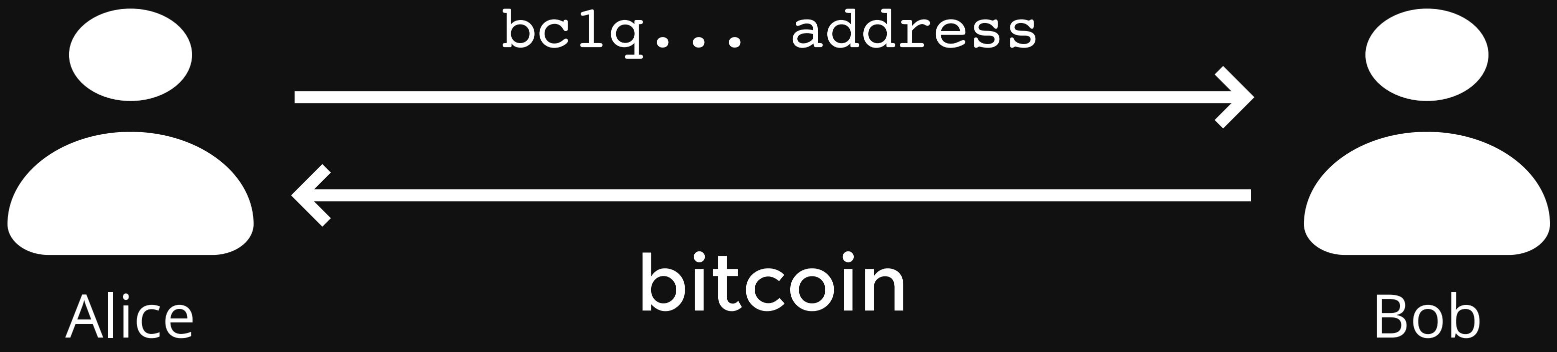
After an astounding victory, the small block narrative, that end users had to agree to protocol rule changes, was finally seen as compelling.

- Jonathan Bier, The Blocksize War

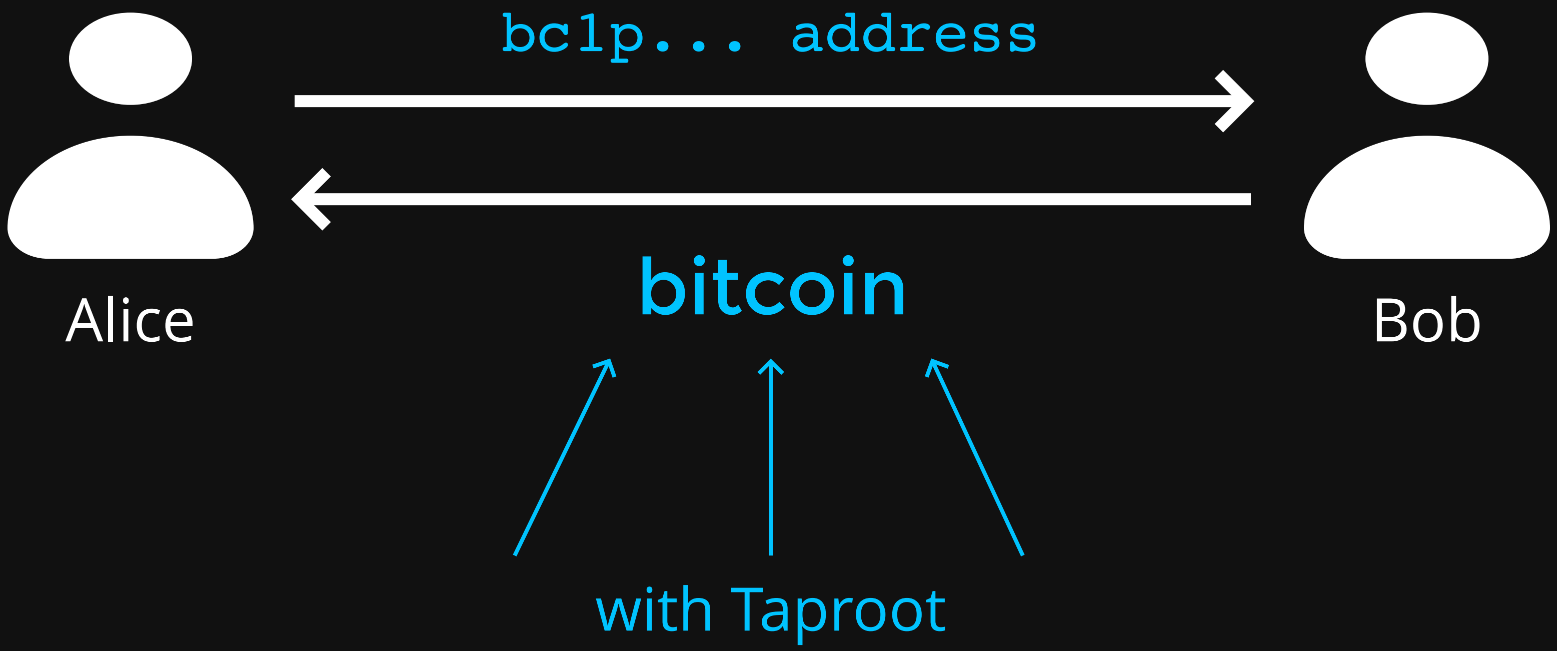
Instead, make most of existing constraints:

The Taproot Upgrade

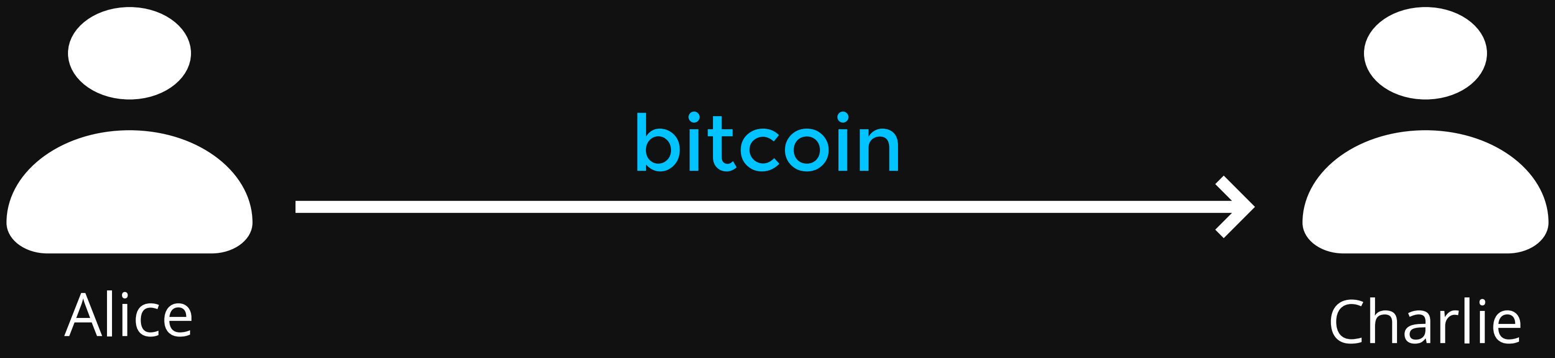
Bitcoin Today



Bitcoin after Taproot Upgrade



Schnorr Sigs in Taproot



bitcoin

Alice

Charlie

Authorization of Transactions:

In blockchain: ~~ECDSA~~ Schnorr signature of Alice's public key

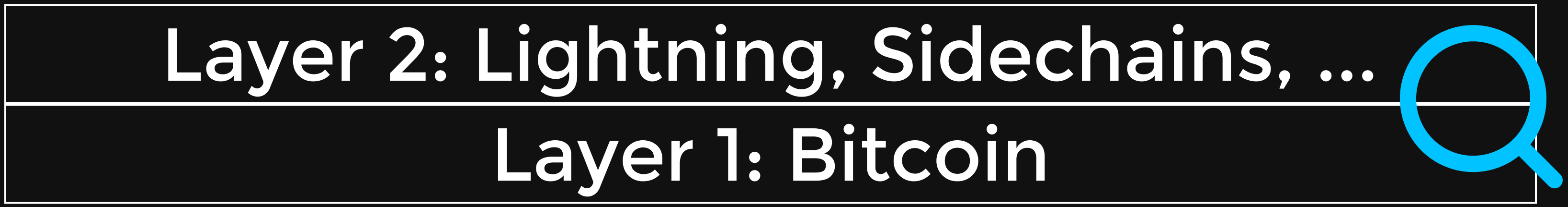
Schnorr Batch Verification

- **without:** full node must verify every signature in blockchain individually
- **with:** full node verfies batch of signatures at once
- example batch = 10,000: verification is twice as fast
- therefore, full node cost reduced
  - key requirement for principle of resistance
- status: proof-of-concept implementation exists

Still, on-chain transactions don't scale

Layered Scaling

Idea: use Bitcoin as settlement layer, build protocols on top with different trust assumptions

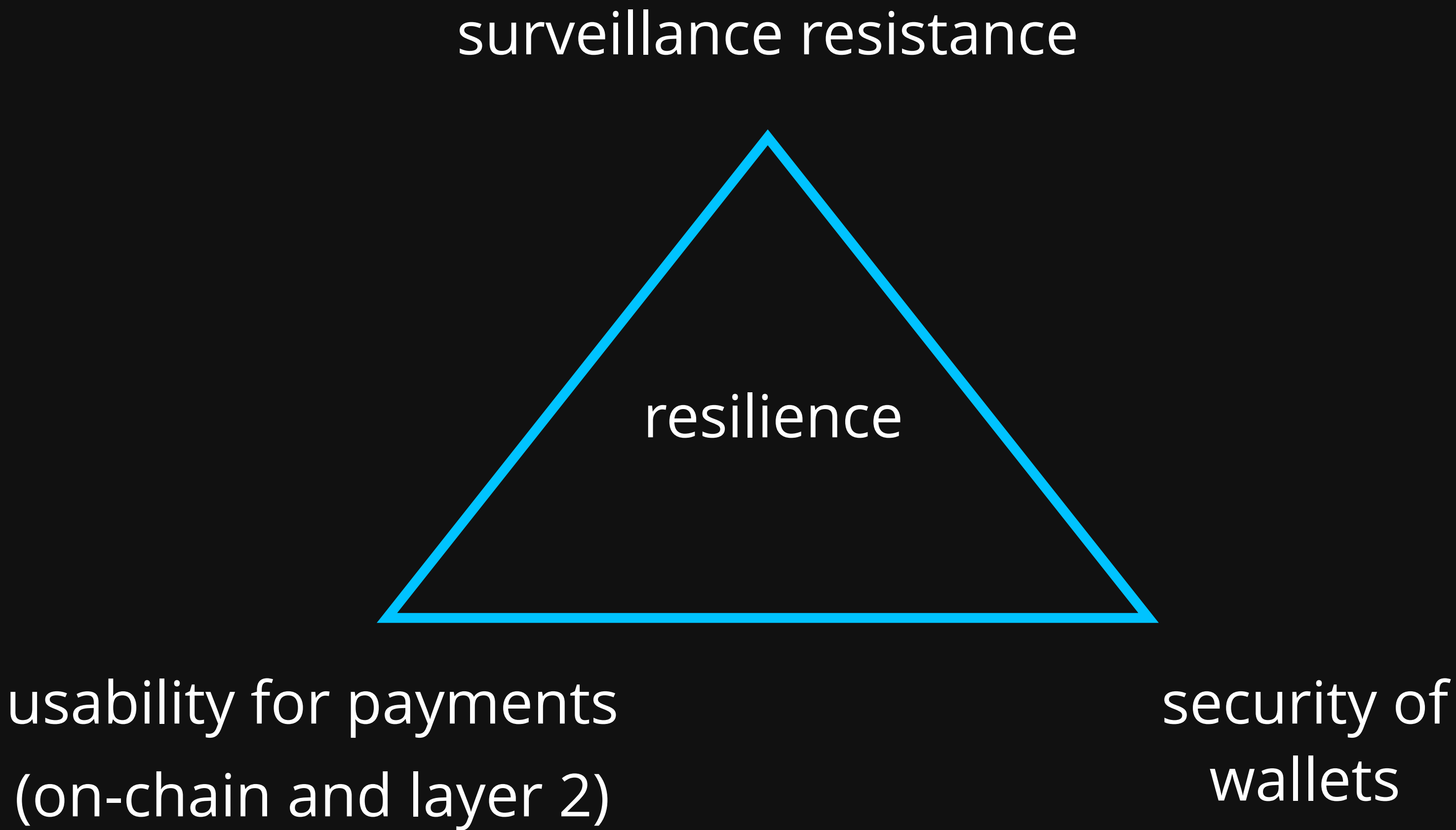


Layer 2: Lightning, Sidechains, ...

Layer 1: Bitcoin

Layer 2	(Multiparty-) Payment Channels	Sidechains	Federated E-Cash
off-chain primitives	Batch Verify	Key Agg	Adaptor Sig
on-chain	Taproot	½ Sig Agg	Sig Agg

Principles



surveillance resistance

resilience

usability for payments
(on-chain and layer 2)

security of
wallets

Indistinguishability

- transactions that are part of complex protocols look the same as simple payments to an observer

Normal Payment?

Sidechain?

Transaction

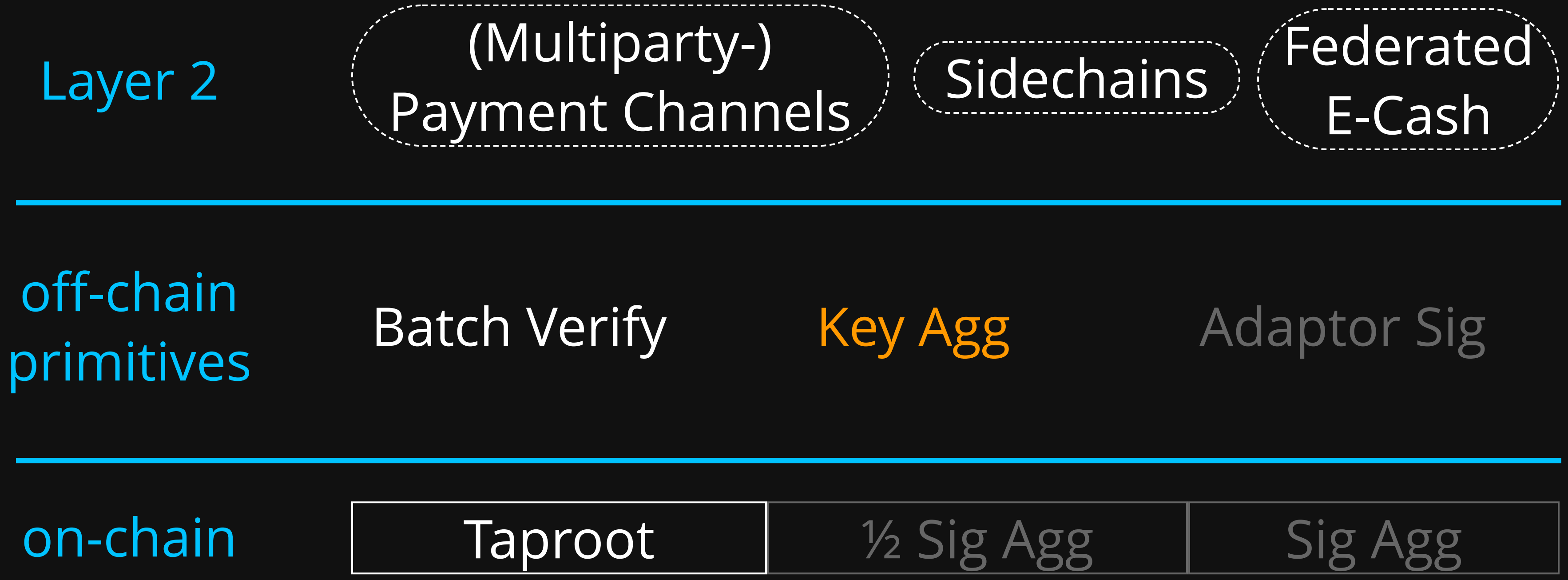
Multisig?

Lightning?

- makes spying with blockchain more difficult
- layer 2 (L2) and multisig cheaper

Taproot, in short

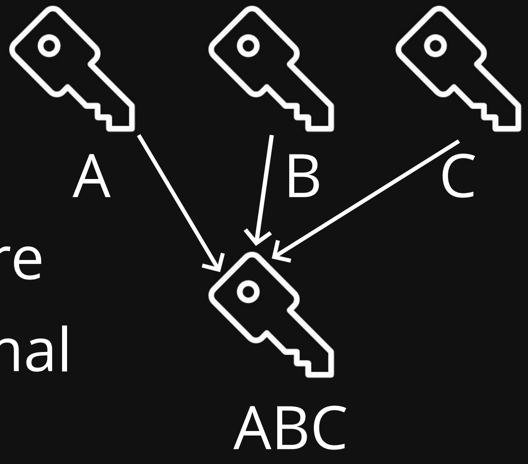
- example: coin that can be spent by
    - `or(`**`Alice`**`,and(`**`Bob`**`, older(1000))`
    - "Alice immediately or Bob after 1000 blocks"
- taproot allows hiding unexecuted branches
- if Alice spends the coin, it looks like an ordinary payment



Layer 2	(Multiparty-) Payment Channels	Sidechains	Federated E-Cash
off-chain primitives	Batch Verify	Key Agg	Adaptor Sig
on-chain	Taproot	½ Sig Agg	Sig Agg

Multi Sig + Key Aggregation

- example: Alice, Bob and Charlie have a 2-of-3 multisig wallet
- **without:** All three keys and two signatures must be written to the chain
- **with:**
  - one aggregate key
  - cooperatively create single signature
- therefore, indistinguishable from normal transactions

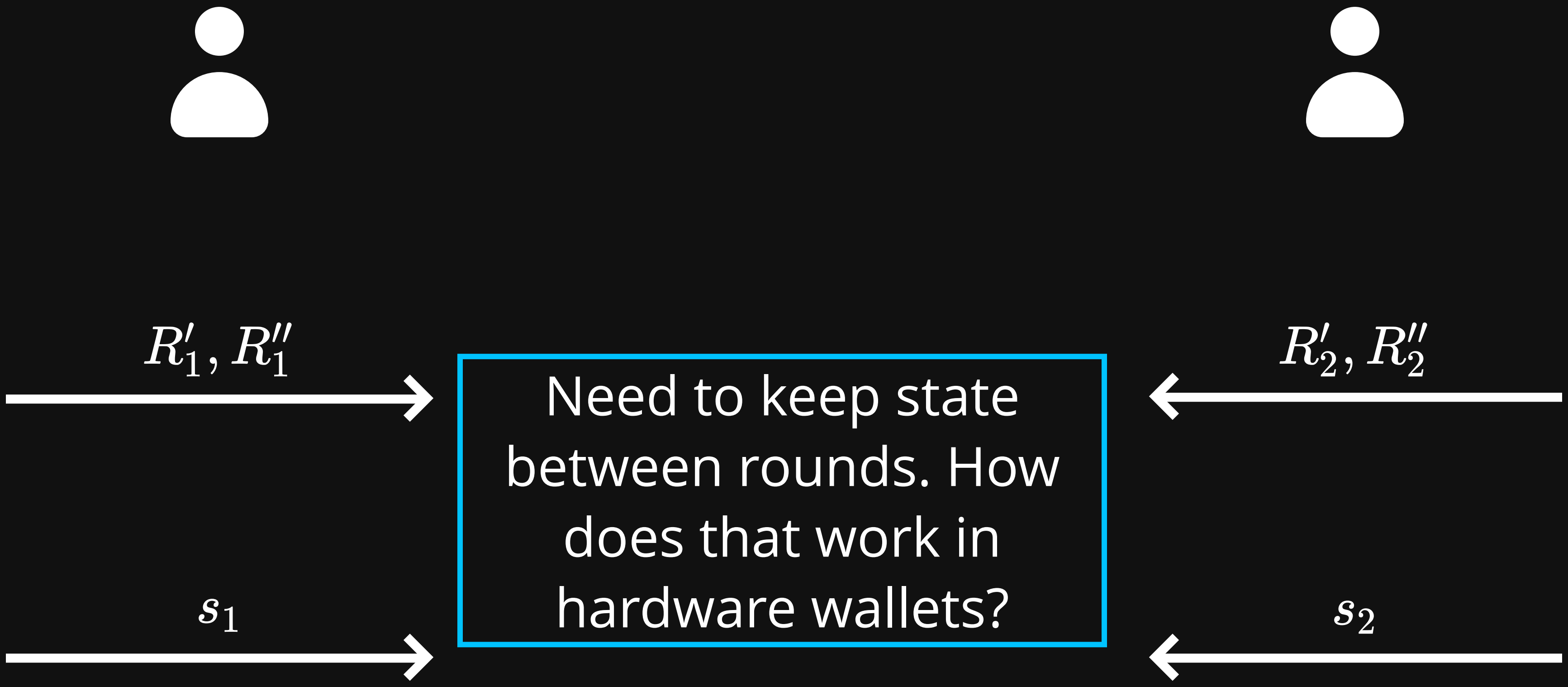


MuSig vs MuSig2 vs FROST

- **MuSig**: n-of-n multisignatures*
  - status: replaced by MuSig2
- **MuSig2**: n-of-n multisignatures*
  - needs less communication
    - in particular useful in Lightning routing
  - status: implementation & spec. in progress
- **FROST**: m-of-n threshold signatures
  - example: -75% cost of 11-of-15 federation
  - interactive setup: key shares of other parties need to be backed up securely
  - status: implementation of prototype in progress

*t-of-n with taproot in best case

MuSig2:
2-of-2 example



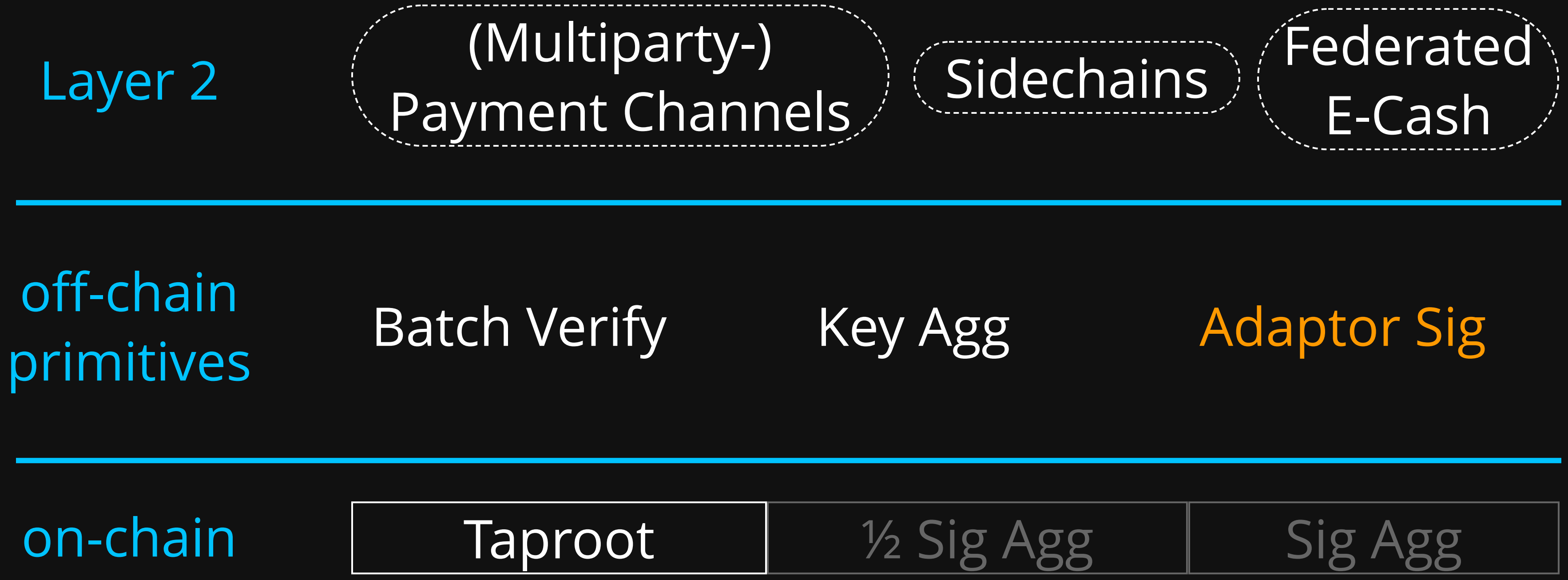
R'_1, R''_1

R'_2, R''_2

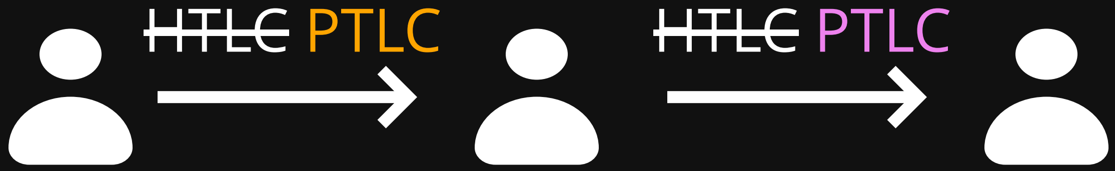
Need to keep state between rounds. How does that work in hardware wallets?

s_1

s_2



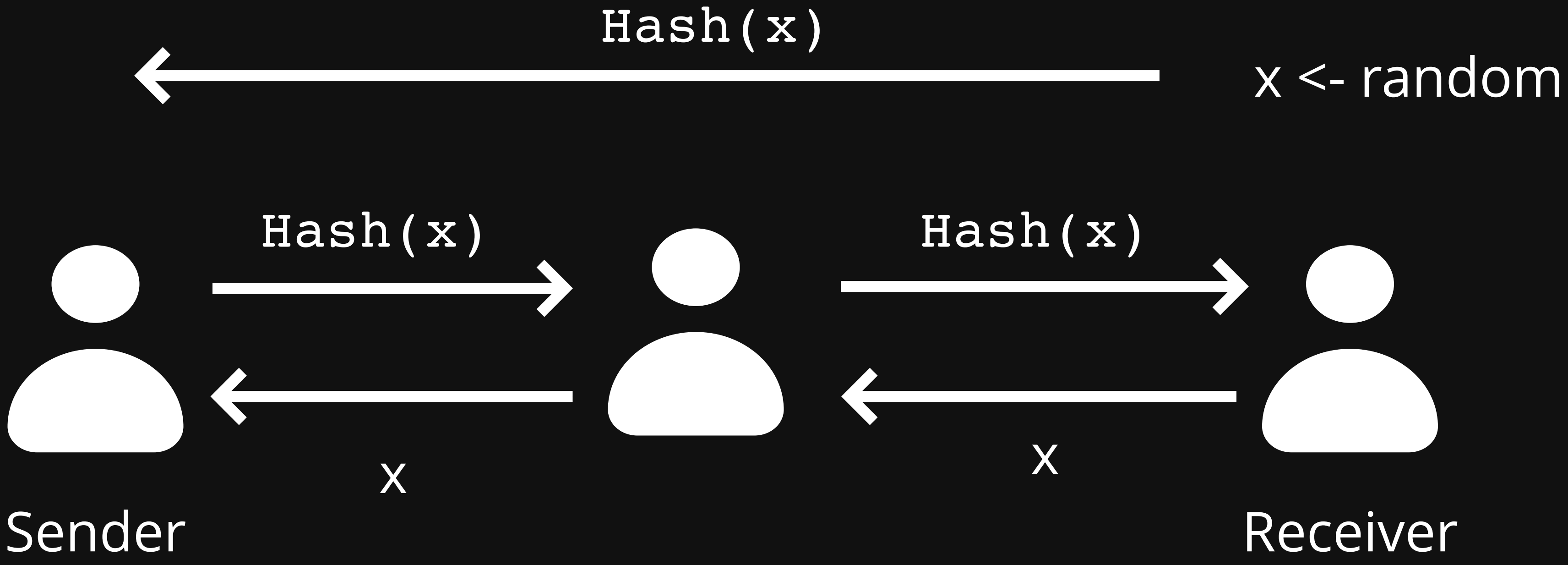
Lightning:



~~HTLC~~ PTLC

~~HTLC~~ PTLC

Lightning Network

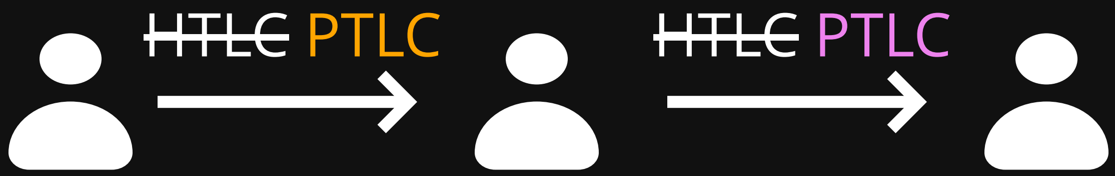


- HTLC: Hash Timelocked Contract, visible on-chain, same on every hop of route

Adaptor Signatures

- example: atomic swaps like Lightning payment, DLCs, peerswap
- **without:** requires on-chain hash
- **with:** instead, off-chain adaptor signature

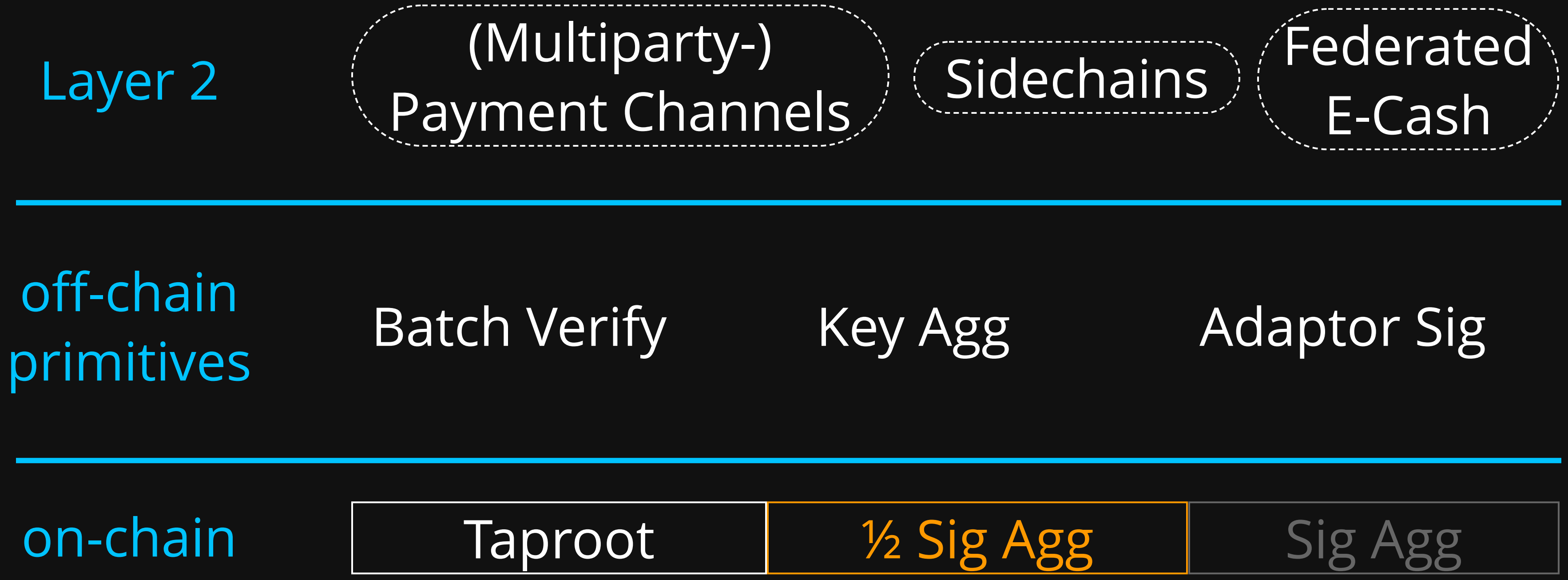
Lightning:



~~HTLC~~ PTLC

~~HTLC~~ PTLC

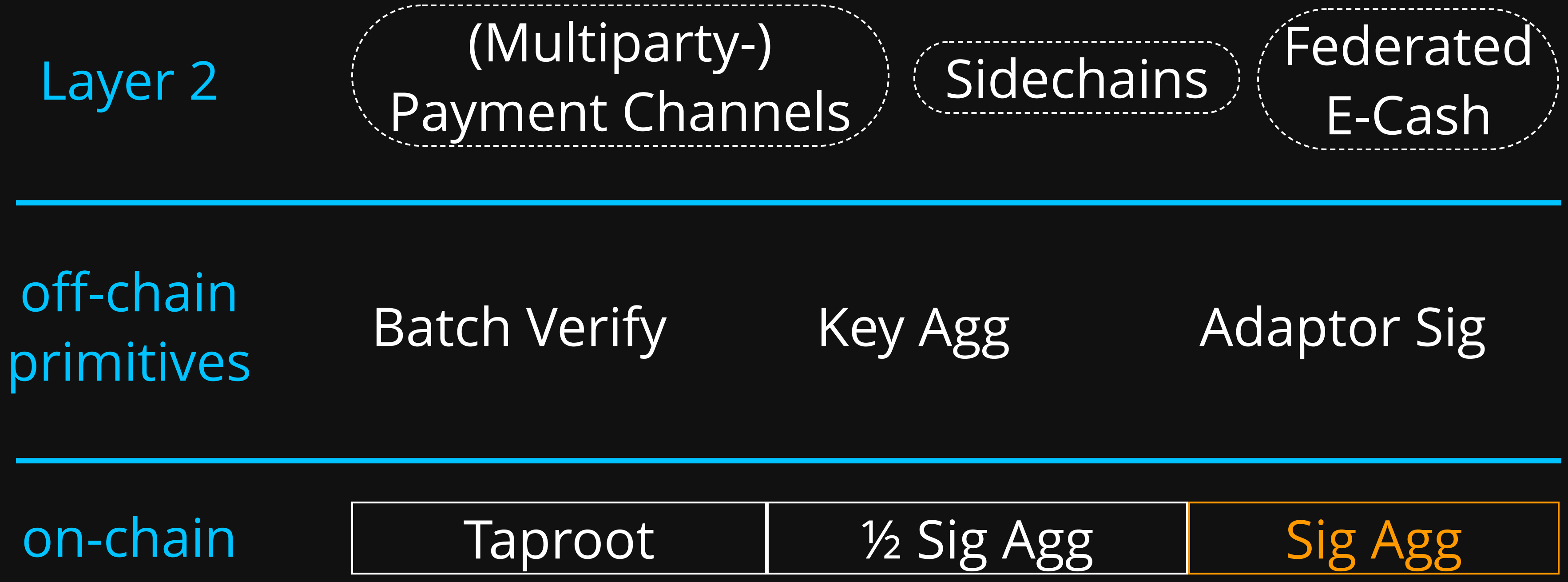
- PTLC: Point Timelocked Contract



Layer 2	(Multiparty-) Payment Channels	Sidechains	Federated E-Cash
off-chain primitives	Batch Verify	Key Agg	Adaptor Sig
on-chain	Taproot	½ Sig Agg	Sig Agg

Schnorr Half Aggregation

- **with:** blocks contain (at least) one signature per spent coin
- **with:** contain a "half"-aggregate signature, that is half as big as the sum of individual sigs
  - $\mathsf{Aggregate}(\mathsf{sig}_1, \ldots, \mathsf{sig}_n) \rightarrow \mathsf{sig}$
  - non-interactive
- example: 10 half-aggregate signatures take same space as 6 ordinary sigs
- therefore, more transactions per block
- status: research, requires softfork



Layer 2

(Multiparty-) Payment Channels

Sidechains

Federated E-Cash

off-chain primitives

Batch Verify

Key Agg

Adaptor Sig

on-chain

Taproot

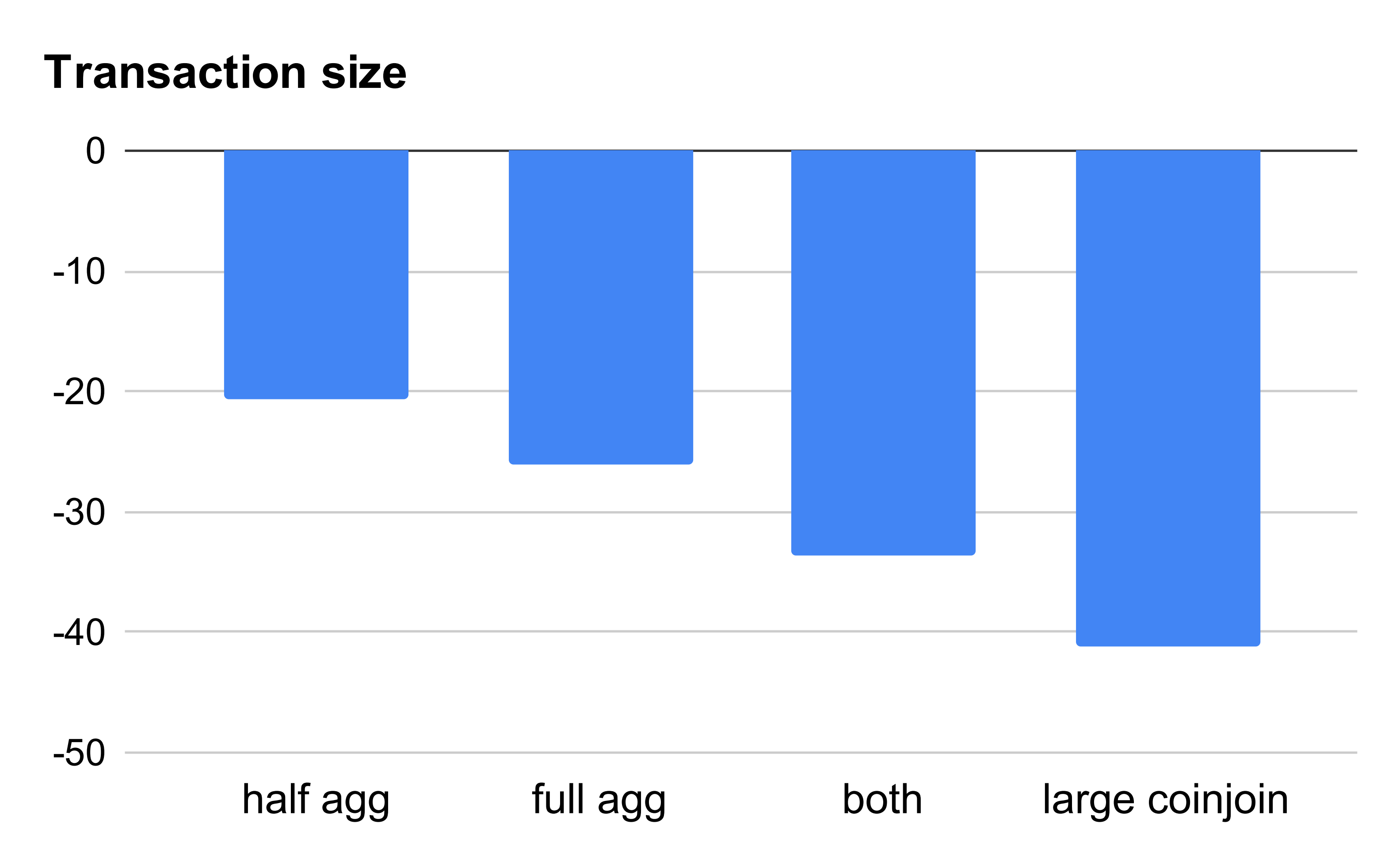
½ Sig Agg

Sig Agg

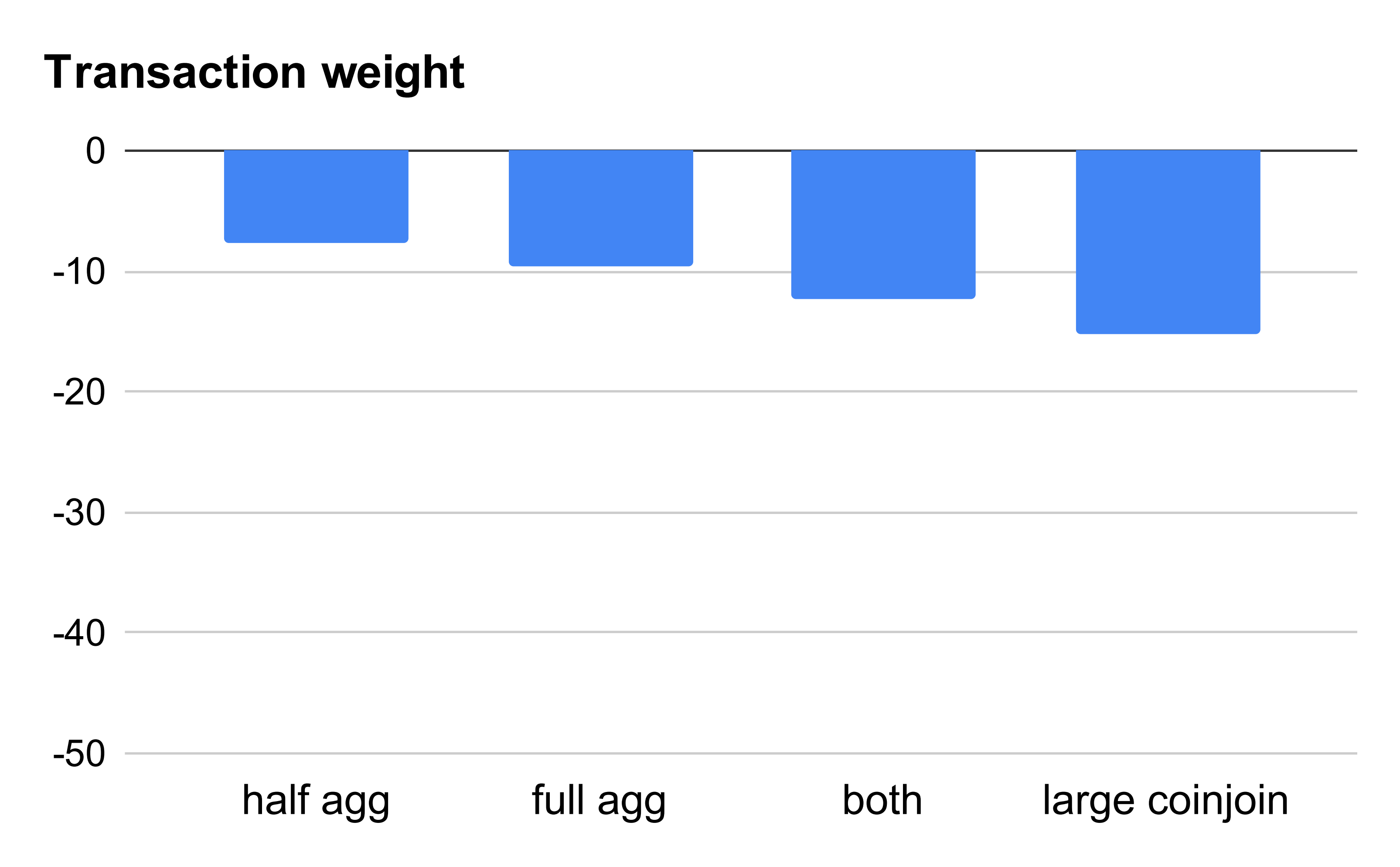
Schnorr Full Aggregation

- **without:** transactions contain (at least) one signature per spent coin
- **with:** transactions contain exactly one, aggregate sig
- size same as ordinary Schnorr signature
- signing is interactive
- smaller transactions, incentive for CoinJoin
- status: research, requires softfork

Aggregate Size



Aggregate Weight



Conclusion

- nickler.ninja/slides/
- unclear what trade-offs are going to be made for mass-adoption
- staying resilient takes precedence
  - surveillance resistance
  - usability for payments
  - wallet security
- indistinguishability
- key aggregation: aggregation in multisig wallet
- sig aggregation: aggregation across wallets

Conclusion

- Get involved
    - bitcoinops.org
    - bitcoin-dev mailing list
    - lightning-dev mailing list
    - bitcoinproblems.org

Summary

Protocol	Application	Benefits	Status
Batch verify	Faster verification	Full node ressources	Prototype implementation
TR Merkle tree	Hidden script paths	Smaller txs, surveillance resistance	-
MuSig2	n-of-n multisig	Smaller txs, surveillence resistance	Specification in progress
FROST	t-of-n multisig	"	Implementation in progress
Recursive Key Agg	Multisig of multisig	L2 tricks	Research
Adaptor Sig	Swaps, HTLCs	Useful for L2, surveillance resistance	Specification in progress
Blind Sigs	Blind swap	Surveillance resistance	Applications where?
Thresh.BlindSigs	Federated E-cash	L2, Surveillance resistance	Implementation in progress
Half Agg	All txs	Smaller txs	Research, requires softfork
Full Agg	All txs	Smaller txs	Research, requires softfork