Der Schnorr Zoo

Protokolle auf Taproot & mehr

Jonas Nick
nickler.ninja

Blockstream

Blockstream Research

Blockstream

Blockstream Research

- Forschung an
  - Signaturverfahren
  - Scriptsprachen (Miniscript, Simplicity)
- für das Bitcoin Protokoll, Wallets, Elements Sidechain, Lightning Network, etc...

Blockstream

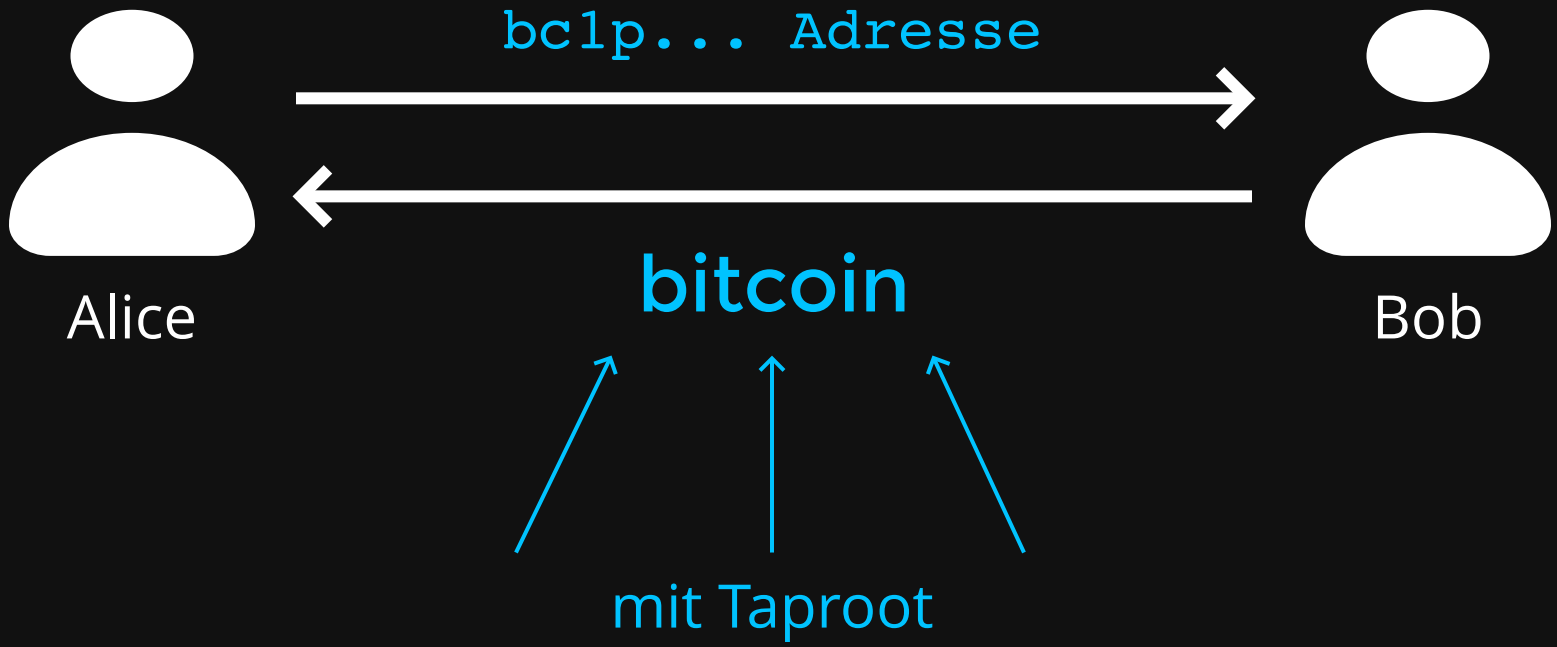
Blockstream Research

- Forschung an
  - Signaturverfahren
  - Scriptsprachen (Miniscript, Simplicity)
- für das Bitcoin Protokoll, Wallets, Elements Sidechain, Lightning Network, etc...
- Sowie Beiträge zu Open Source Projekten wie Bitcoin Core, libsecp, rust-bitcoin, uvm.

Blockstream

Bitcoin Heute

Bitcoin nach Taproot Upgrade



Taproot Aktivierung

- Taproot ist eingelockt, und wird Mitte November aktiviert.

Taproot Aktivierung

- Taproot ist eingelockt, und wird Mitte November aktiviert.
- Das Bitcoin Regelwerk wird von *Full Nodes* durchgesetzt, also Computern mit Software die die Blockchain verifizieren.

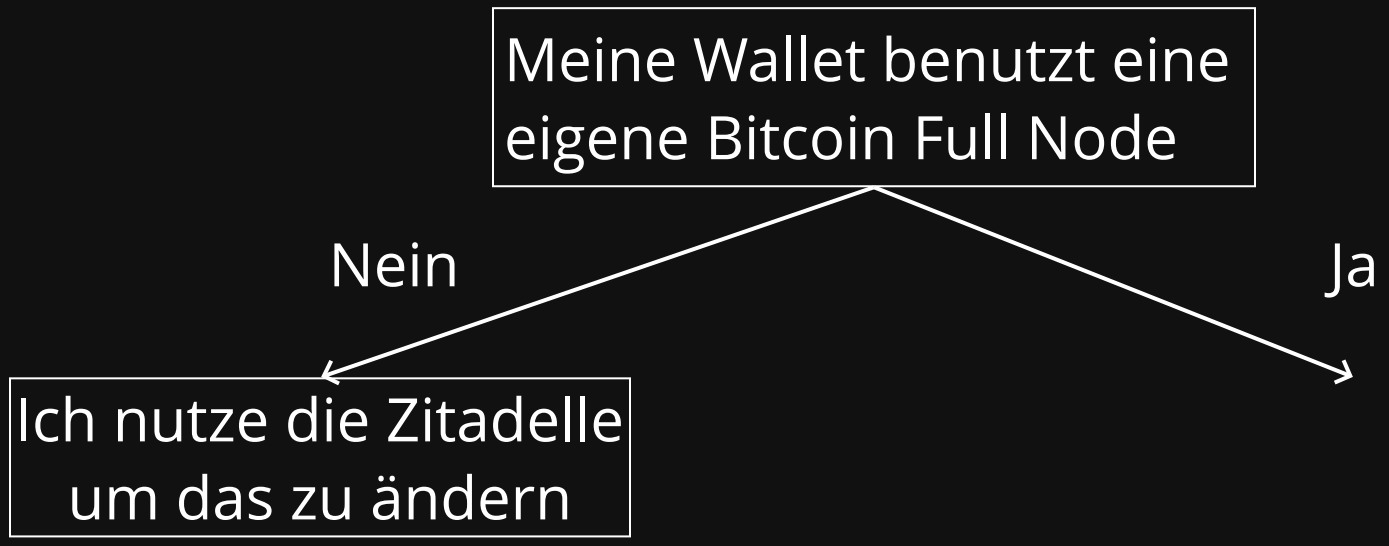
Taproot Aktivierung

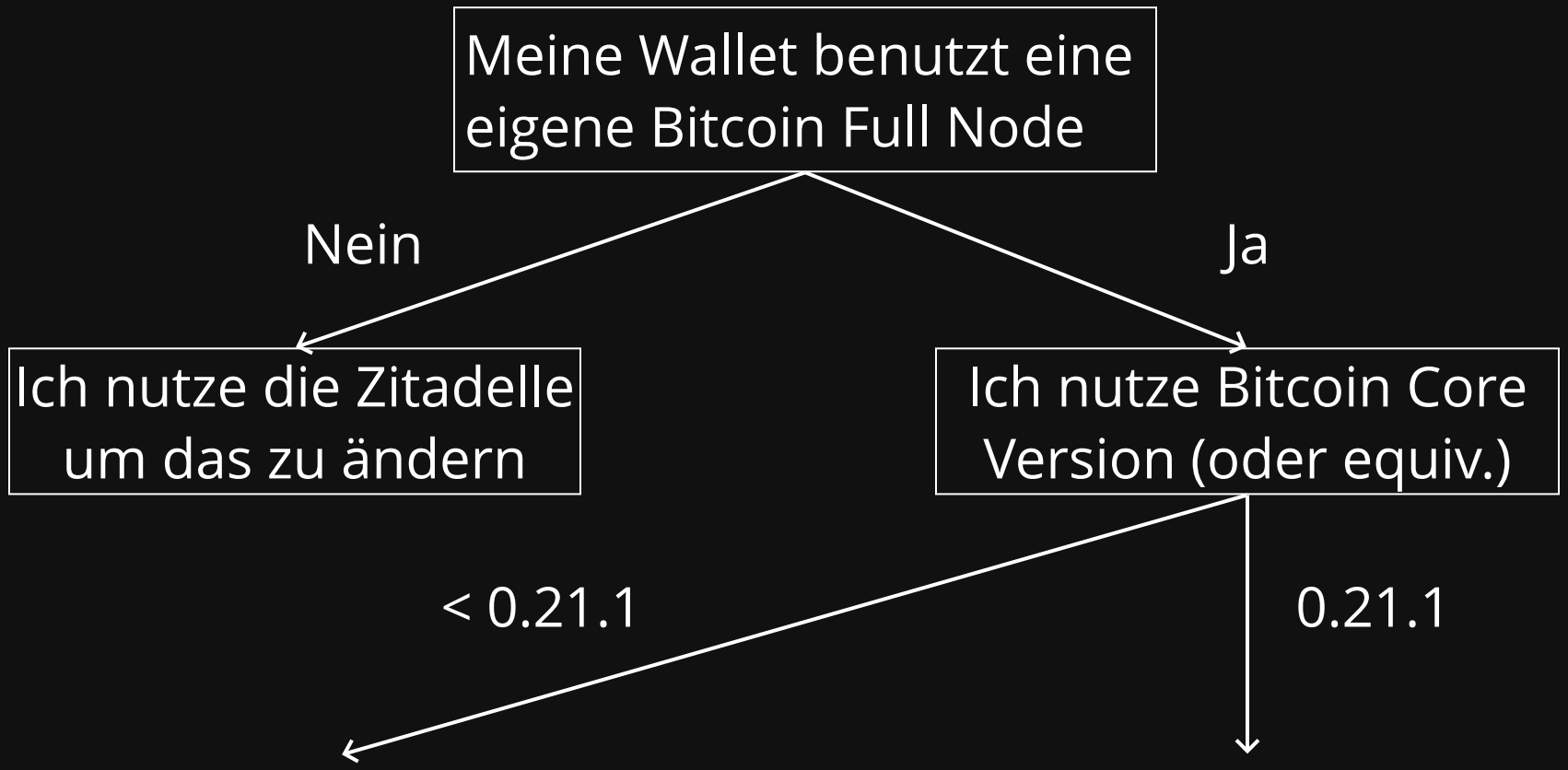
- Taproot ist eingelockt, und wird Mitte November aktiviert.
- Das Bitcoin Regelwerk wird von *Full Nodes* durchgesetzt, also Computern mit Software die die Blockchain verifizieren.
- Für reibungsloses Update, muss überwältigende Mehrheit Taproot unterstützen.

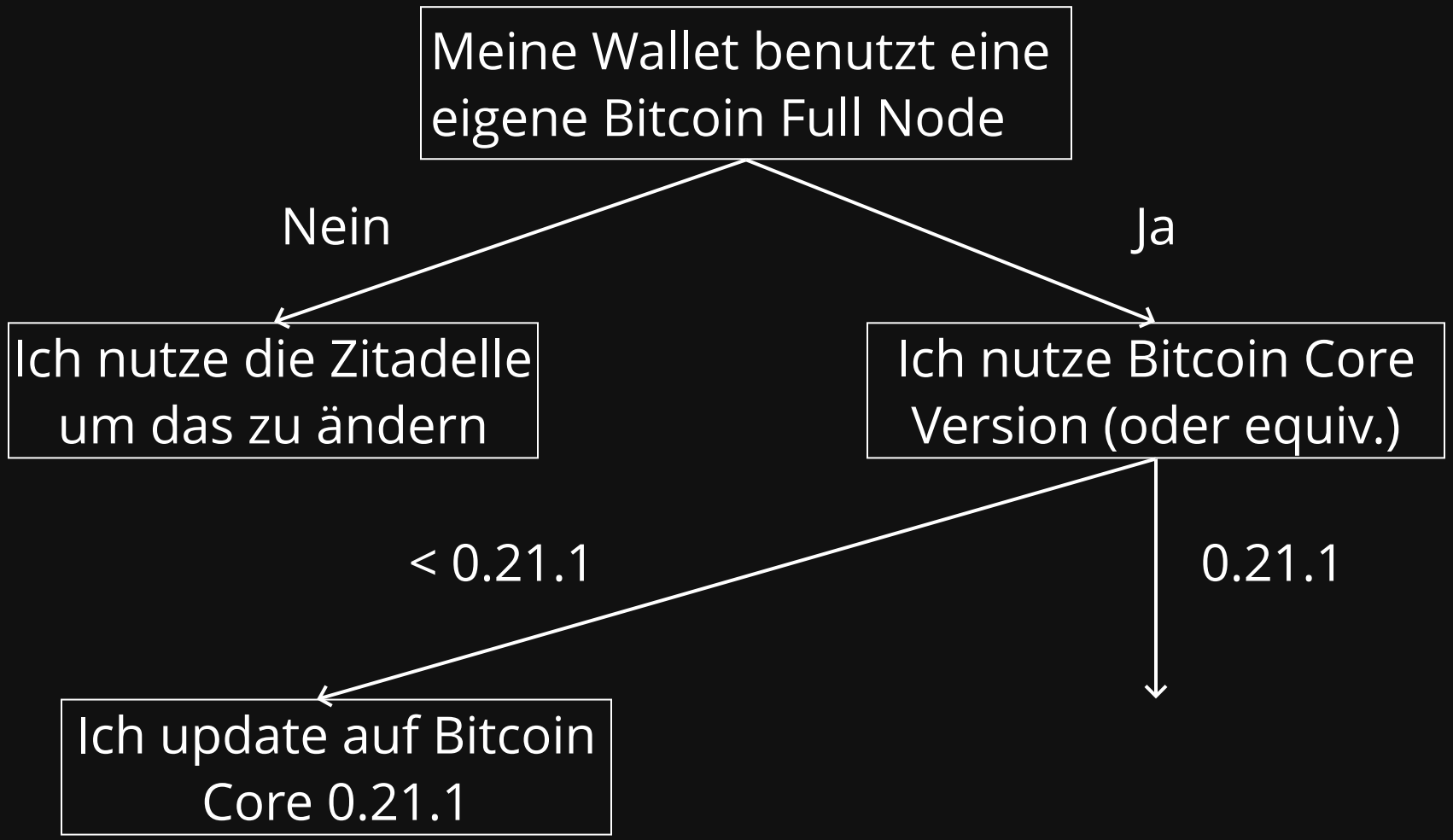
Taproot Aktivierung

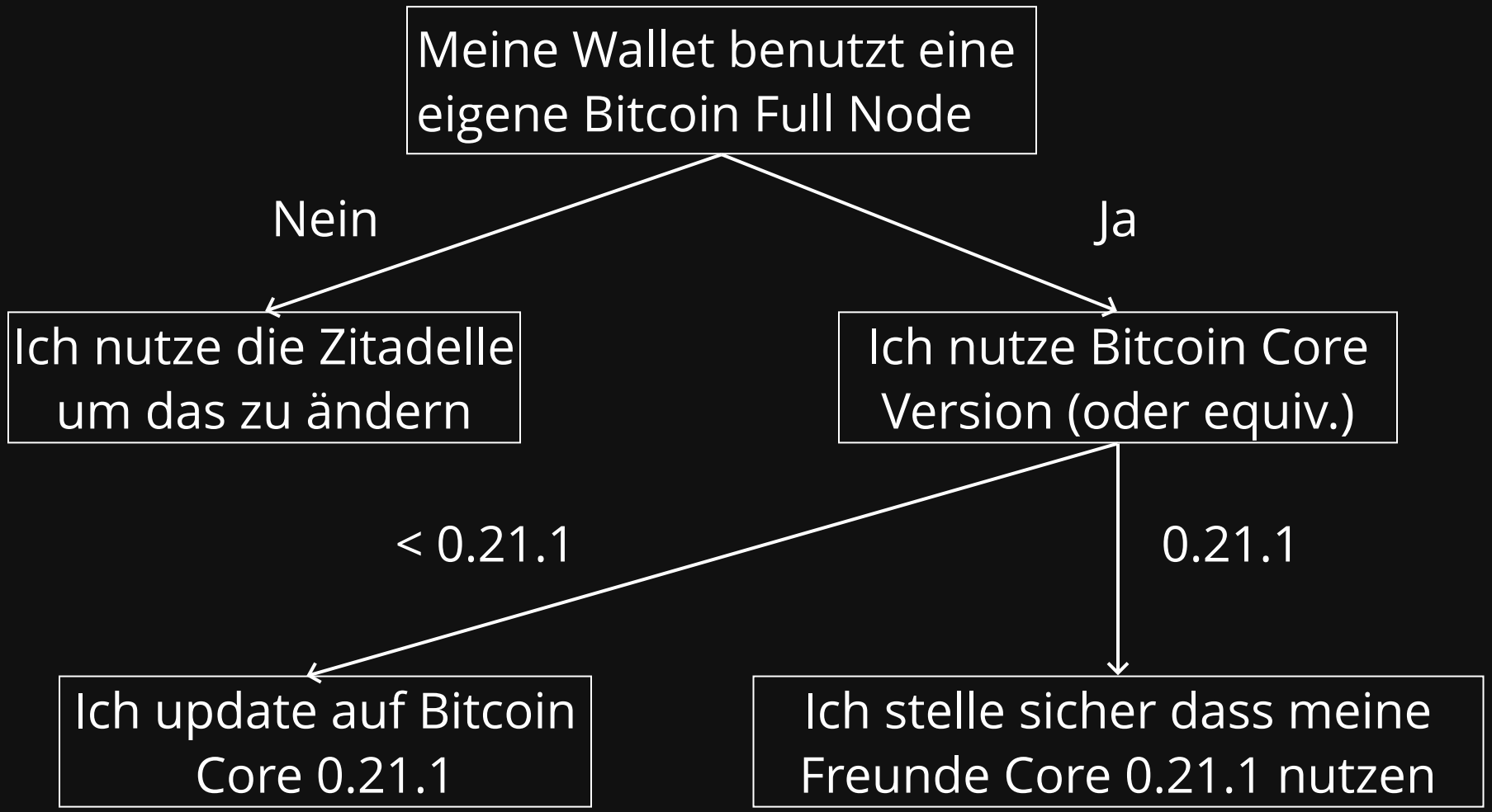
- Taproot ist eingelockt, und wird Mitte November aktiviert.
- Das Bitcoin Regelwerk wird von *Full Nodes* durchgesetzt, also Computern mit Software die die Blockchain verifizieren.
- Für reibungsloses Update, muss überwältigende Mehrheit Taproot unterstützen.
- Heute (2021-07-28) sind es nur etwa 36% bis 41% der Full Nodes.


Meine Wallet benutzt eine eigene Bitcoin Full Node
Nein
Ja


Meine Wallet benutzt eine eigene Bitcoin Full Node
Nein
Ja
Ich nutze die Zitadelle um das zu ändern


Meine Wallet benutzt eine eigene Bitcoin Full Node
Nein
Ja
Ich nutze die Zitadelle um das zu ändern
Ich nutze Bitcoin Core Version (oder equiv.)
< 0.21.1
0.21.1


Meine Wallet benutzt eine eigene Bitcoin Full Node
Nein
Ja
Ich nutze die Zitadelle um das zu ändern
Ich nutze Bitcoin Core Version (oder equiv.)
< 0.21.1
0.21.1
Ich update auf Bitcoin Core 0.21.1


Meine Wallet benutzt eine eigene Bitcoin Full Node
Nein
Ja
Ich nutze die Zitadelle um das zu ändern
Ich nutze Bitcoin Core Version (oder equiv.)
< 0.21.1
0.21.1
Ich update auf Bitcoin Core 0.21.1
Ich stelle sicher dass meine Freunde Core 0.21.1 nutzen

Taproot Basics

Design Goal: Widerstandsfaehigkeit

Design Goal: Widerstandsfaehigkeit

Satoshi:

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.

Design Goal: Widerstandsfaehigkeit

Satoshi:

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.

- Widerstandsfaehigkeit gegenüber Einflussnahme einzelner Parteien

Design Goal: Widerstandsfaehigkeit

Satoshi:

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.

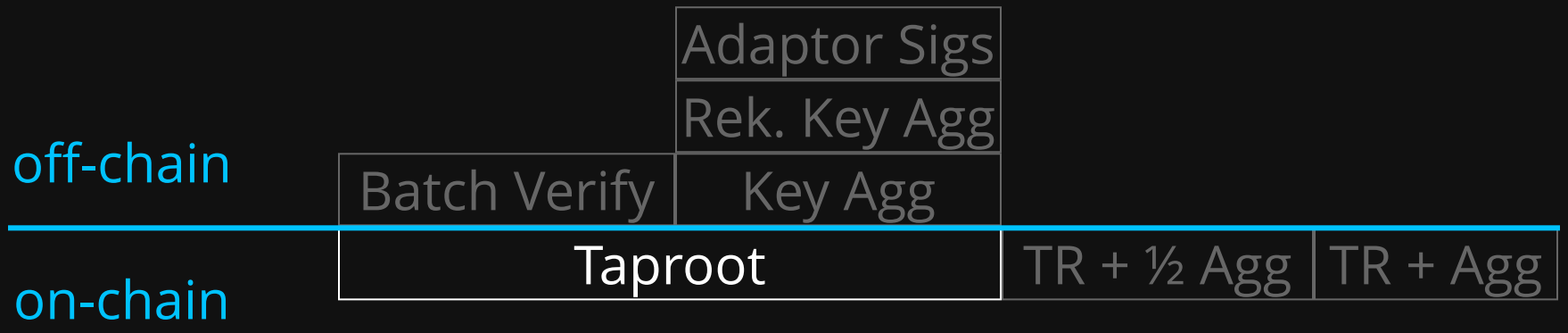
- Widerstandsfaehigkeit gegenüber Einflussnahme einzelner Parteien
- Entscheidender Faktor: Benötigte Ressourcen für Full Node

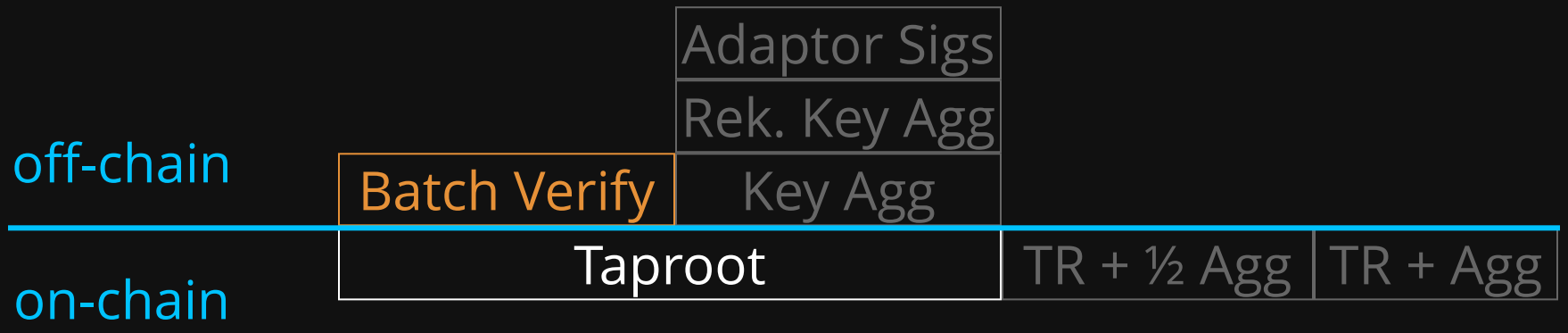
Schnorr Sigs in Taproot

Autorisierung von Transaktion:

In Blockchain: ~~ECDSA~~ Schnorr Signatur von Alices öffentlichem Schlüssel (Key)

Der Schnorr Zoo


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Schnorr Batch Verifizierung

Schnorr Batch Verifizierung

- **Ohne:** Full Node muss jede Signatur in Blockchain einzeln verifizieren

Schnorr Batch Verifizierung

- **Ohne:** Full Node muss jede Signatur in Blockchain einzeln verifizieren
- **Mit:** Full Node verfiziert einen Batch von Signaturen auf einmal

Schnorr Batch Verifizierung

- **Ohne:** Full Node muss jede Signatur in Blockchain einzeln verifizieren
- **Mit:** Full Node verfiziert einen Batch von Signaturen auf einmal
- Beispiel Batch = 10.000: Verifizierung doppelt so schnell

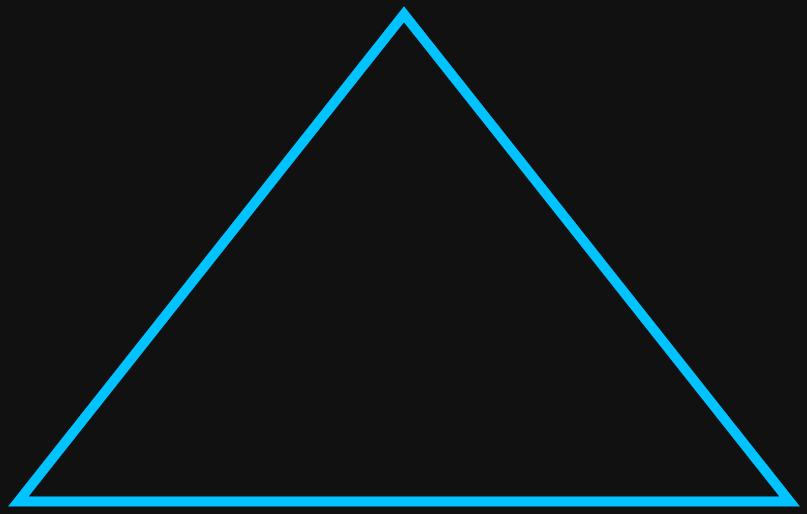
Schnorr Batch Verifizierung

- **Ohne:** Full Node muss jede Signatur in Blockchain einzeln verifizieren
- **Mit:** Full Node verfiziert einen Batch von Signaturen auf einmal
- Beispiel Batch = 10.000: Verifizierung doppelt so schnell
- Daher Full Node Ressourcenverbrauch reduziert

Schnorr Batch Verifizierung

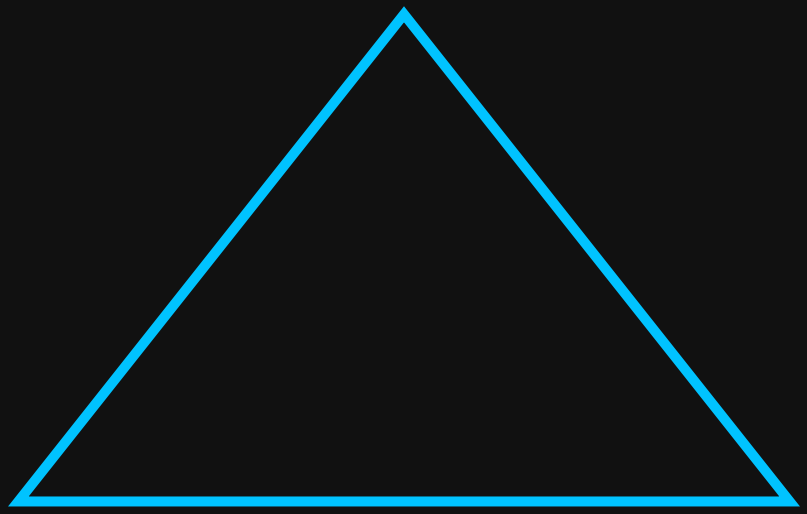
- **Ohne:** Full Node muss jede Signatur in Blockchain einzeln verifizieren
- **Mit:** Full Node verfiziert einen Batch von Signaturen auf einmal
- Beispiel Batch = 10.000: Verifizierung doppelt so schnell
- Daher Full Node Ressourcenverbrauch reduziert
- Status: Proof-of-Concept Implementierung existiert

Weitere Design Goals



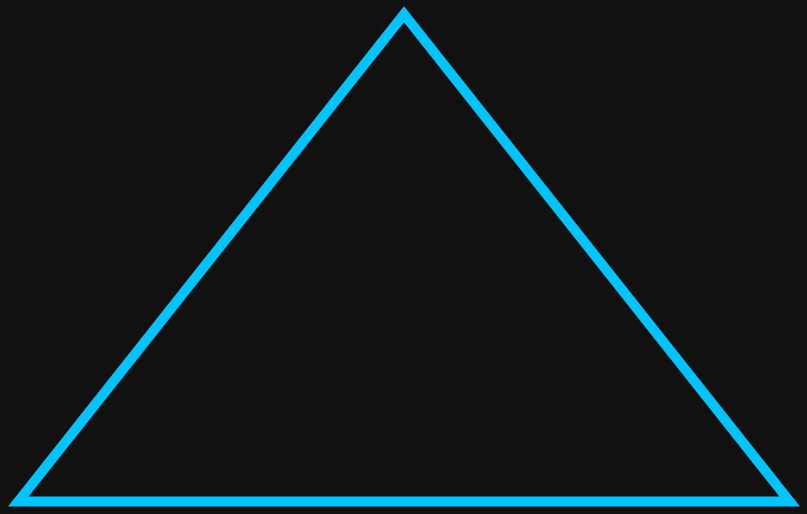
Weitere Design Goals

Überwachungsresistenz



Weitere Design Goals

Überwachungsresistenz

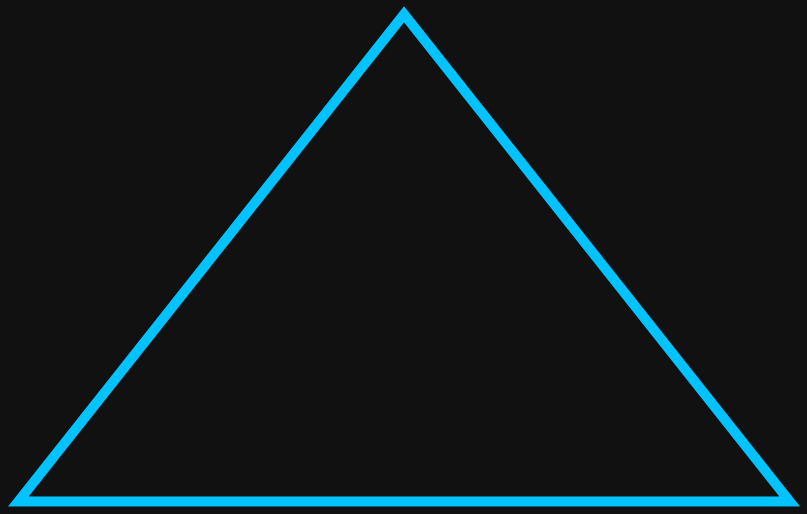


Tauglichkeit als
Zahlungsmittel

(On-chain und Layer 2)

Weitere Design Goals

Überwachungsresistenz



Tauglichkeit als Zahlungsmittel

(On-chain und Layer 2)

Sicherheit von Wallets

Ununterscheidbarkeit

Ununterscheidbarkeit

- Transaktionen haben gleiche Struktur, unabhängig davon ob sie einfache Zahlungen sind oder Teil komplexerer Protokolle.

Ununterscheidbarkeit

- Transaktionen haben gleiche Struktur, unabhängig davon ob sie einfache Zahlungen sind oder Teil komplexerer Protokolle.

Normale Zahlung?

Sidechain?

Transaktion

Multisig?

Lightning?

Ununterscheidbarkeit

- Transaktionen haben gleiche Struktur, unabhängig davon ob sie einfache Zahlungen sind oder Teil komplexerer Protokolle.

Normale Zahlung?

Sidechain?

Transaktion

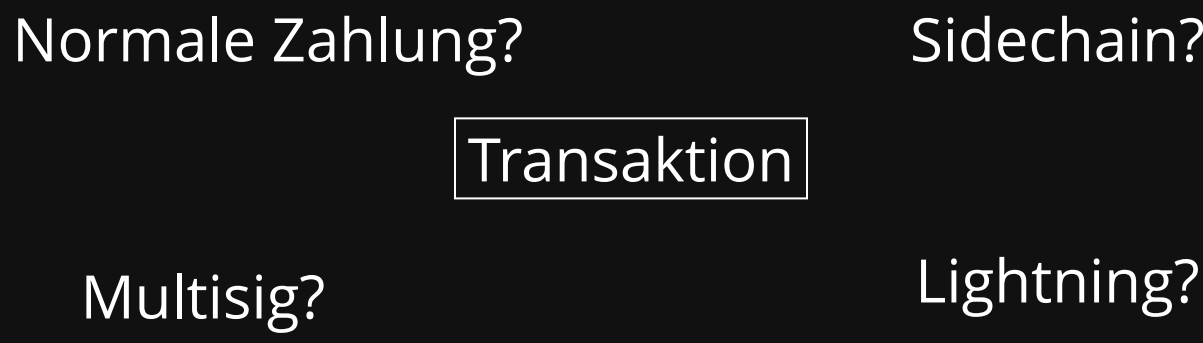
Multisig?

Lightning?

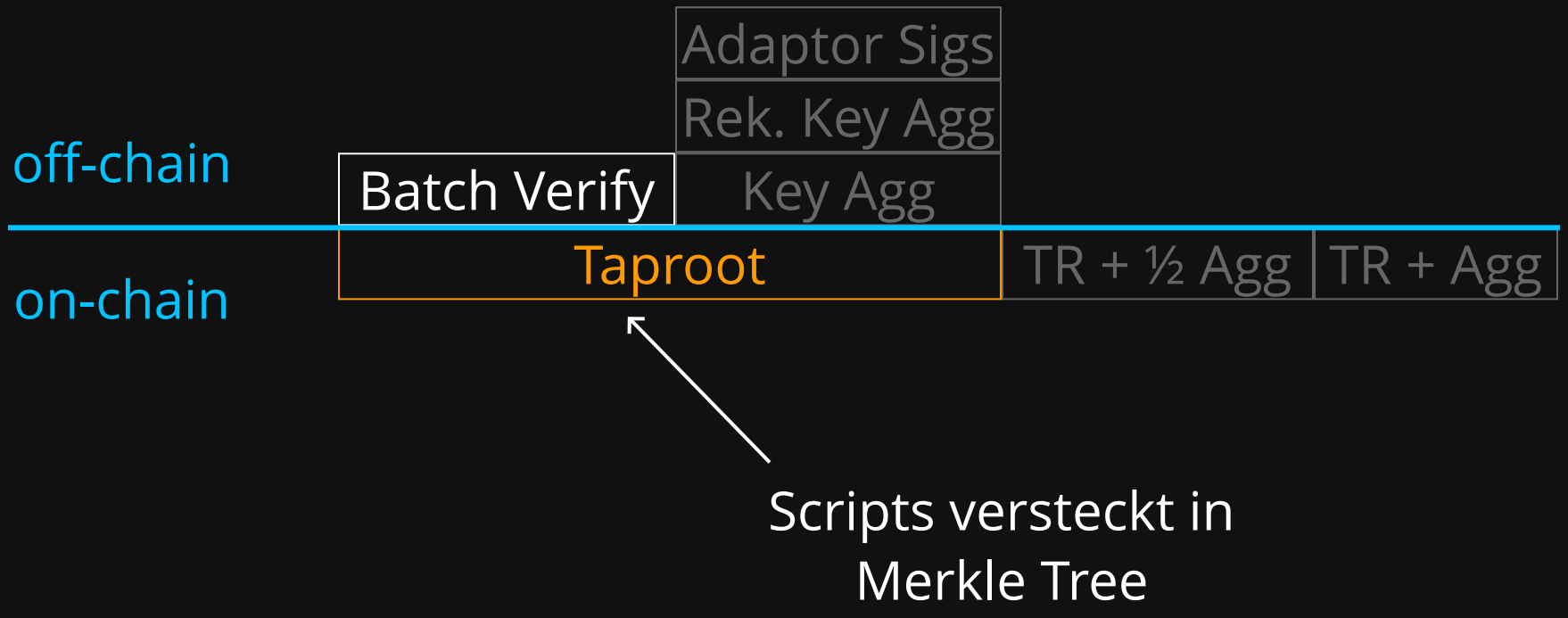
- Erschwert Ausspähen mit Blockchain

Ununterscheidbarkeit

- Transaktionen haben gleiche Struktur, unabhängig davon ob sie einfache Zahlungen sind oder Teil komplexerer Protokolle.



- Erschwert Ausspähen mit Blockchain
- Layer 2 (L2) und Multisig kostengünstiger


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
on-chain
Taproot
TR + ½ Agg
TR + Agg
Scripts versteckt in
Merkle Tree


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Key Aggregation

Key Aggregation

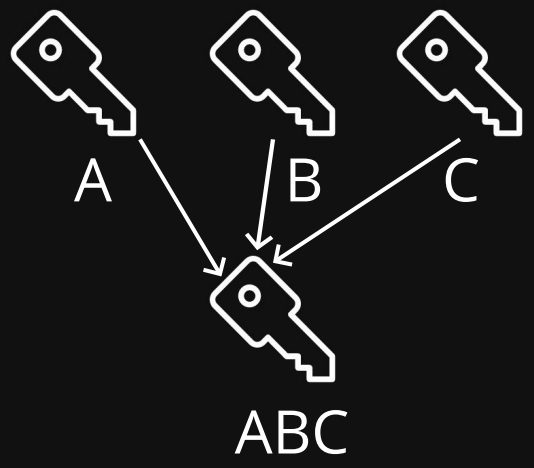
- Beispiel: Alice, Bob und Charlie haben ein 2-aus-3 Multisig Wallet.

Key Aggregation

- Beispiel: Alice, Bob und Charlie haben ein 2-aus-3 Multisig Wallet.
- **Ohne:** Alle drei Keys und zwei Signaturen müssen in die Chain geschrieben werden

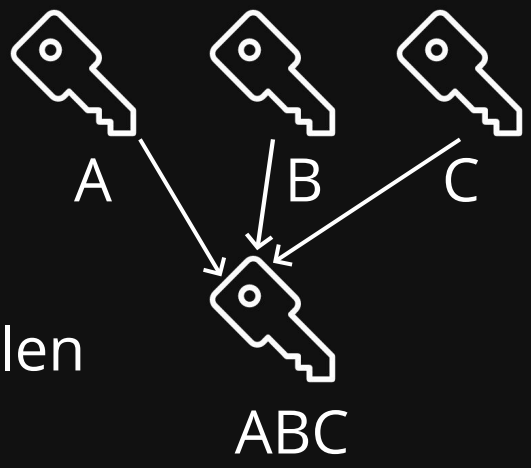
Key Aggregation

- Beispiel: Alice, Bob und Charlie haben ein 2-aus-3 Multisig Wallet.
- **Ohne:** Alle drei Keys und zwei Signaturen müssen in die Chain geschrieben werden
- **Mit:**
  - Gemeinsamer Key
  - Gemeinsam erstellte Signatur



Key Aggregation

- Beispiel: Alice, Bob und Charlie haben ein 2-aus-3 Multisig Wallet.
- **Ohne:** Alle drei Keys und zwei Signaturen müssen in die Chain geschrieben werden
- **Mit:**
  - Gemeinsamer Key
  - Gemeinsam erstellte Signatur
- Daher ununterscheidbar zu normalen Transaktionen



MuSig vs MuSig2 vs FROST

MuSig vs MuSig2 vs FROST

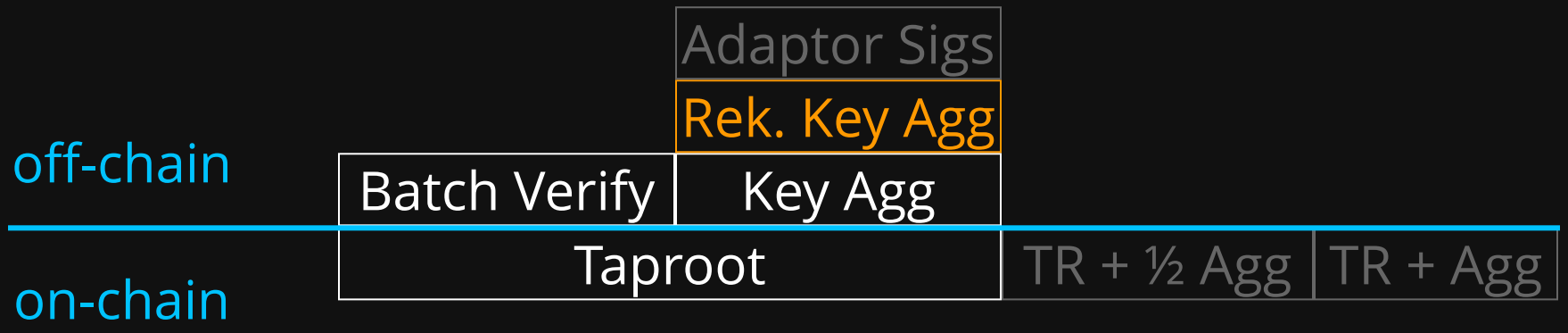
- **MuSig**: n-aus-n Multisig
  - Status: Ersetzt durch MuSig2

MuSig vs MuSig2 vs FROST

- **MuSig**: n-aus-n Multisig
  - Status: Ersetzt durch MuSig2
- **MuSig2**: n-aus-n Multisig
  - benötigt weniger Kommunikation
    - vor allem wichtig für Lightning Routing
  - Status: Impl. & Spezifizierung im Gange

MuSig vs MuSig2 vs FROST

- **MuSig**: n-aus-n Multisig
  - Status: Ersetzt durch MuSig2
- **MuSig2**: n-aus-n Multisig
  - benötigt weniger Kommunikation
    - vor allem wichtig für Lightning Routing
  - Status: Impl. & Spezifizierung im Gange
- **FROST**: m-aus-n multisig
  - Beispiel: reduziert Kosten einer 11-aus-15 Multisig Federation um 75%
  - Status: Impl. eines Prototyps im Gange


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Rekursive Key Agg

Rekursive Key Agg

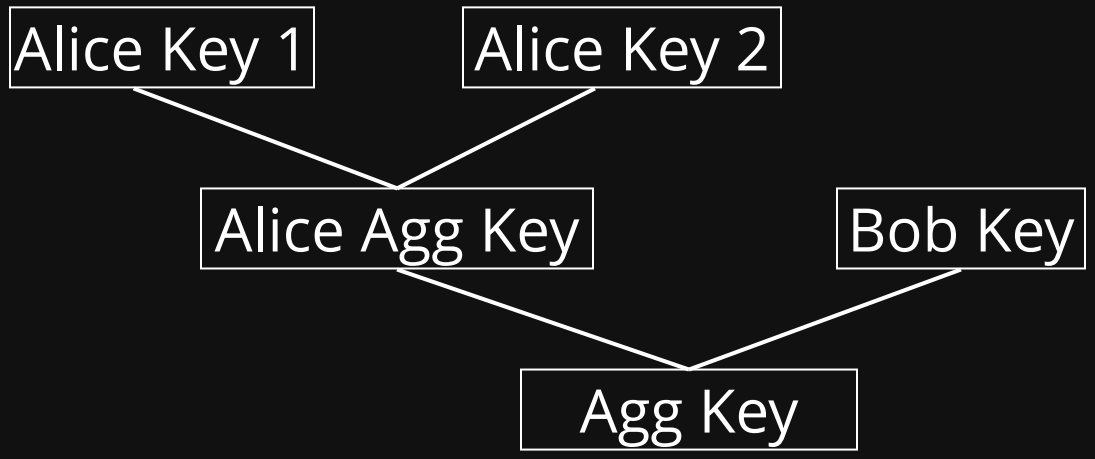
- Beispiel: Alice verteilt Schlüssel ihres Lightning Channels mit Bob auf mehrere Geräte

Rekursive Key Agg

- Beispiel: Alice verteilt Schlüssel ihres Lightning Channels mit Bob auf mehrere Geräte
- **Ohne:** Änderung des Lightning Protokolls, Bob erkennt Alice's Setup

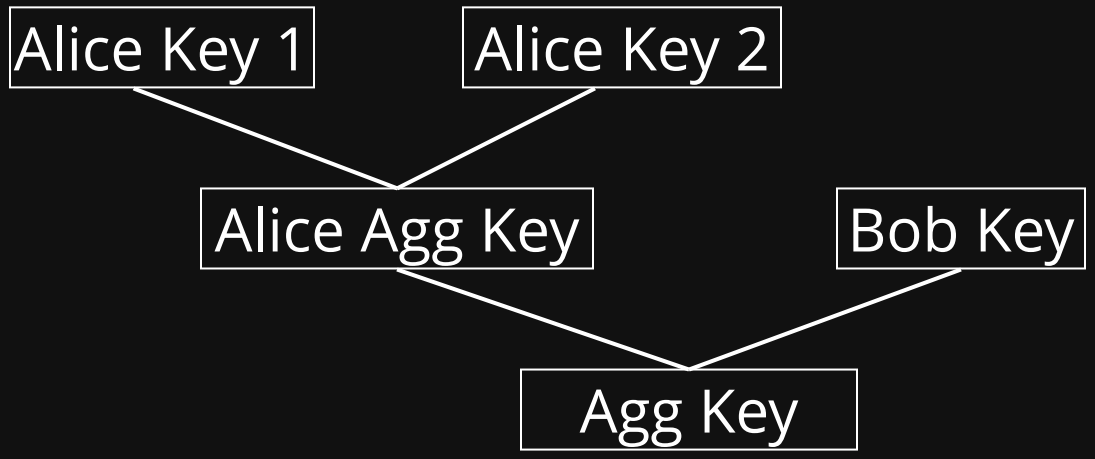
Rekursive Key Agg

- Beispiel: Alice verteilt Schlüssel ihres Lightning Channels mit Bob auf mehrere Geräte
- **Ohne:** Änderung des Lightning Protokolls, Bob erkennt Alice's Setup
- **Mit:**

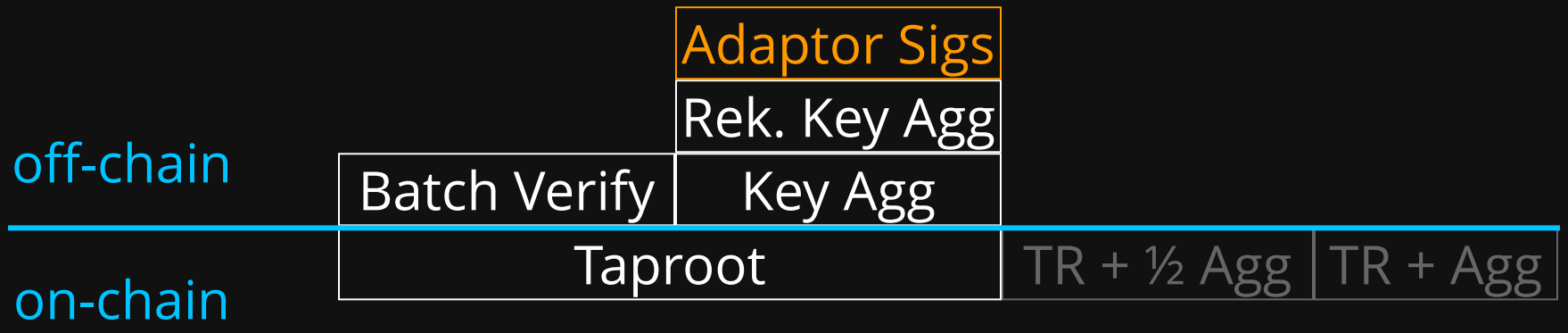


Rekursive Key Agg

- Beispiel: Alice verteilt Schlüssel ihres Lightning Channels mit Bob auf mehrere Geräte
- **Ohne:** Änderung des Lightning Protokolls, Bob erkennt Alice's Setup
- **Mit:**



- Status: Forschungsgegenstand


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Adaptor Signatures

Adaptor Signatures

- Beispiel: Atomic Swaps wie Lightning Payment, DLCs, Cross-Chain Swaps

Adaptor Signatures

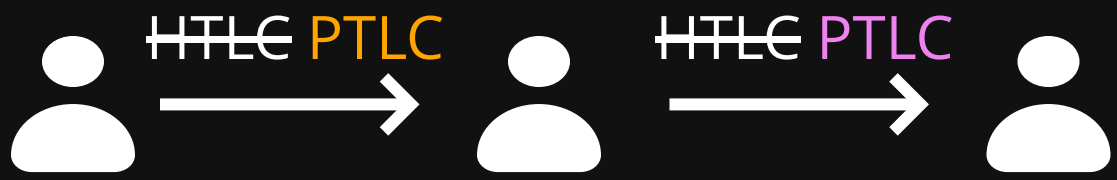
- Beispiel: Atomic Swaps wie Lightning Payment, DLCs, Cross-Chain Swaps
- **Ohne:** Benötigt on-chain Hash

Adaptor Signatures

- Beispiel: Atomic Swaps wie Lightning Payment, DLCs, Cross-Chain Swaps
- **Ohne:** Benötigt on-chain Hash
- **Mit:** Stattdessen off-chain Adaptor Signature

Adaptor Signatures

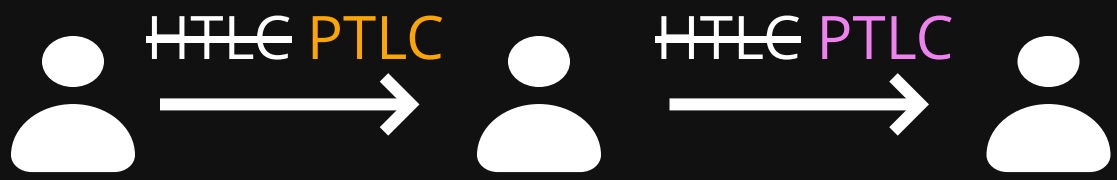
- Beispiel: Atomic Swaps wie Lightning Payment, DLCs, Cross-Chain Swaps
- **Ohne:** Benötigt on-chain Hash
- **Mit:** Stattdessen off-chain Adaptor Signature

Lightning: 

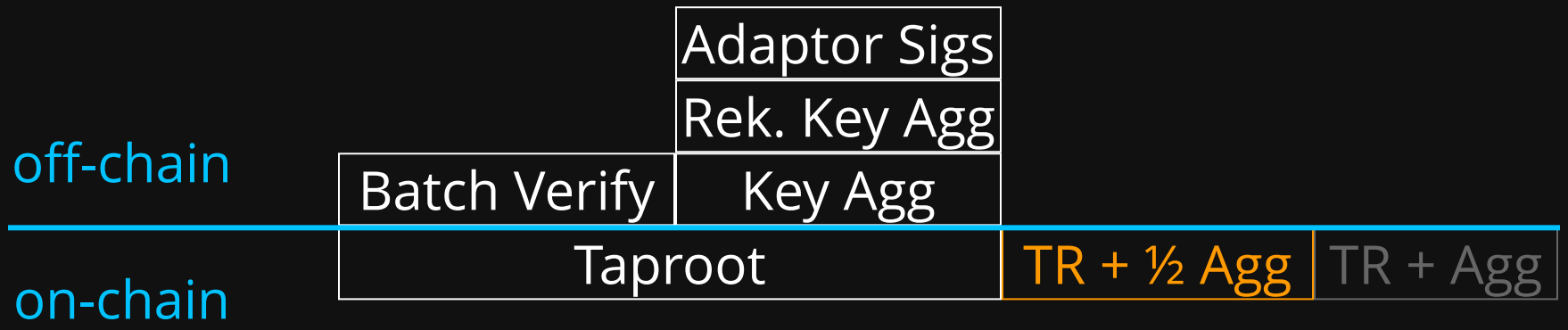
- HTLC: Hash Timelocked Contract, sichtbar on-chain, auf der Route dasselbe
- PTLC: Point Timelocked Contract

Adaptor Signatures

- Beispiel: Atomic Swaps wie Lightning Payment, DLCs, Cross-Chain Swaps
- **Ohne:** Benötigt on-chain Hash
- **Mit:** Stattdessen off-chain Adaptor Signature

Lightning: 

- HTLC: Hash Timelocked Contract, sichtbar on-chain, auf der Route dasselbe
- PTLC: Point Timelocked Contract
- Status: Spezifizierung (des Primitivs) im Gange


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Schnorr Half Aggregation

Schnorr Half Aggregation

- **Ohne:** Blöcke enthalten (mind.) eine Signatur pro ausgegebenen Coin

Schnorr Half Aggregation

- **Ohne:** Blöcke enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Blöcke enthalten eine "halb-"aggregierte Signatur, die etwa halb so groß ist wie die Summe der einzelnen Signaturen
  - $\mathsf{Aggregate}(\mathsf{sig}_1, \ldots, \mathsf{sig}_n) \to \mathsf{sig}$
  - Nicht-interaktiv

Schnorr Half Aggregation

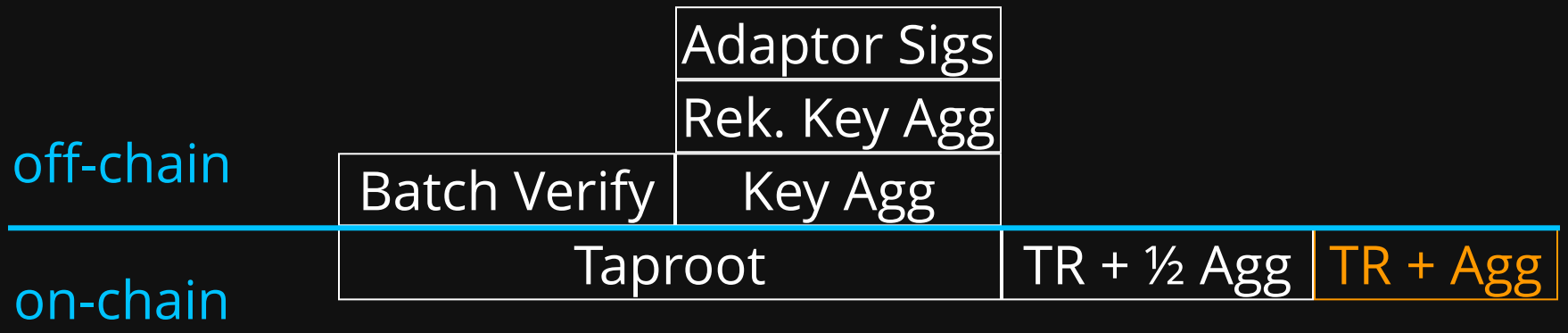
- **Ohne:** Blöcke enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Blöcke enthalten eine "halb-"aggregierte Signatur, die etwa halb so groß ist wie die Summe der einzelnen Signaturen
  - $\mathsf{Aggregate}(\mathsf{sig}_1, \ldots, \mathsf{sig}_n) \rightarrow \mathsf{sig}$
  - Nicht-interaktiv
- Beispiel: 10 halb-aggregierte Signaturen benötigen Platz von 6 normalen Signaturen

Schnorr Half Aggregation

- **Ohne:** Blöcke enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Blöcke enthalten eine "halb-"aggregierte Signatur, die etwa halb so groß ist wie die Summe der einzelnen Signaturen
  - $\mathsf{Aggregate}(\mathsf{sig}_1, \ldots, \mathsf{sig}_n) \to \mathsf{sig}$
  - Nicht-interaktiv
- Beispiel: 10 halb-aggregierte Signaturen benötigen Platz von 6 normalen Signaturen
- Daher mehr Transaktion pro Block möglich

Schnorr Half Aggregation

- **Ohne:** Blöcke enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Blöcke enthalten eine "halb-"aggregierte Signatur, die etwa halb so groß ist wie die Summe der einzelnen Signaturen
  - $\mathsf{Aggregate}(\mathsf{sig}_1, \ldots, \mathsf{sig}_n) \rightarrow \mathsf{sig}$
  - Nicht-interaktiv
- Beispiel: 10 halb-aggregierte Signaturen benötigen Platz von 6 normalen Signaturen
- Daher mehr Transaktion pro Block möglich
- Status: Forschungsgegenstand, würde weiteren Softfork benötigen


Adaptor Sigs
Rek. Key Agg
off-chain
Batch Verify
Key Agg
Taproot
TR + ½ Agg
TR + Agg
on-chain

Schnorr Full Aggregation

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Transaktionen enthalten genau eine, aggregierte Signatur

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Transaktionen enthalten genau eine, aggregierte Signatur
- Größe gleich normaler Schnorr Signatur

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Transaktionen enthalten genau eine, aggregierte Signatur
- Größe gleich normaler Schnorr Signatur
- Signieren ist interaktiv

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Transaktionen enthalten genau eine, aggregierte Signatur
- Größe gleich normaler Schnorr Signatur
- Signieren ist interaktiv
- Kleinere Transaktionen, schafft Anreiz für CoinJoin

Schnorr Full Aggregation

- **Ohne:** Transaktionen enthalten (mind.) eine Signatur pro ausgegebenen Coin
- **Mit:** Transaktionen enthalten genau eine, aggregierte Signatur
- Größe gleich normaler Schnorr Signatur
- Signieren ist interaktiv
- Kleinere Transaktionen, schafft Anreiz für CoinJoin
- Status: Forschungsgegenstand, würde weiteren Softfork benötigen

Zusammenfassung

Zusammenfassung

- nickler.ninja/slides/

Zusammenfassung

- nickler.ninja/slides/
- Full Nodes updaten!

Zusammenfassung

- nickler.ninja/slides/
- Full Nodes updaten!
- Widerstandsfähigkeit ist A und O
  - Überwachungsresistenz
  - Tauglichkeit als Zahlungsmittel
  - Sicherheit von Wallets

Zusammenfassung

- nickler.ninja/slides/
- Full Nodes updaten!
- Widerstandsfähigkeit ist A und O
  - Überwachungsresistenz
  - Tauglichkeit als Zahlungsmittel
  - Sicherheit von Wallets
- Ununterscheidbarkeit

Zusammenfassung

- nickler.ninja/slides/
- Full Nodes updaten!
- Widerstandsfähigkeit ist A und O
  - Überwachungsresistenz
  - Tauglichkeit als Zahlungsmittel
  - Sicherheit von Wallets
- Ununterscheidbarkeit
- Key Aggregation: Aggregierung in einem Multisig Wallet

Zusammenfassung

- nickler.ninja/slides/
- Full Nodes updaten!
- Widerstandsfähigkeit ist A und O
  - Überwachungsresistenz
  - Tauglichkeit als Zahlungsmittel
  - Sicherheit von Wallets
- Ununterscheidbarkeit
- Key Aggregation: Aggregierung in einem Multisig Wallet
- Sig Aggregation: Aggregierung über Wallets hinweg

Zusammenfassung

Protokoll	**Anwendung**	**Nutzen**	**Status (inkl. Warten auf TR)**
Batch Verify	Schnelleres Verifizieren	Full Node Ressourcen	Prototyp Impl.
TR Merkle Tree	Versteckte Script Paths	Kleinere Txs, Überwachungsresistenz	-
MuSig	n-aus-n Multisig	Kleinere Txs, Überwachungsresistenz	Ersetzt durch MuSig2
MuSig2	"	"	Spezifizierung in Gange
FROST	t-aus-n Multisig	"	Impl. in Gange
Rek. Key Agg	Multisig aus Multisigs	L2 Tricks	Forschungsgegenstand
Adaptor Sigs	Swaps, HTLCs	nützlich fuer L2, Überwachungsresistenz	Spezifizierung in Gange
Blind Sigs	Blind Swap	Überwachungsresistenz	Anwendungen?
Half Agg	Alle Txs	Kleinere Txs	Forschungsgegenstand, benötigt Softfork
Full Agg	Alle Txs	Kleinere Txs	Forschungsgegenstand, benötigt Softfork