November 28th, 2016

CSAIL

Massachusetts Institute of Technology

Catena: Preventing Lies with bitcoin

Alin Tomescu
alinush@mit.edu
MIT CSAIL

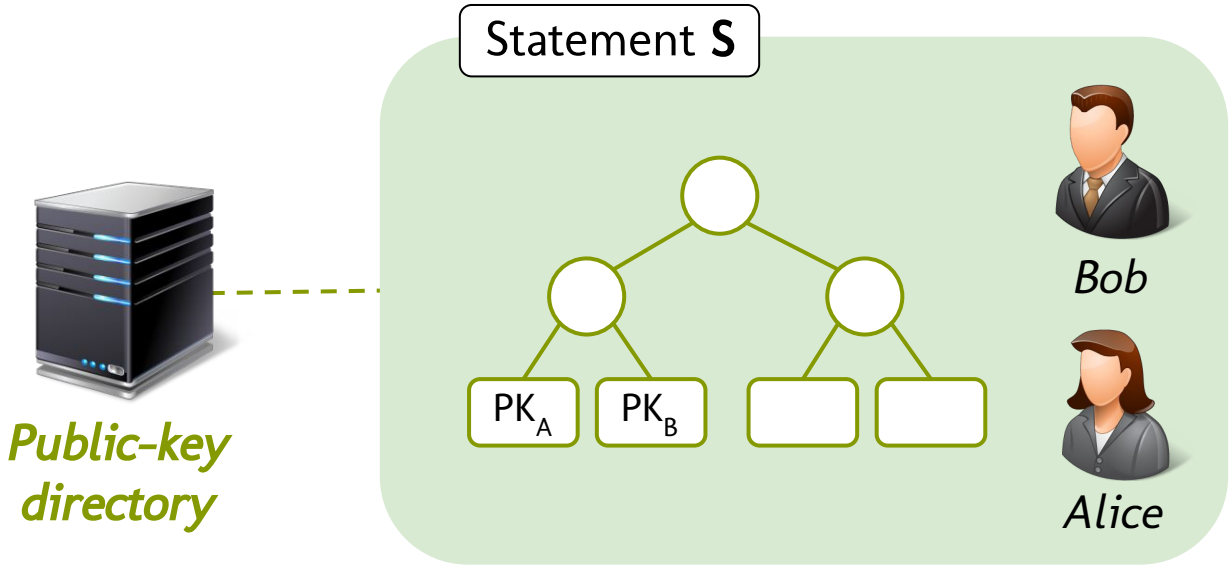
Srinivas Devadas
devadas@mit.edu
MIT CSAIL

New England Security Day (NESD), Fall '16

The problem: Equivocation

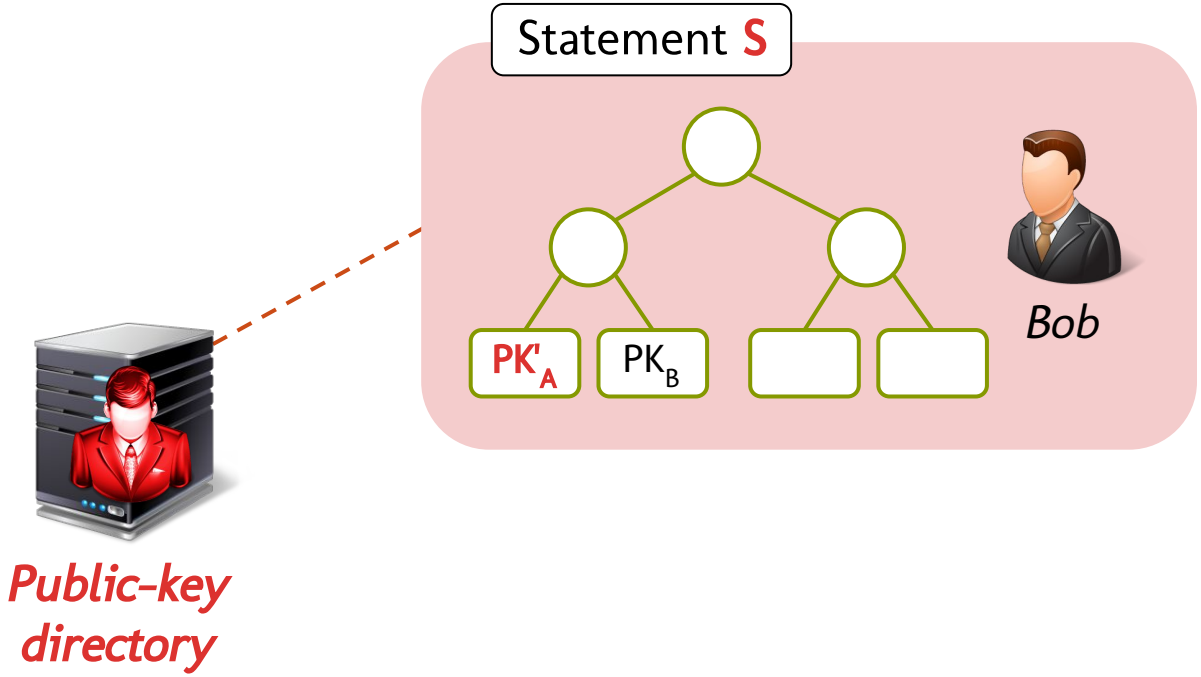
The problem: Equivocation

Good: *"Stating the same thing to all people."*



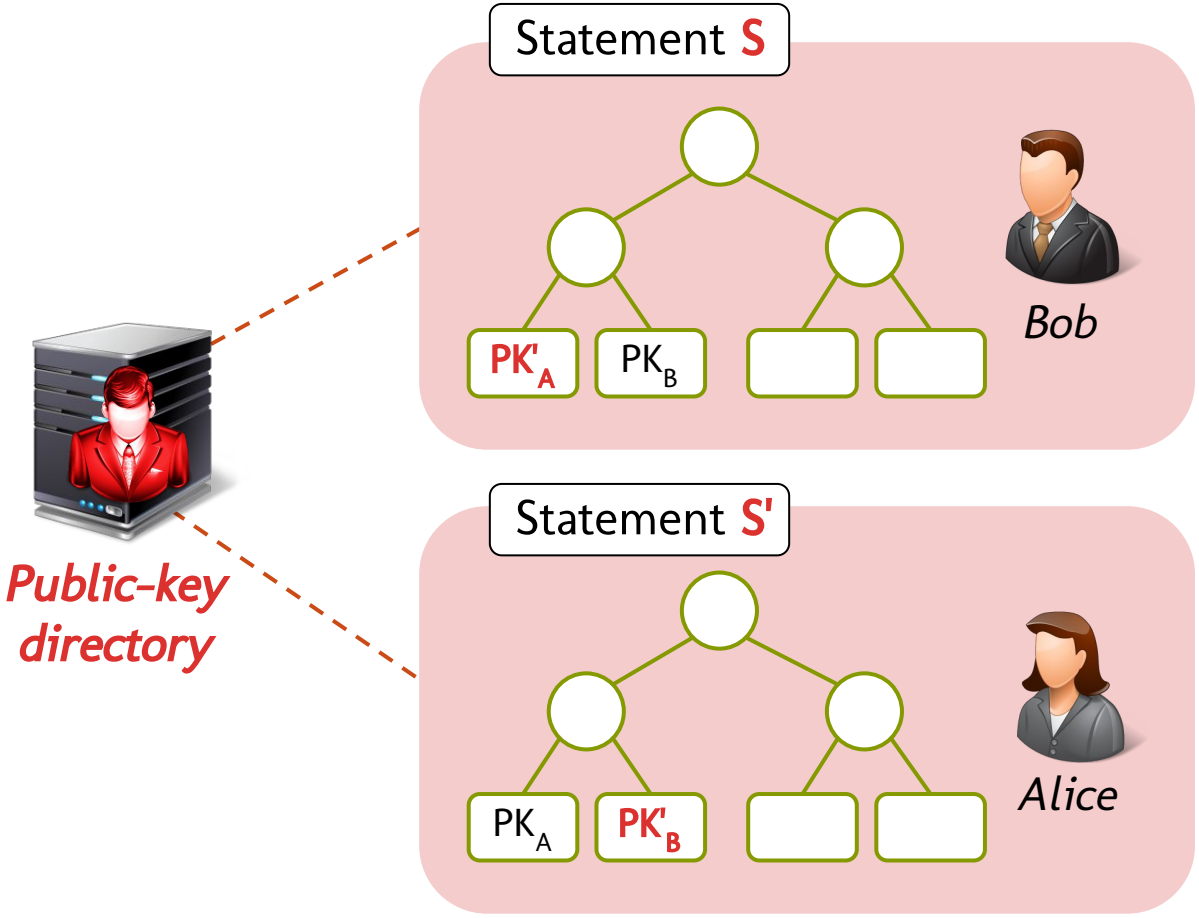
The problem: Equivocation

Bad: *"Stating different things to different people."*



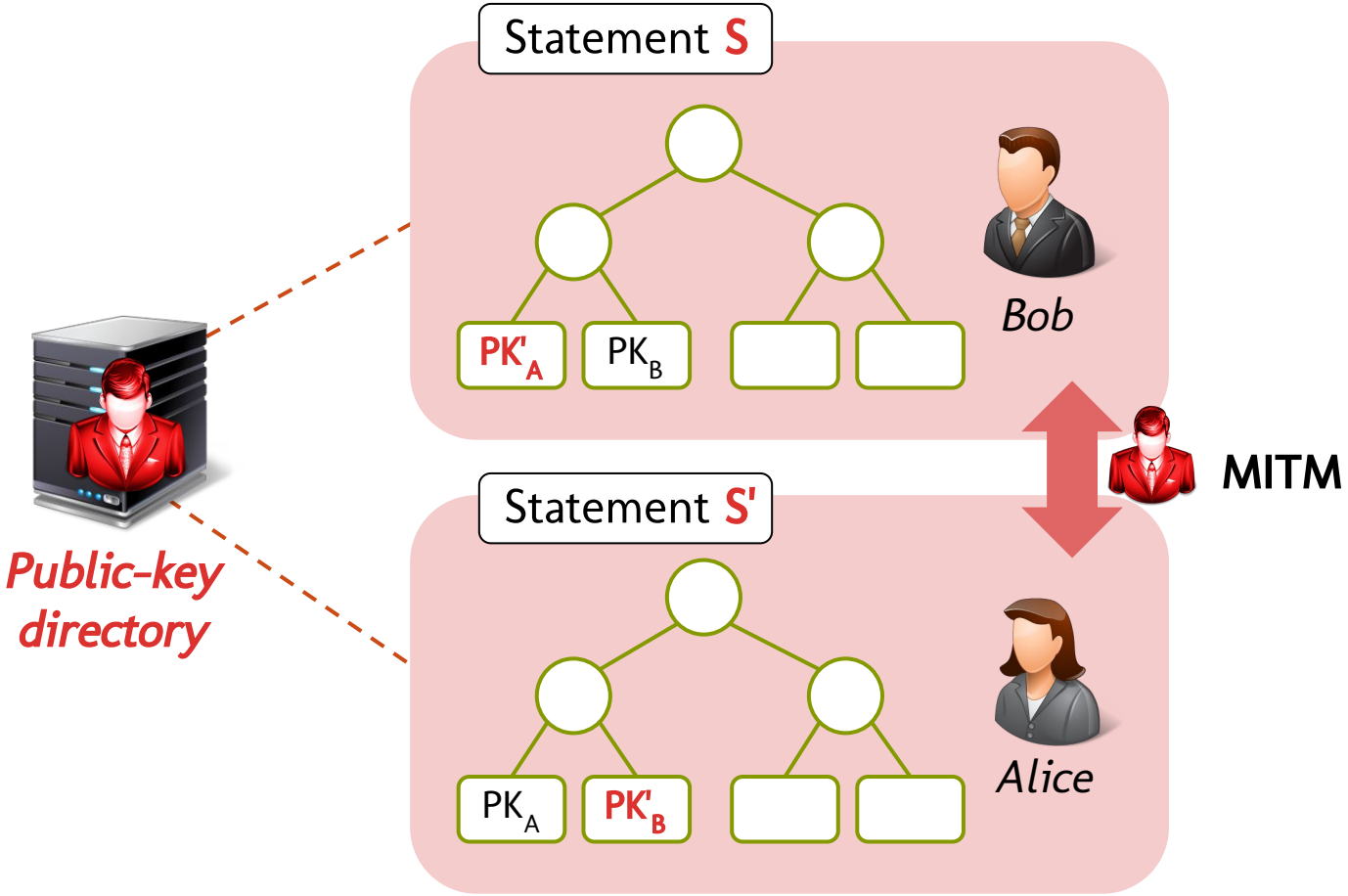
The problem: Equivocation

Bad: *"Stating different things to different people."*



The problem: Equivocation

Bad: *"Stating different things to different people."*



Why is non-equivocation important?

Why is non-equivocation important?

Public-key distribution

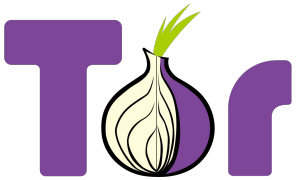
- HTTPS
- Secure messaging
- Security research often assumes a PKI

Why is non-equivocation important?

Public-key distribution

- HTTPS
- Secure messaging
- Security research often assumes a PKI

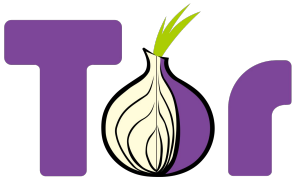
Tor Directory Servers



Why is non-equivocation important?

Public-key distribution

- HTTPS
- Secure messaging
- Security research often assumes a PKI



Tor Directory Servers

Software transparency schemes

- Apple vs. FBI

Previous work

Previous work

Can detect, but not prevent equivocation with gossip.

Previous work

Can detect, but not prevent equivocation with gossip.

blockstack

Must download 90 GB of blockchain data.

Previous work

Can detect, but not prevent equivocation with gossip.

blockstack

Must download 90 GB of blockchain data.

CoSi
(S&P '16)

Requires a large, diverse, trustworthy set of witnesses.

Previous work

Can detect, but not prevent equivocation with gossip.

blockstack

Must download 90 GB of blockchain data.

CoSi
(S&P '16)

Requires a large, diverse, trustworthy set of witnesses.

"Liar, liar, coins on fire!"
(CCS '15)

Only disincentivizes equivocation. Vulnerable to malicious outsiders.

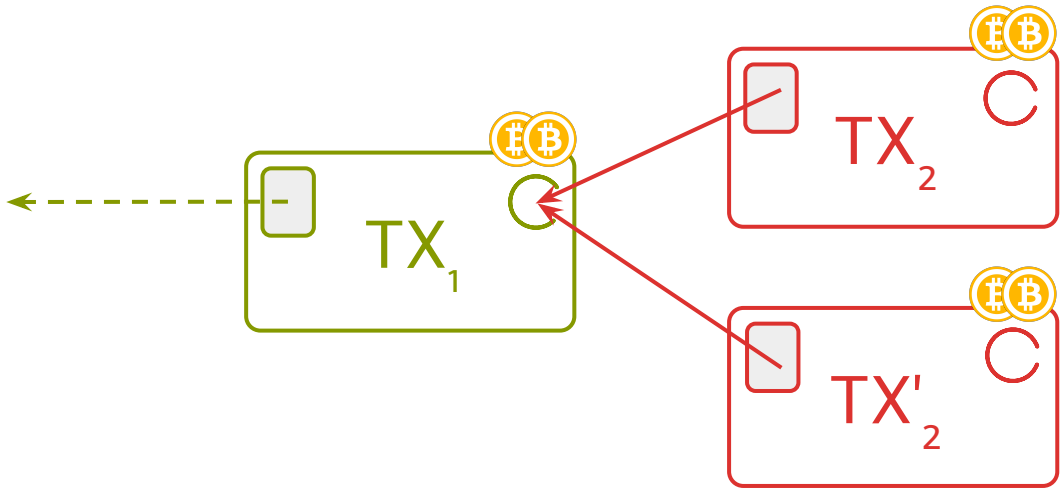
Key idea

Key idea

Efficiently use Bitcoin's mechanism that prevents double spends as a proof of non-equivocation.

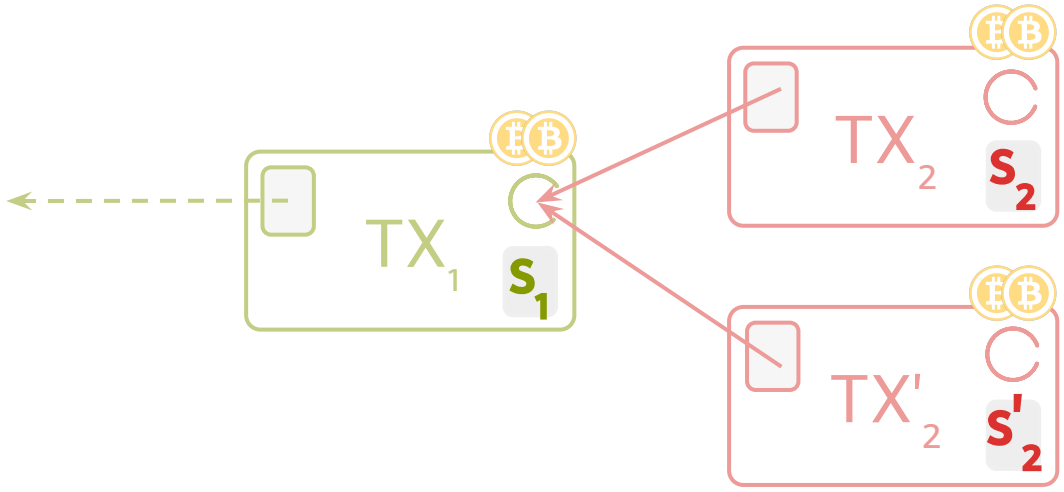
Key idea

Efficiently use Bitcoin's mechanism that prevents double spends as a proof of non-equivocation.



Key idea

Efficiently use Bitcoin's mechanism that prevents double spends as a proof of non-equivocation.



Results

- Bitcoin-based tamper-evident log

Results

- Bitcoin-based tamper-evident log
- As hard-to-fork as the Bitcoin blockchain

Results

- Bitcoin-based tamper-evident log
- As hard-to-fork as the Bitcoin blockchain
- Efficient to audit: 620 bytes / statement + 80 bytes / block

Results

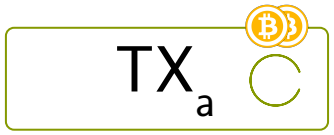
Java implementation in 3500 SLOC

- Bitcoin-based tamper-evident log
- As hard-to-fork as the Bitcoin blockchain
- Efficient to audit: 620 bytes / statement + 80 bytes / block

Bitcoin transactions

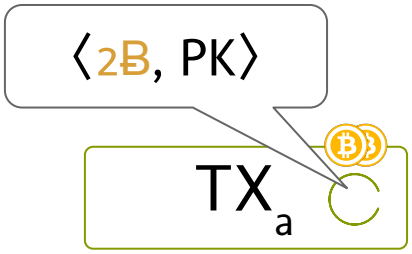
Bitcoin transactions

1. Generate coins (assigns them to a PK)



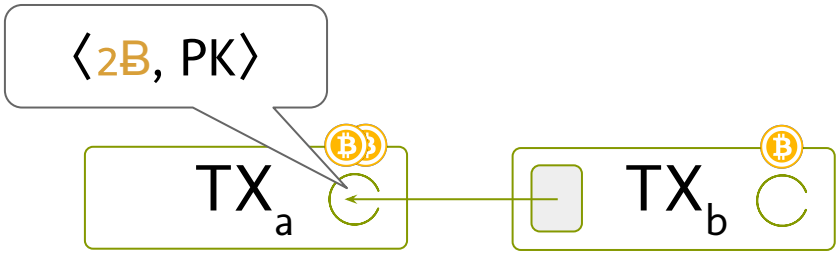
Bitcoin transactions

1. Generate coins (assigns them to a PK)



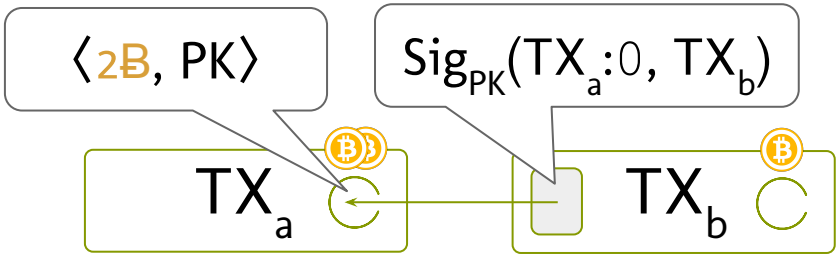
Bitcoin transactions

1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



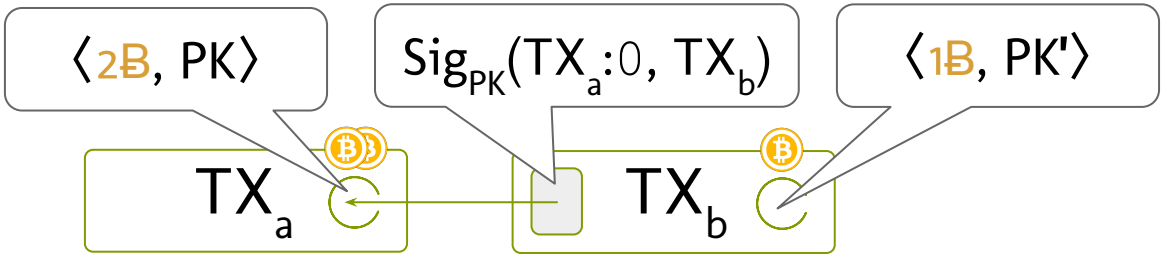
Bitcoin transactions

1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



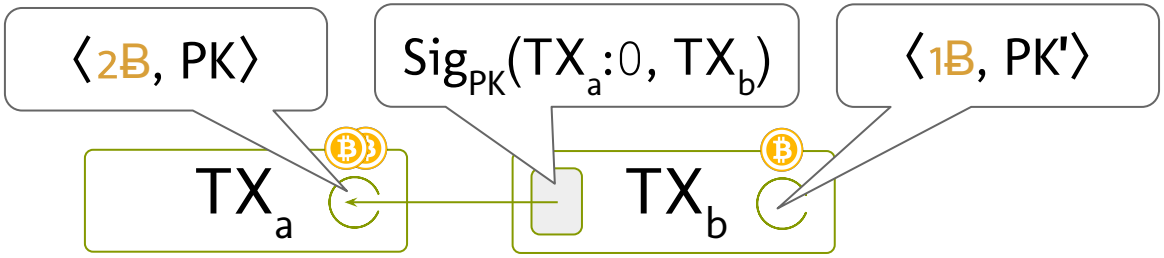
Bitcoin transactions

1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



Bitcoin transactions

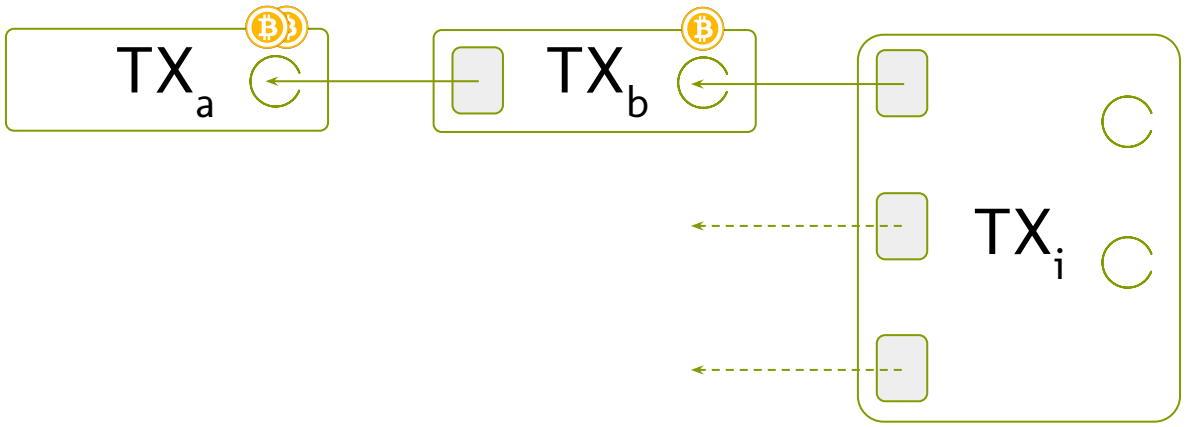
1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



1Ƀ TX fee!

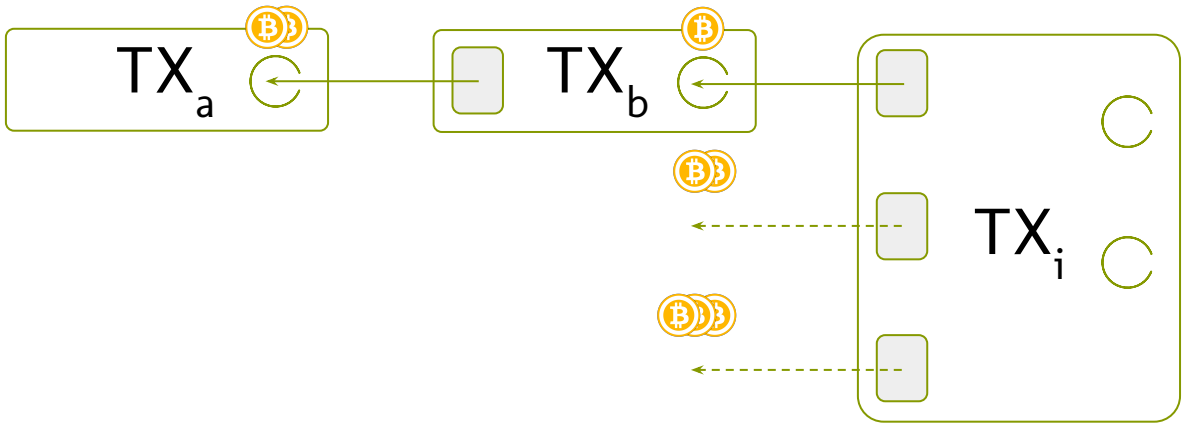
Bitcoin transactions

1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



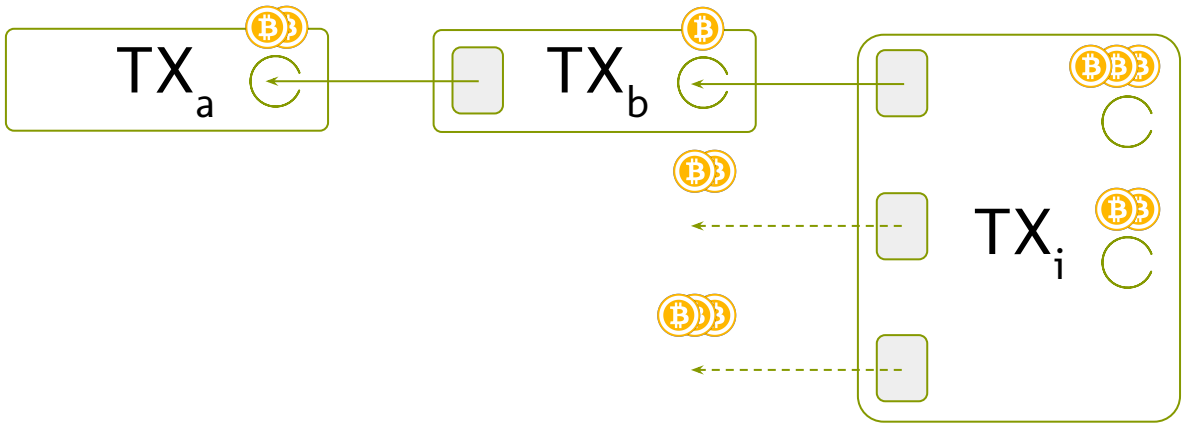
Bitcoin transactions

1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



Bitcoin transactions

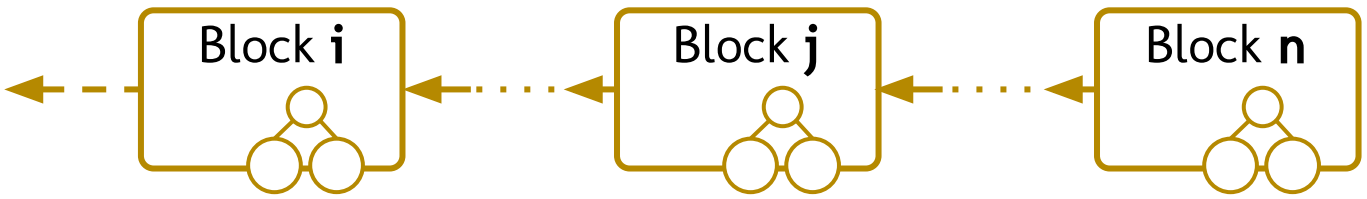
1. Generate coins (assigns them to a PK)
2. Transfer coins (reassign to a new PK via a signature under old PK)



Bitcoin blockchain

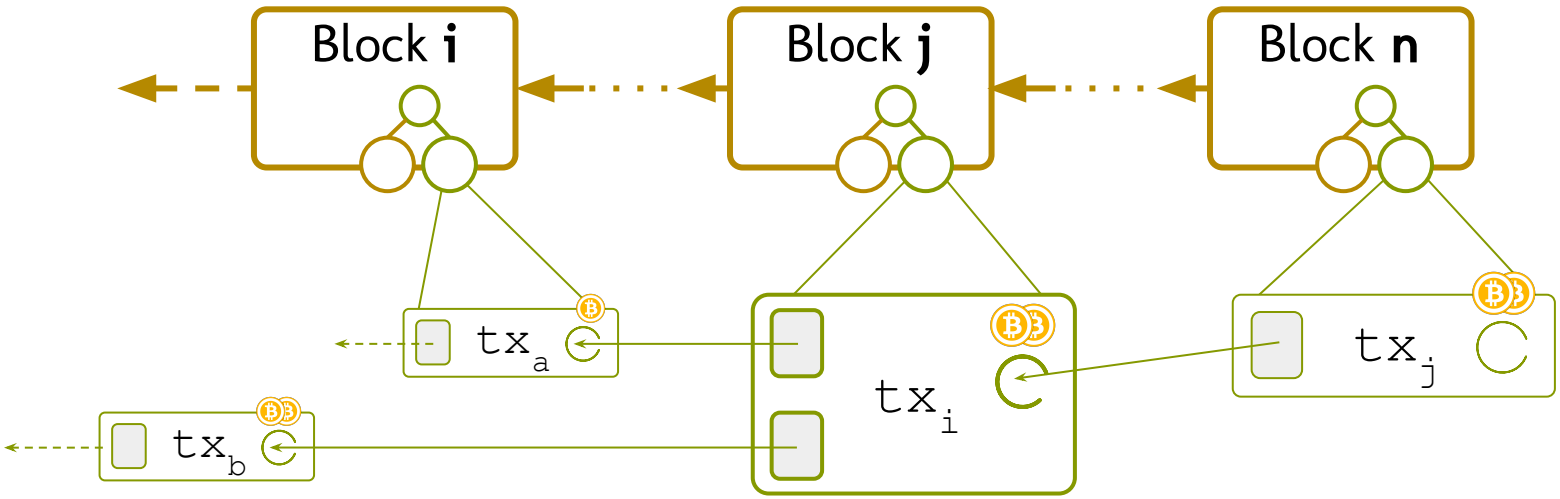
Bitcoin blockchain

1. The time-ordered log of *valid* transactions (PoW consensus)



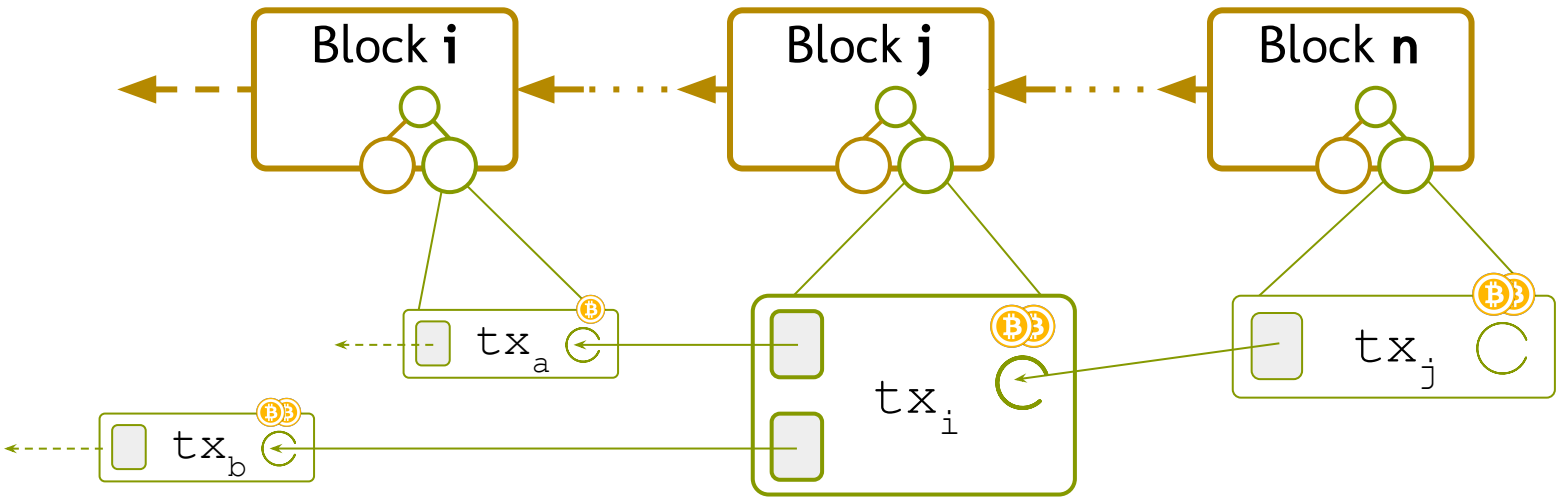
Bitcoin blockchain

1. The time-ordered log of *valid* transactions (PoW consensus)



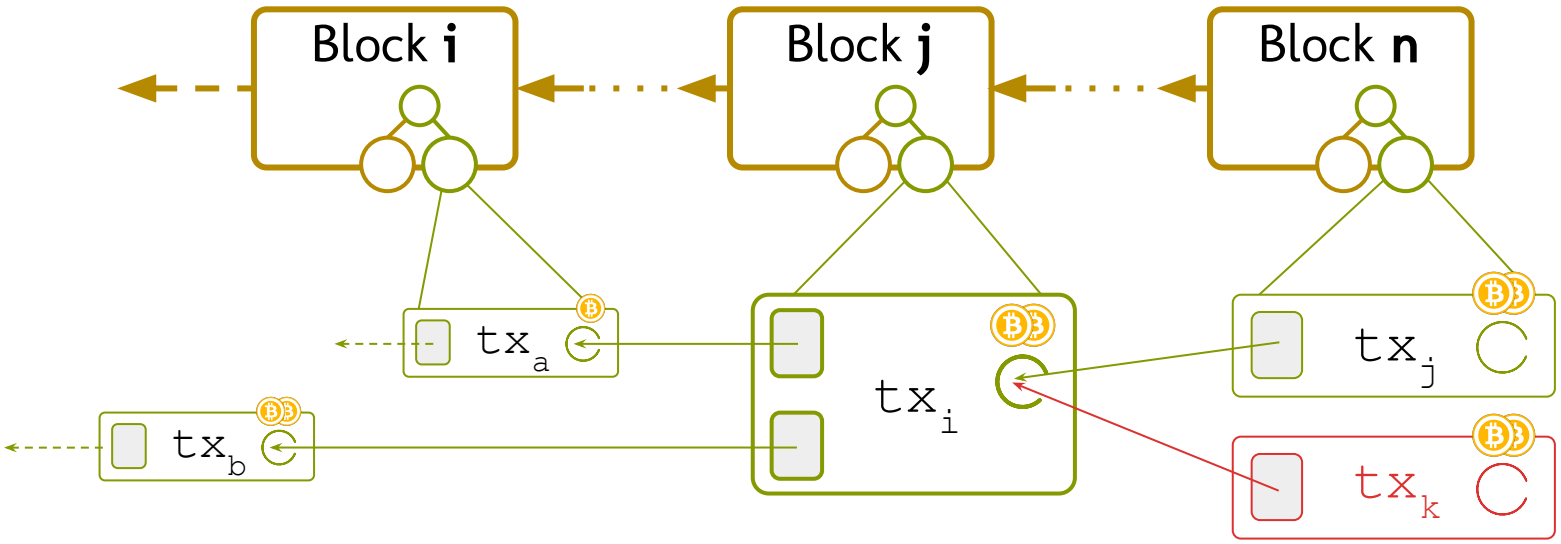
Bitcoin blockchain

1. The time-ordered log of *valid* transactions (PoW consensus)
2. No double spends: A transaction output can only be referred to by a single transaction input.



Bitcoin blockchain

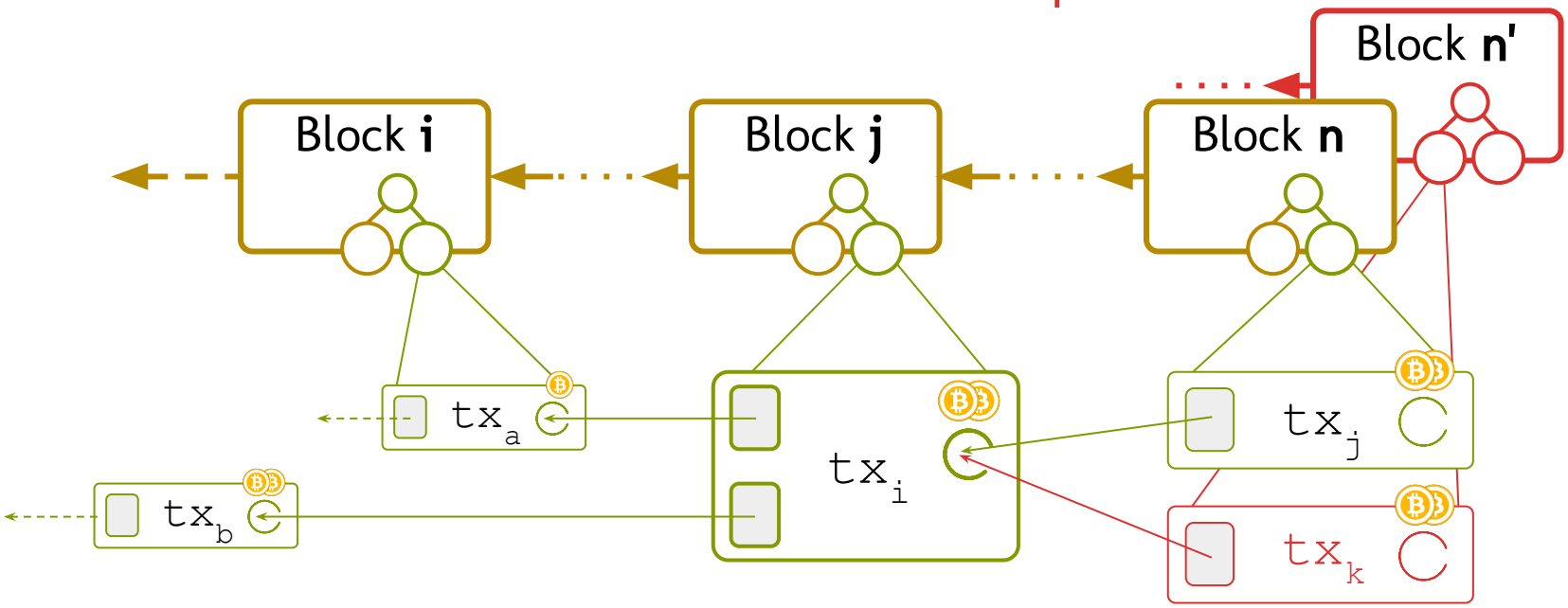
1. The time-ordered log of *valid* transactions (PoW consensus)
2. No double spends: A transaction output can only be referred to by a single transaction input.



Bitcoin blockchain

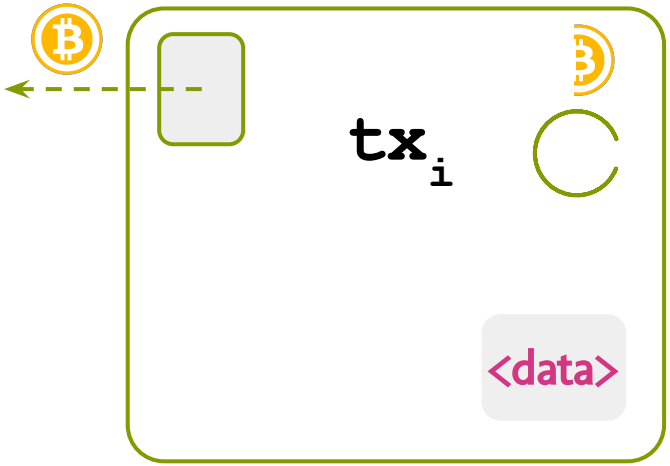
1. The time-ordered log of *valid* transactions (PoW consensus)
2. No double spends: A transaction output can only be referred to by a single transaction input.

Blockchain forks ⇔ Double-spent coins



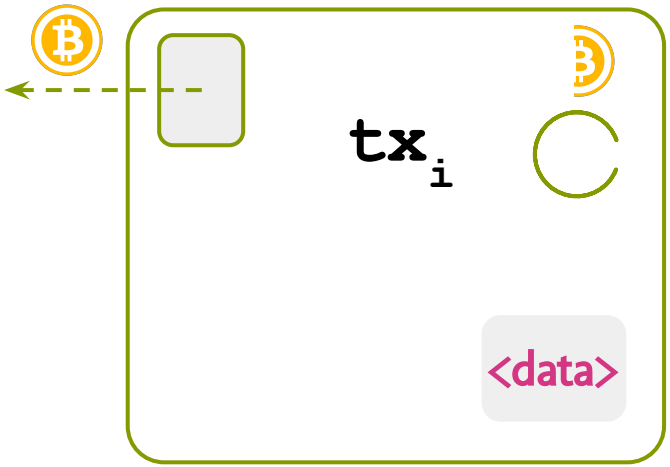
Catena transaction format

Catena transaction format



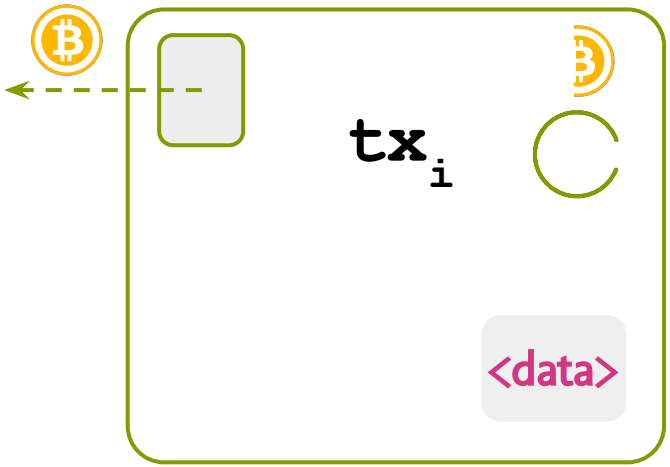
Catena transaction format

Coins from server
for paying TX fees
(digital signature)



Catena transaction format

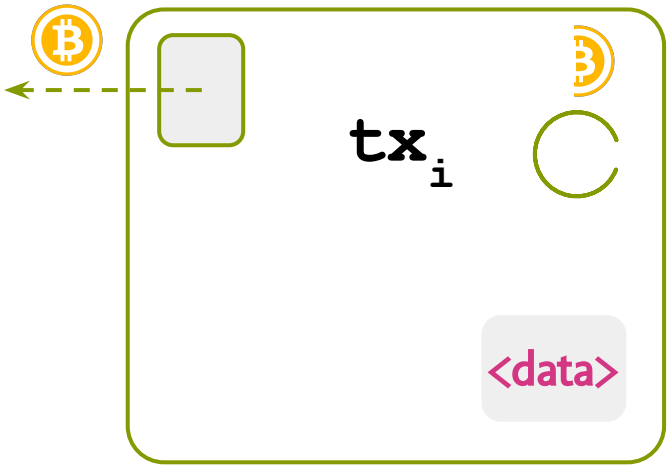
Coins from server
for paying TX fees
(digital signature)



"Change" coins back to server
(public key)

Catena transaction format

Coins from server
for paying TX fees
(digital signature)

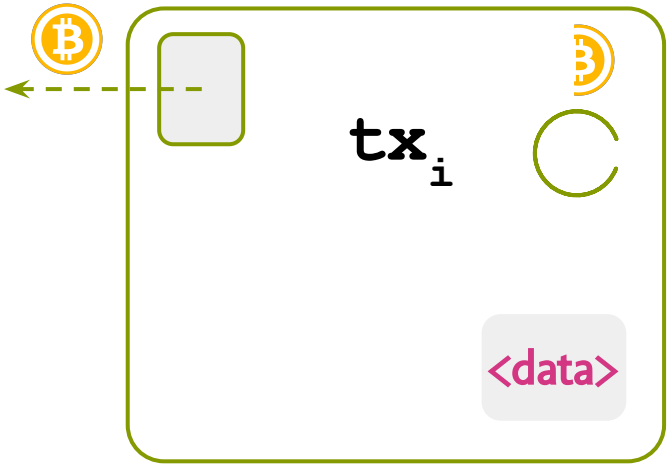


"Change" coins back to server
(public key)

Unspendable `OP_RETURN`
output with arbitrary data

Catena transaction format

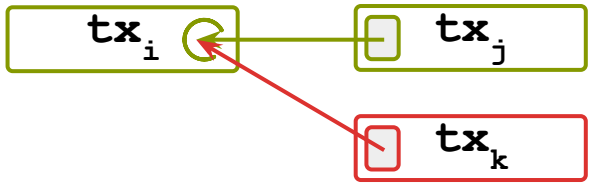
Coins from server for paying TX fees (digital signature)



"Change" coins back to server (public key)

Unspendable `OP_RETURN` output with arbitrary data

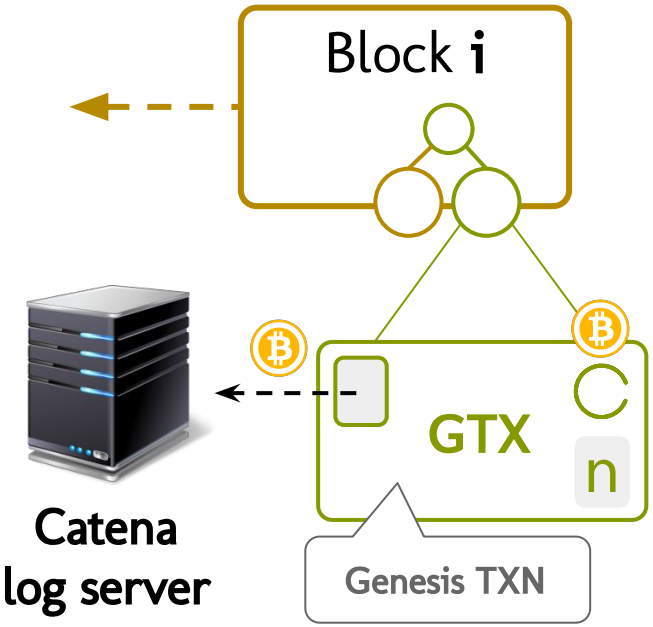
A **single spendable output** ⇒ No forks



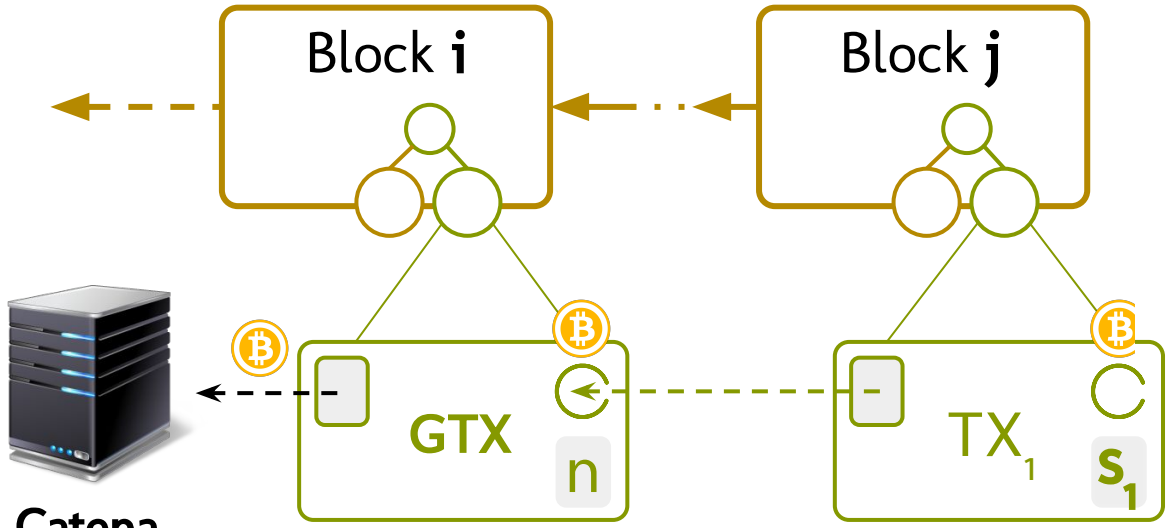
Catena design

Catena log server

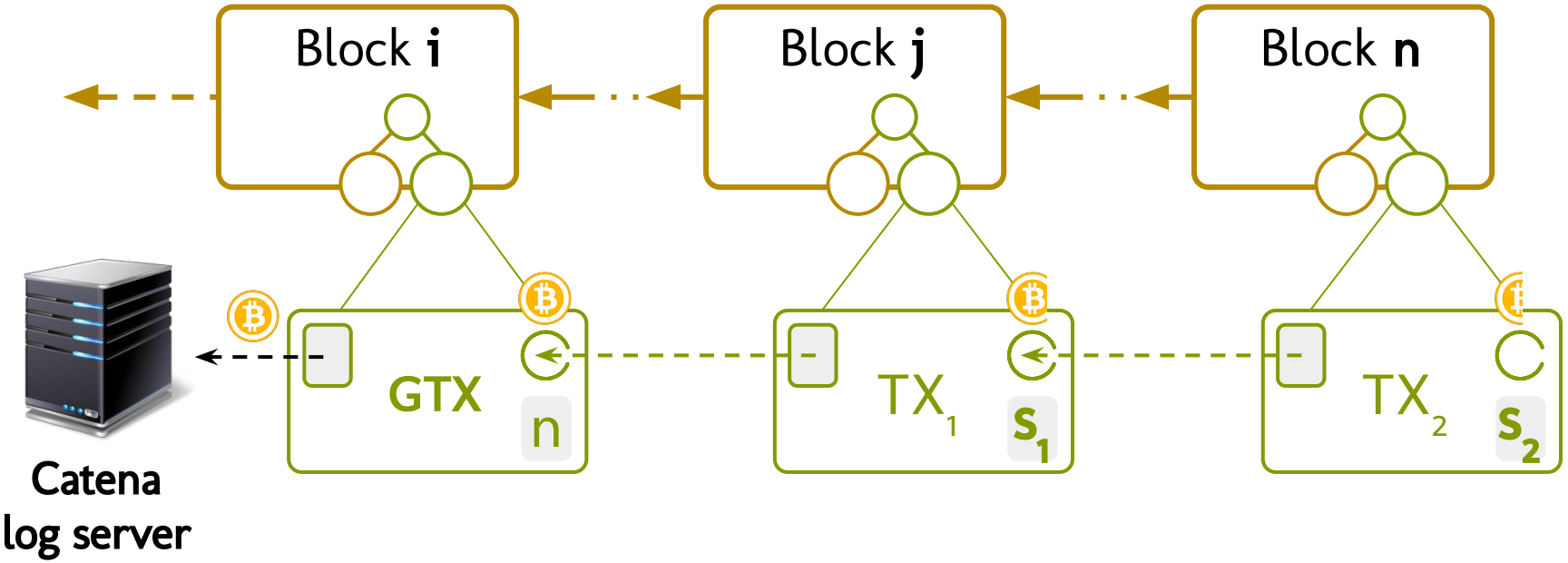
Catena design



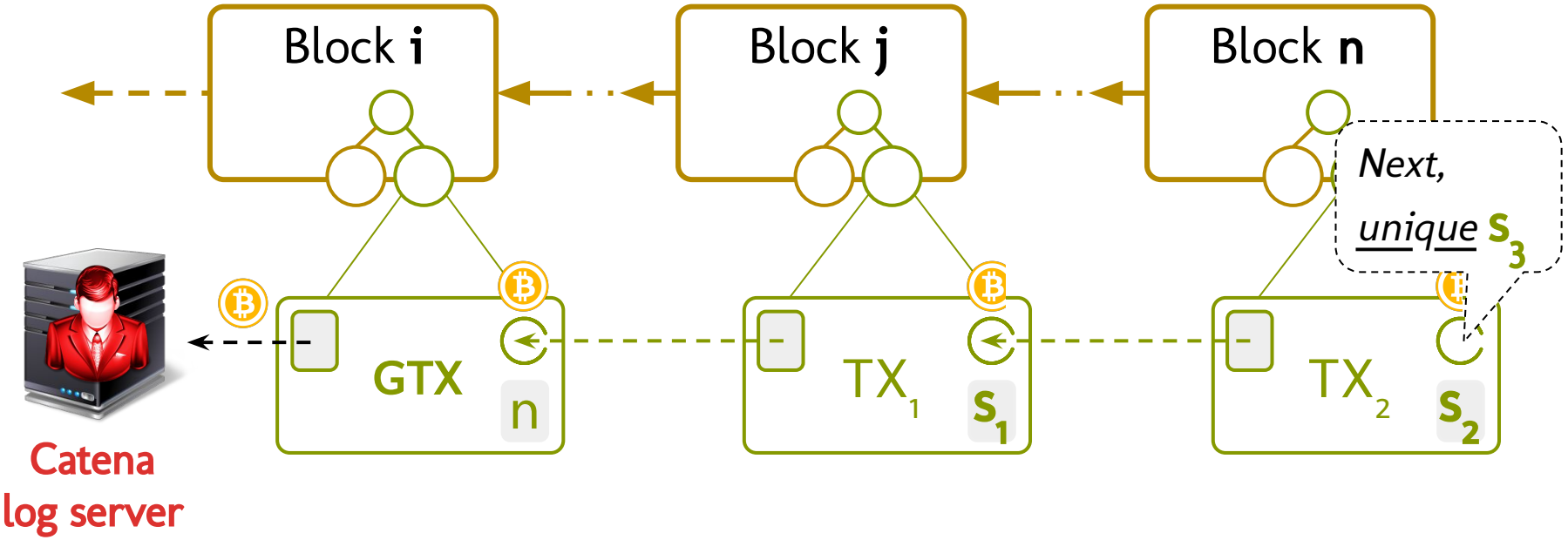
Catena design



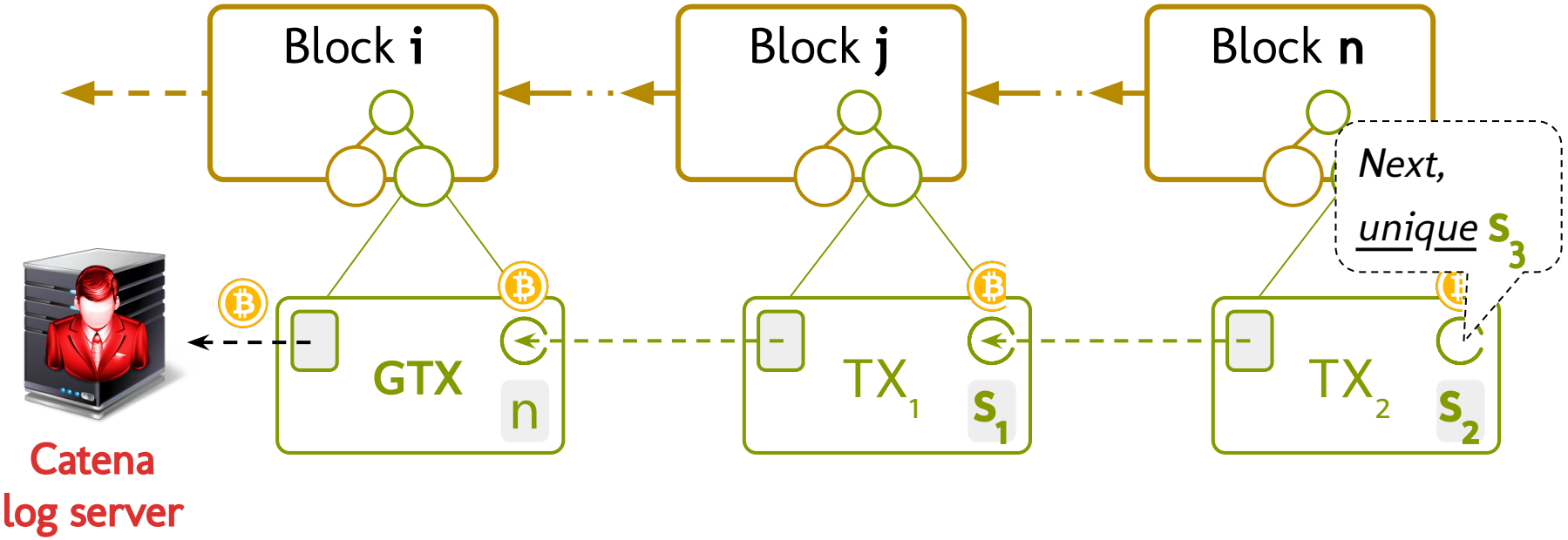
Catena design



Catena design



Catena design

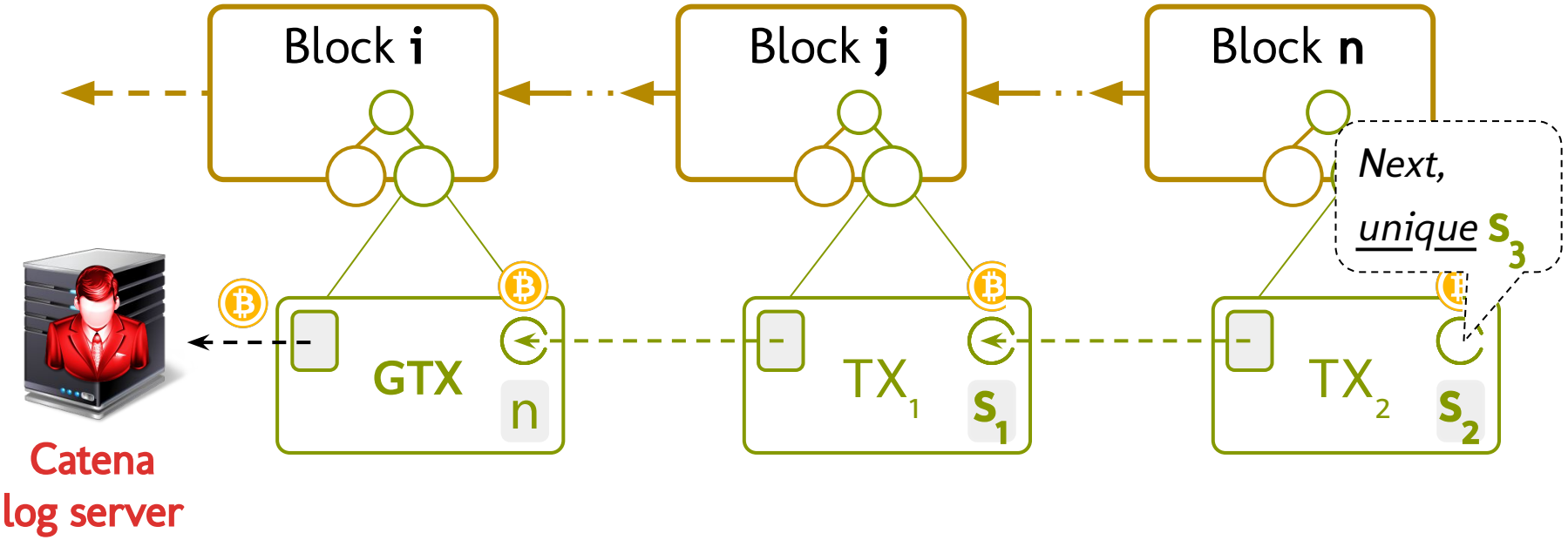


Advantages:

(1) Hard to fork

(2) Efficient to verify

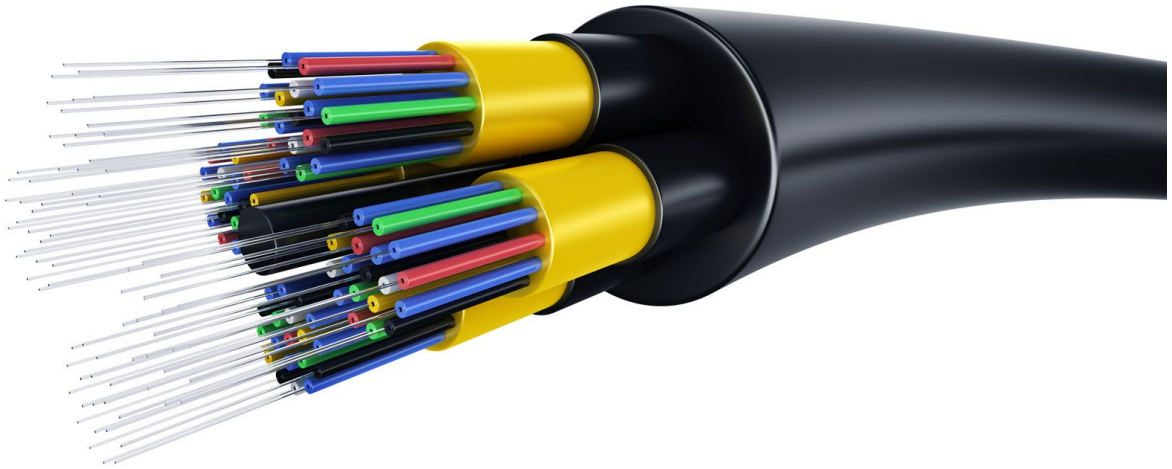
Catena design



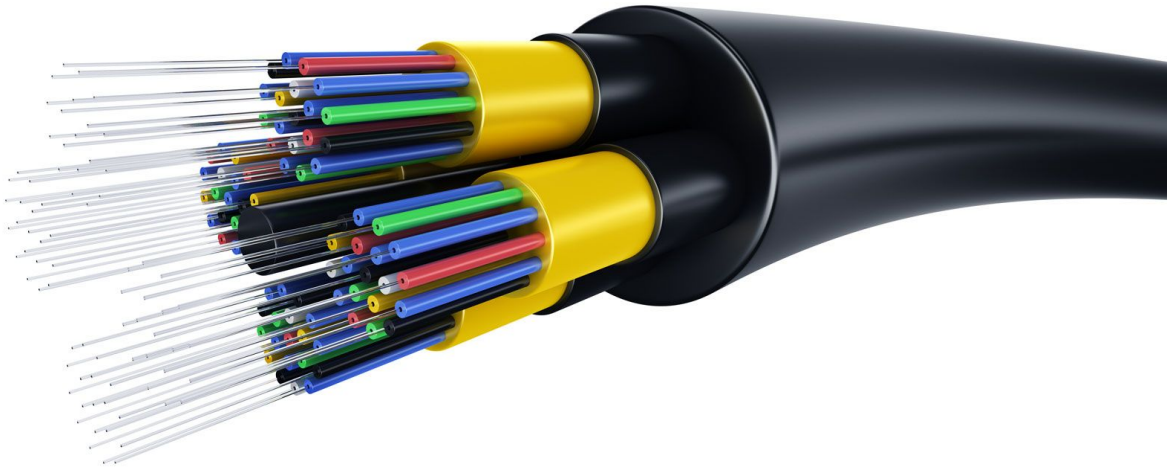
Advantages:
(1) Hard to fork
(2) Efficient to verify

Disadvantages:
(1) 6-block confirmation delay
(2) 1 statement every 10 minutes

Client bandwidth

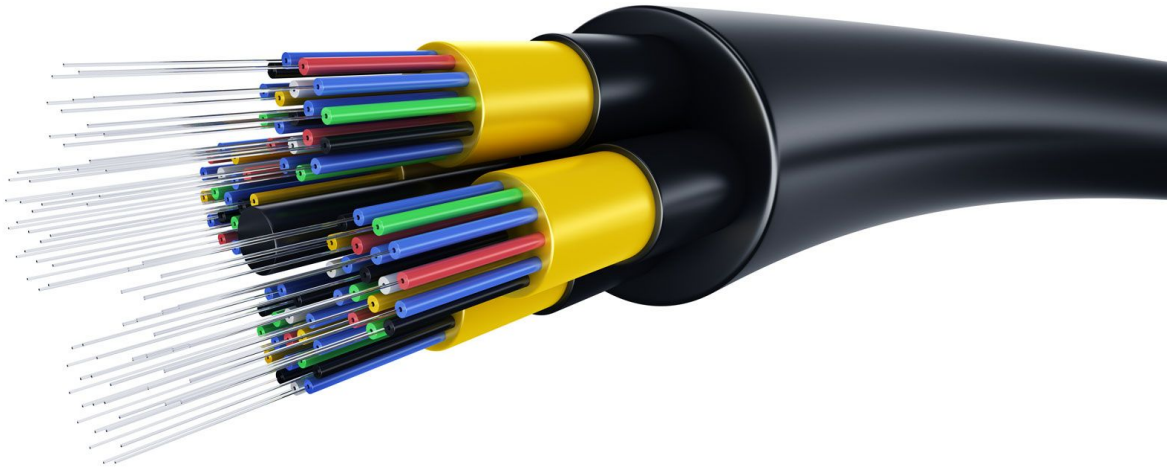


Client bandwidth



n block headers (80 bytes each) + **k** statements (around 600 bytes each)

Client bandwidth



e.g., **440K** block headers + **10K** statements = **~41** MB

(80 bytes each) (around 600 bytes each)

Conclusions

- Efficient Bitcoin witnessing is possible!
  - ~40 MB instead of 90 GB bandwidth overhead

Conclusions

- Efficient Bitcoin witnessing is possible!
  - ~40 MB instead of 90 GB bandwidth overhead
- Important applications
  - Public-key directories
  - Tor Consensus Transparency
  - Software transparency schemes

Conclusions

- Efficient Bitcoin witnessing is possible!
  - ~40 MB instead of 90 GB bandwidth overhead
- Important applications
  - Public-key directories
  - Tor Consensus Transparency
  - Software transparency schemes
- Publicly-verifiable consensus like Bitcoin should be leveraged by applications, efficiently.

The end.

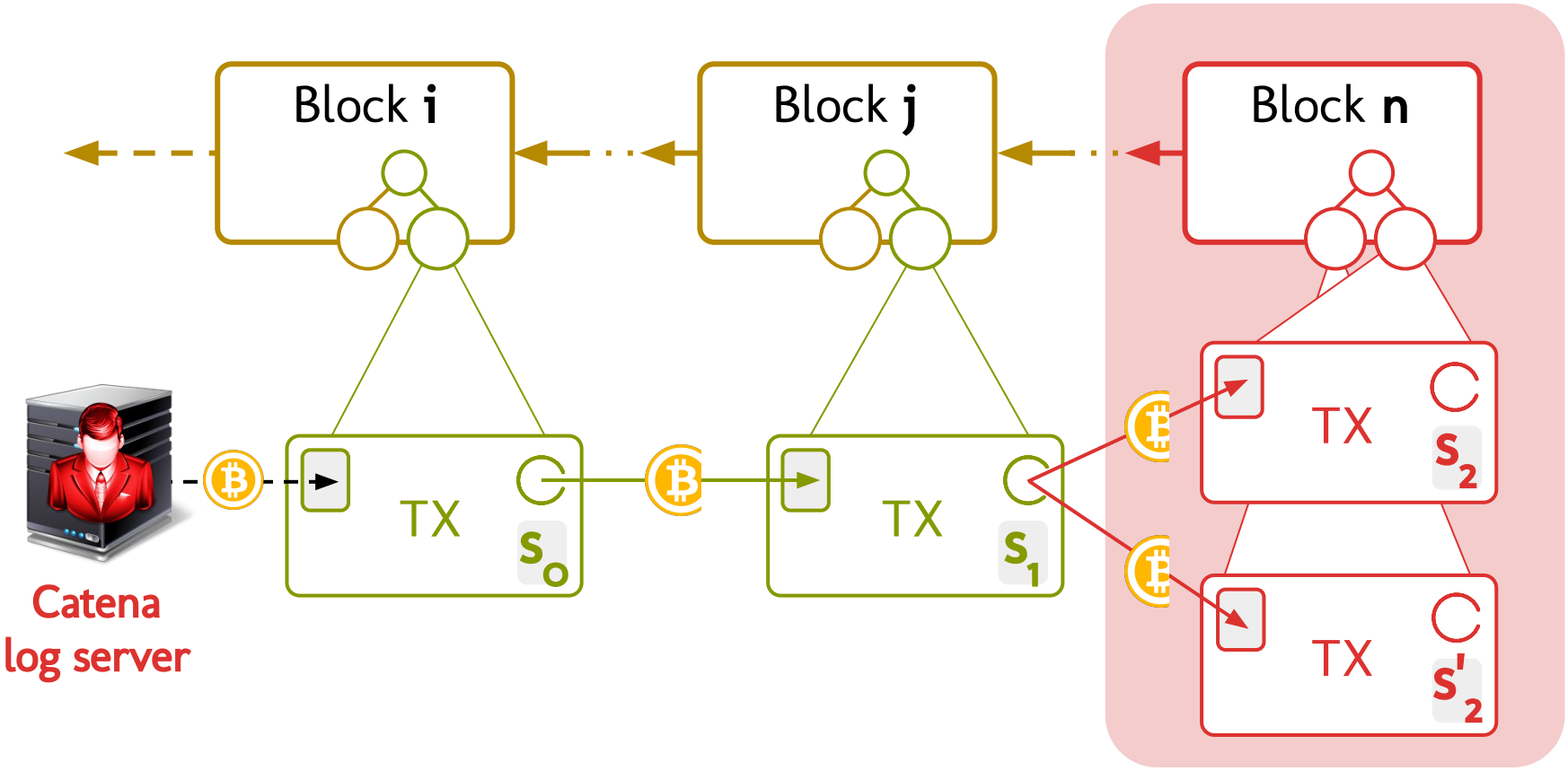
Extra slides

…just in case.

Catena: Preventing forks

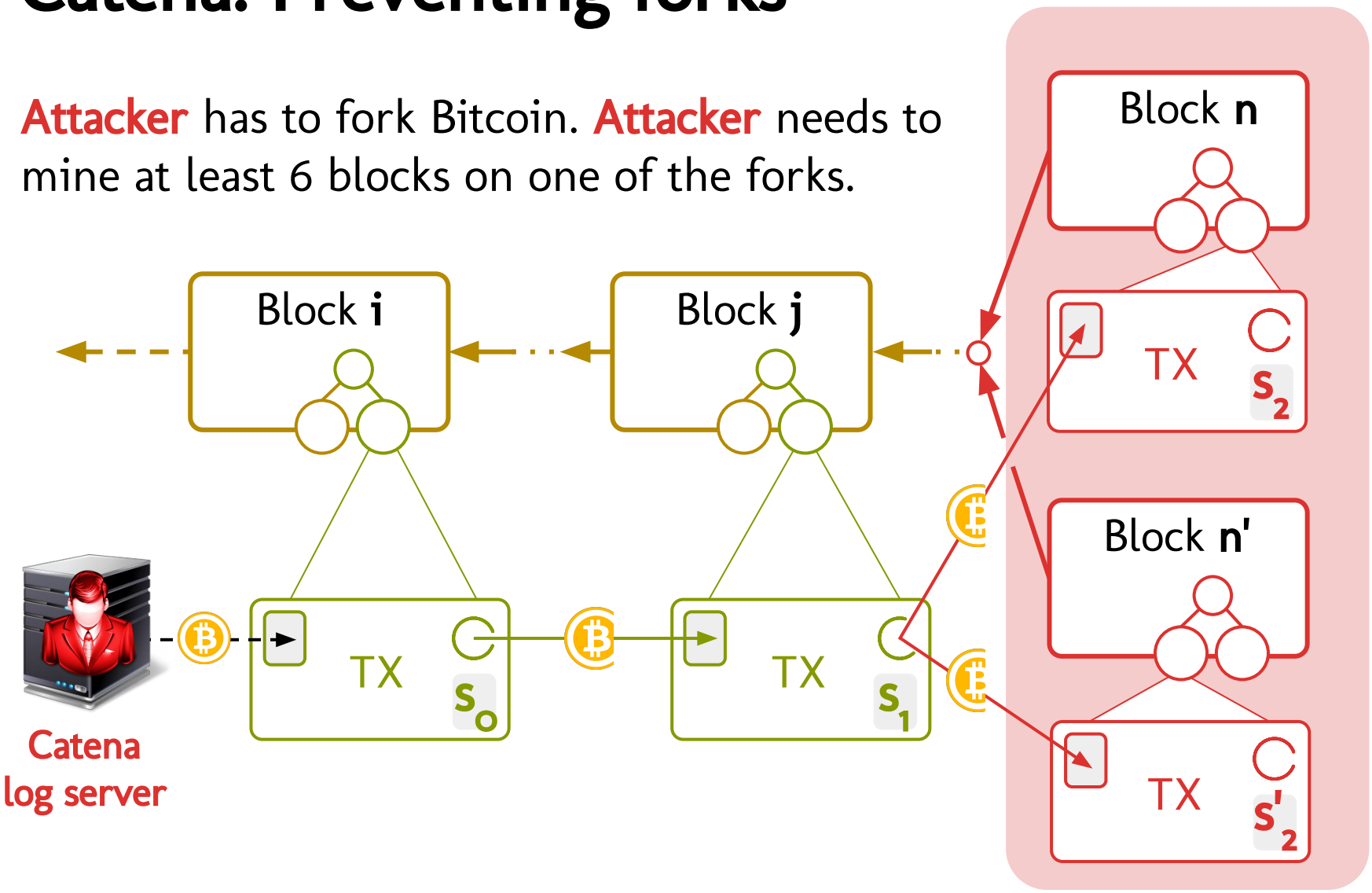
Attacker has to create an invalid block **n** to fork the log ⇒ **Attacker** has to fork Bitcoin.

Invalid block: Breaks miner-enforced **TXO invariant.**



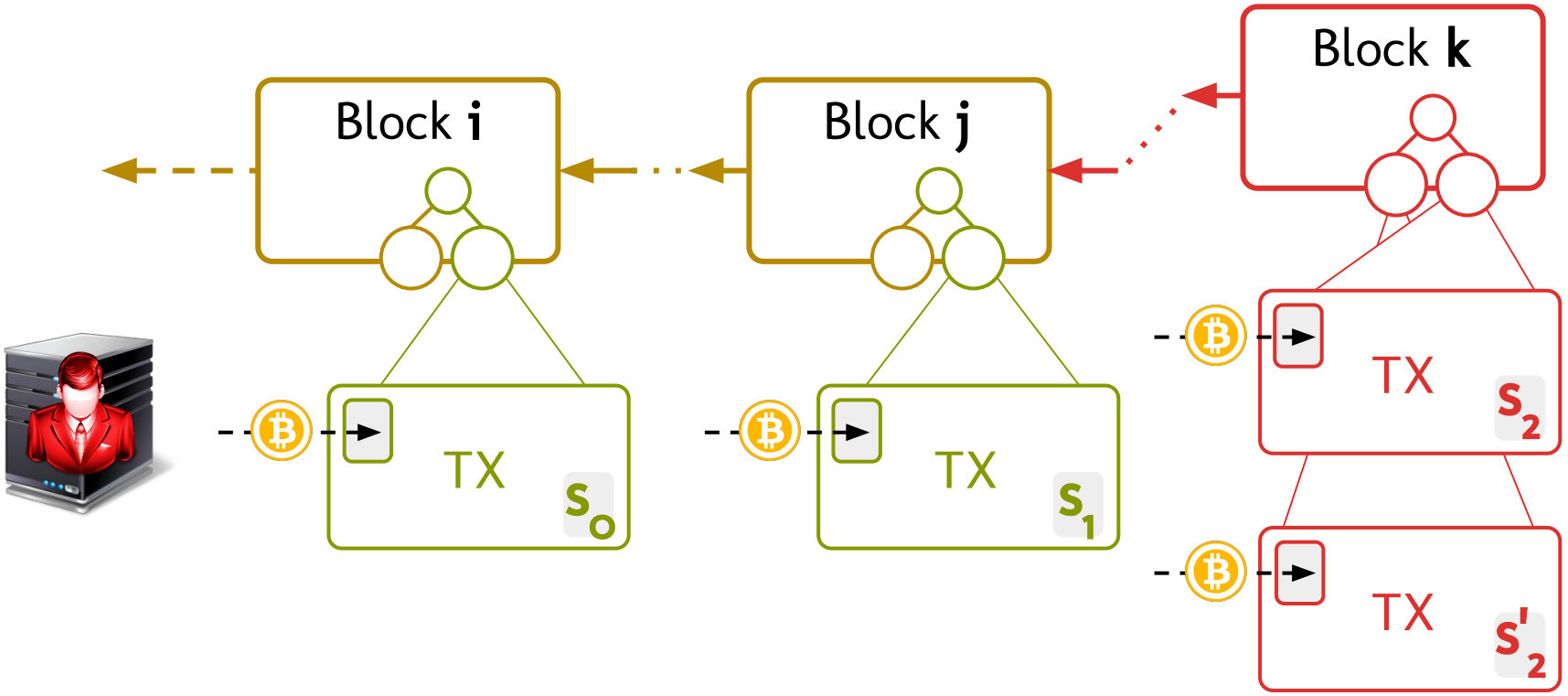
Catena: Preventing forks

Attacker has to fork Bitcoin. **Attacker** needs to mine at least 6 blocks on one of the forks.



Key idea

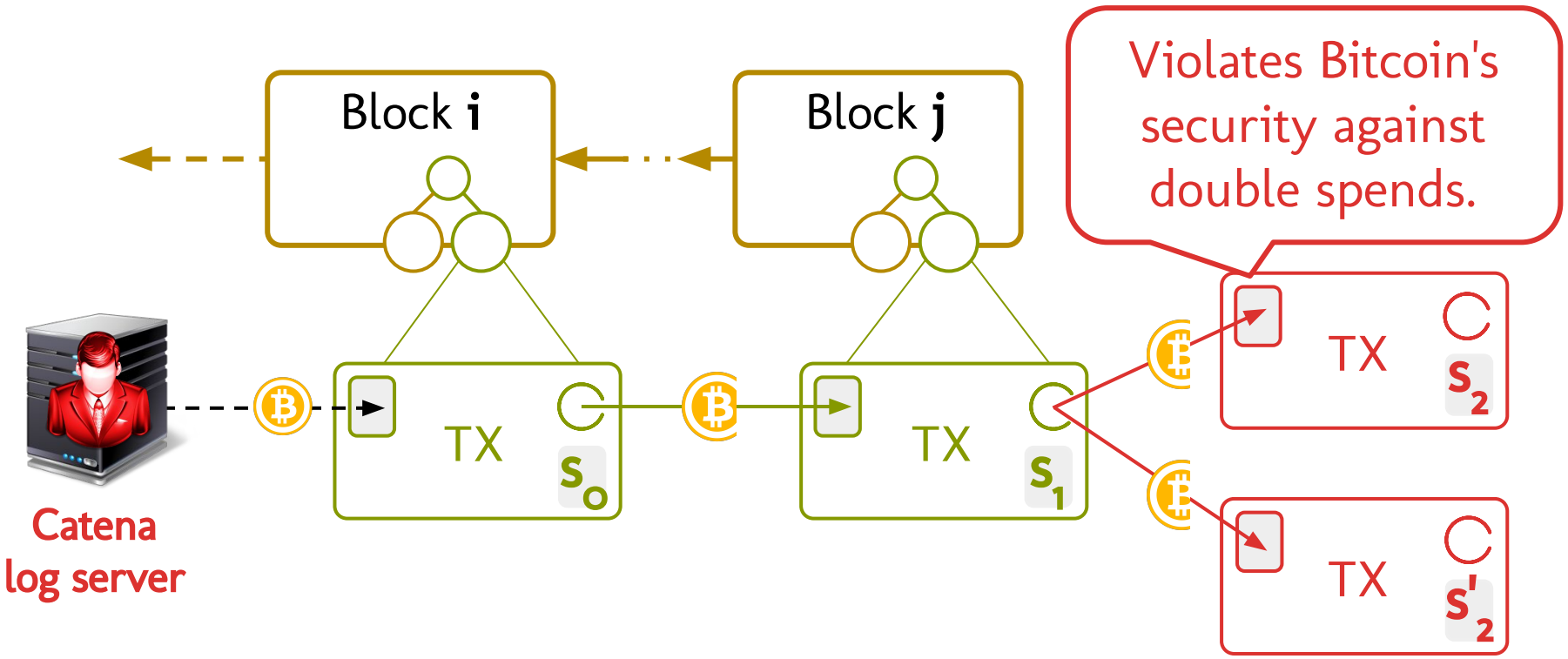
Q: How can we prove to a thin client that there's no s'_2?



Key idea

Q: How can we prove to a thin client that there's no s'_2?

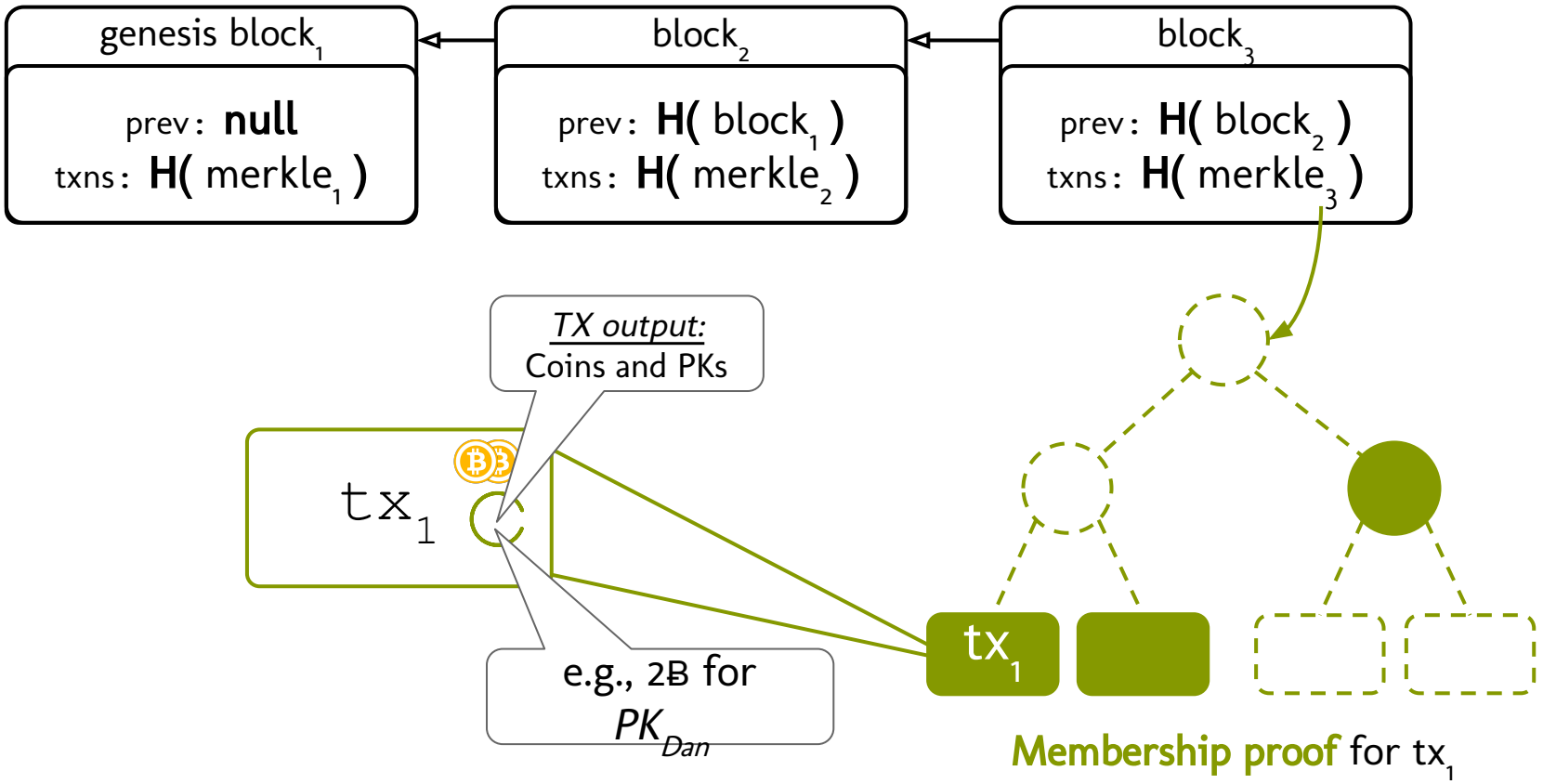
A: Leverage Bitcoin's mechanism against double-spends!



Graveyard

...where old slides rest in peace.

Bitcoin background



Bitcoin background