Tuesday, May 23rd, 2017

Catena: Efficient Non-equivocation via bitcoin

Alin Tomescu
alinush@mit.edu

Srinivas Devadas
devadas@mit.edu

IEEE Security & Privacy 2017, San Jose, CA

CSAIL

What is non-equivocation?

What is non-equivocation?

- At time **i**, publishes *digest* s_i

Public-key directory

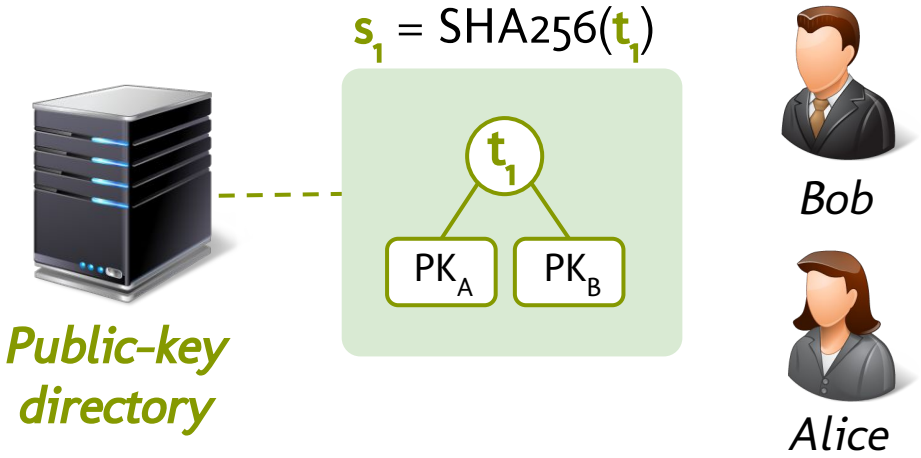
What is non-equivocation?

- At time **i**, publishes a **single** *digest* s_i

Public-key directory

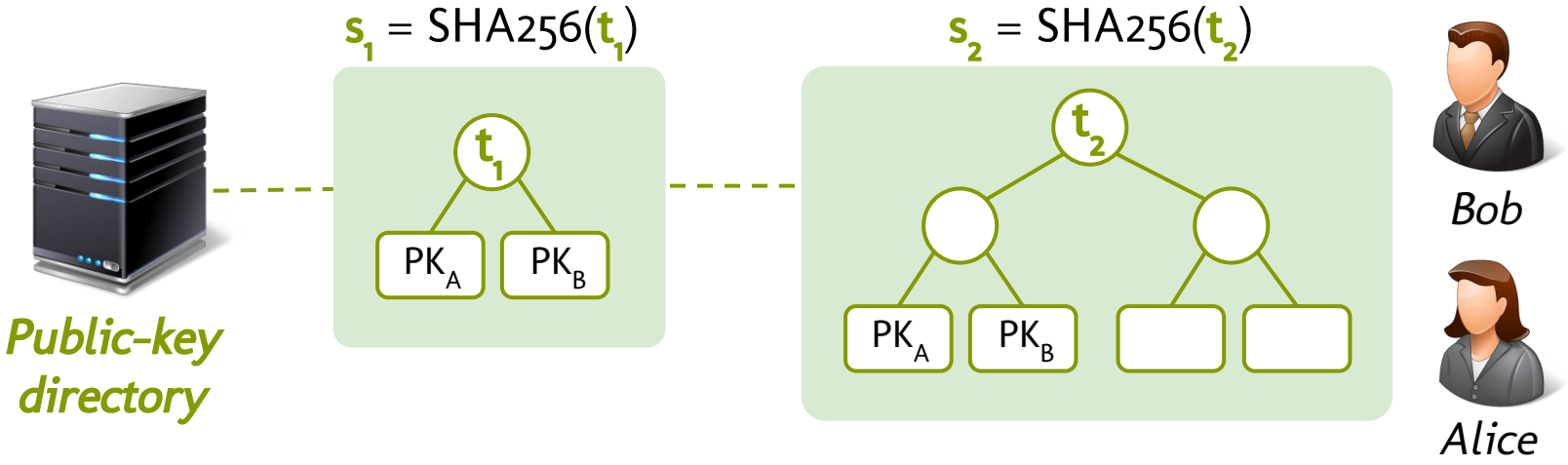
What is non-equivocation?

- At time **i**, publishes a **single** *digest* s_i
- At time **1**, Alice, Bob and others "see" s_1



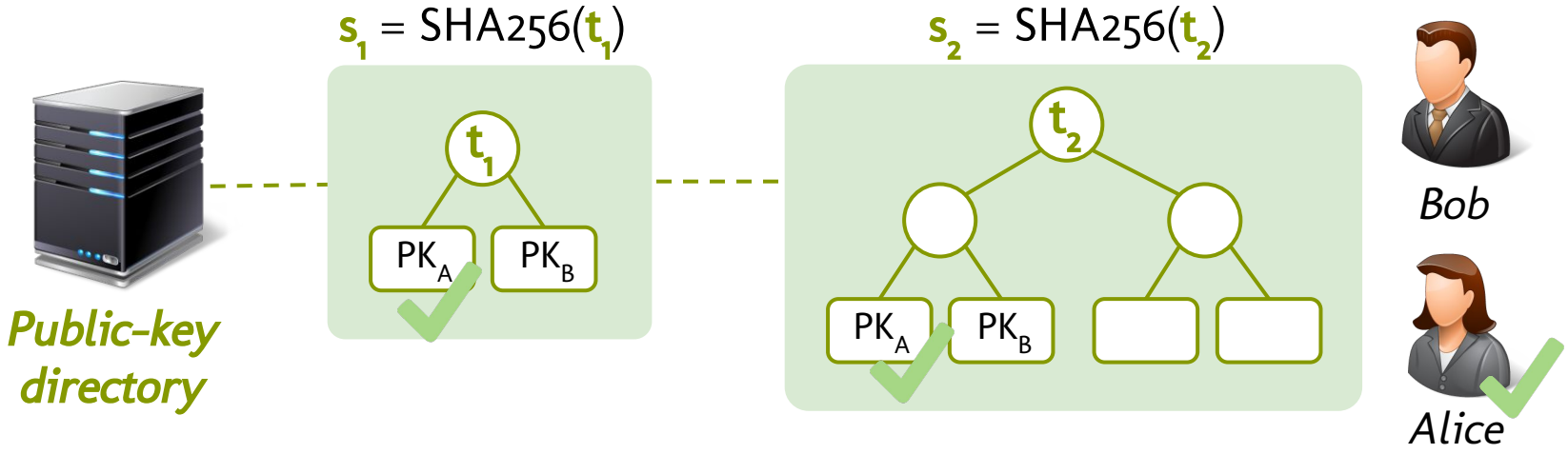
What is non-equivocation?

- At time 2, Alice, Bob and others "see" s_1 , s_2 , ...



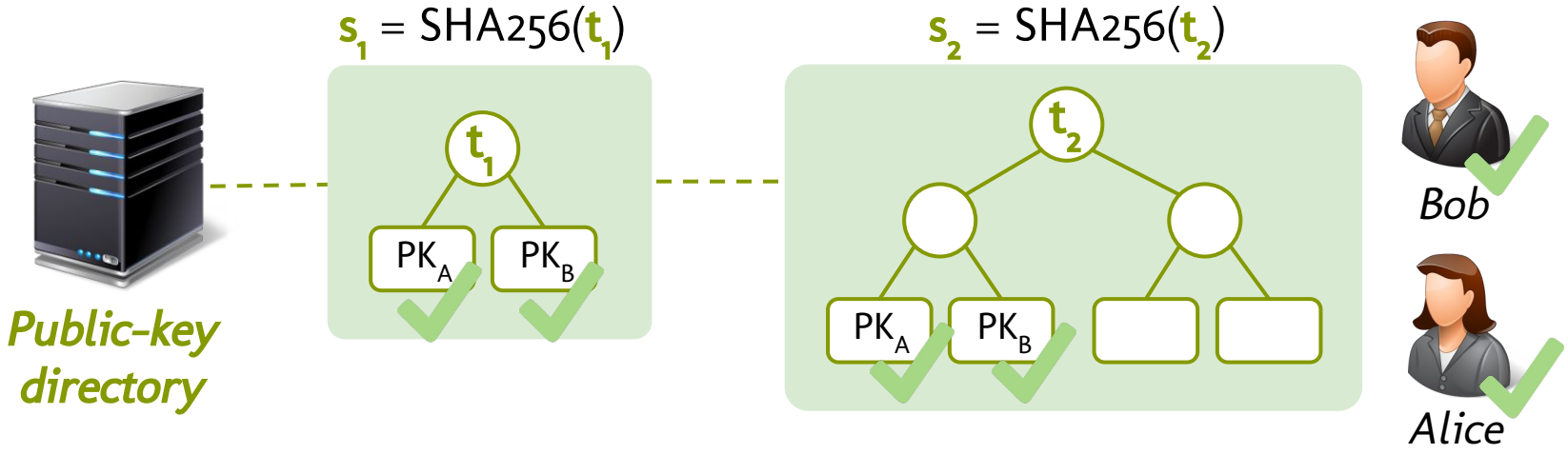
What is non-equivocation?

- Alice and Bob can "monitor" own PKs



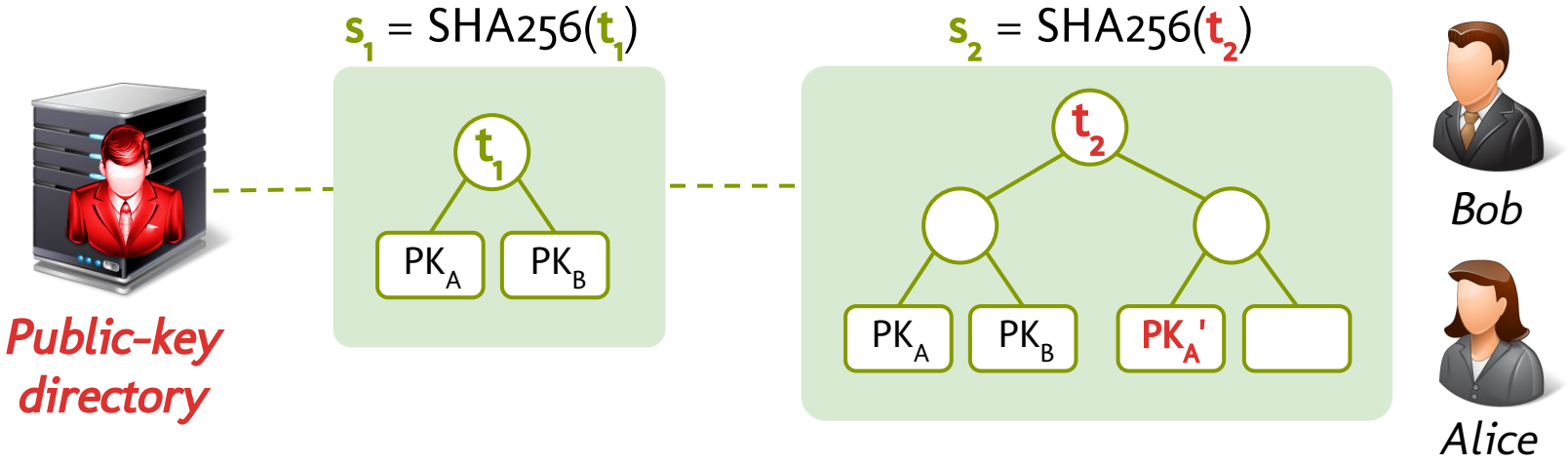
What is non-equivocation?

- Alice and Bob can "monitor" own PKs



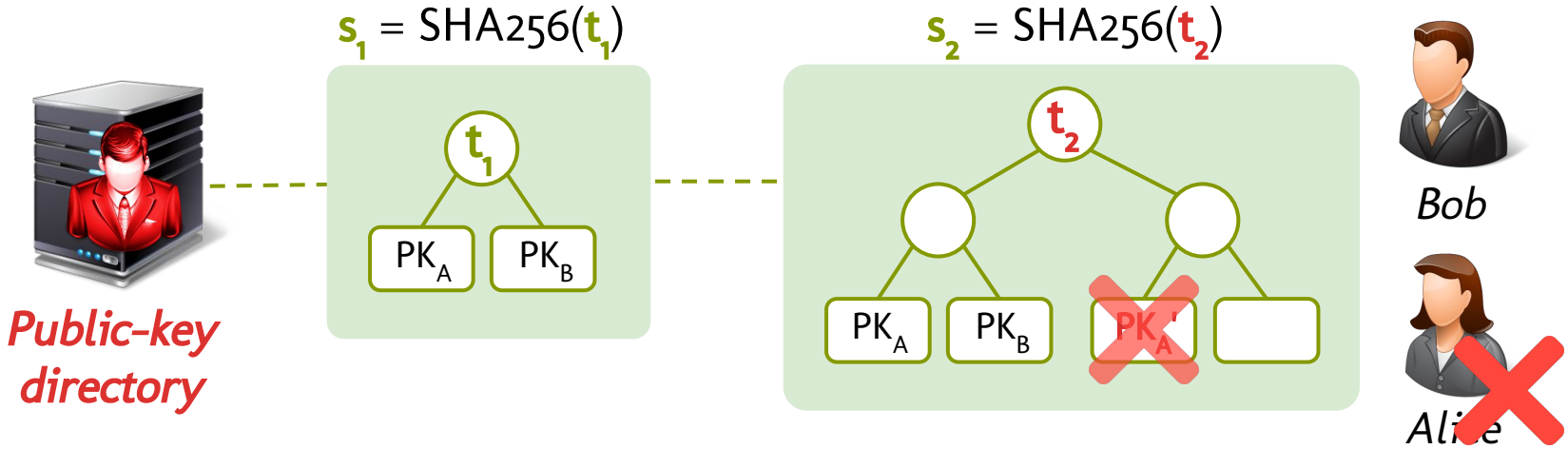
What is non-equivocation?

- Alice and Bob can "monitor" own PKs
- ...and **server** has to impersonate in plain sight



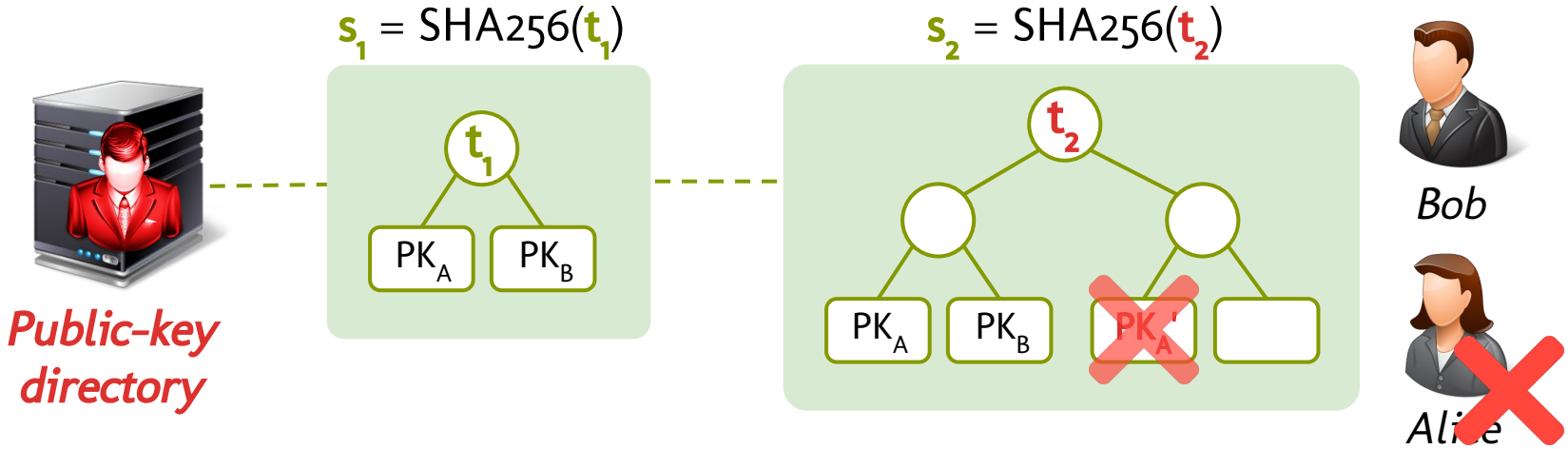
What is non-equivocation?

- Alice and Bob can "monitor" own PKs
- ...and **server** has to impersonate in plain sight



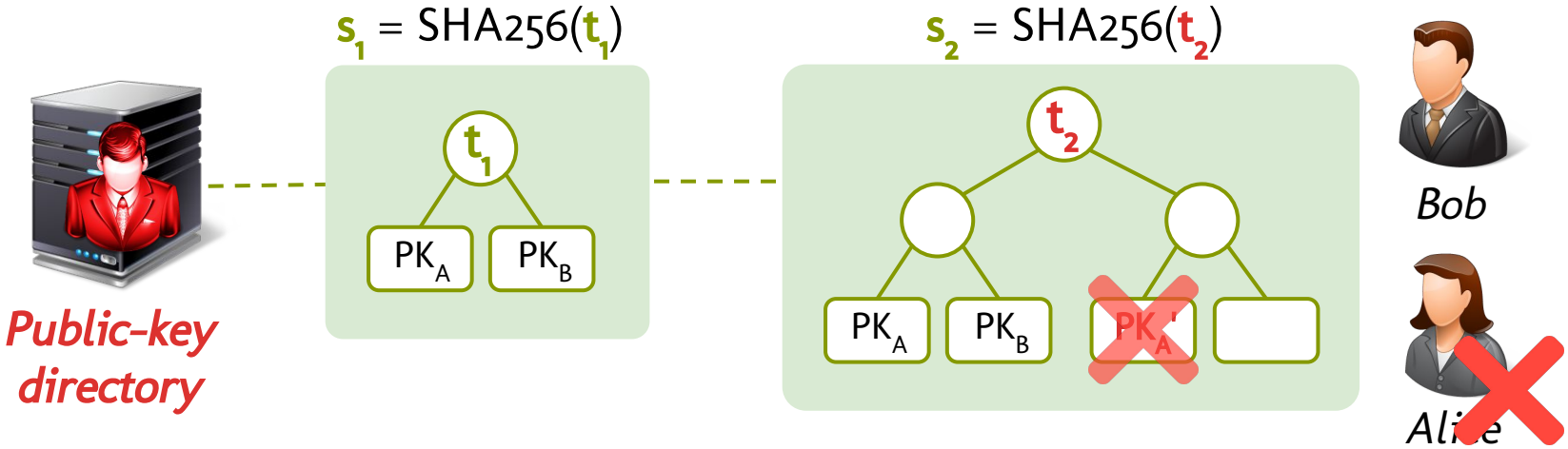
What is non-equivocation?

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



What is non-equivocation?

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

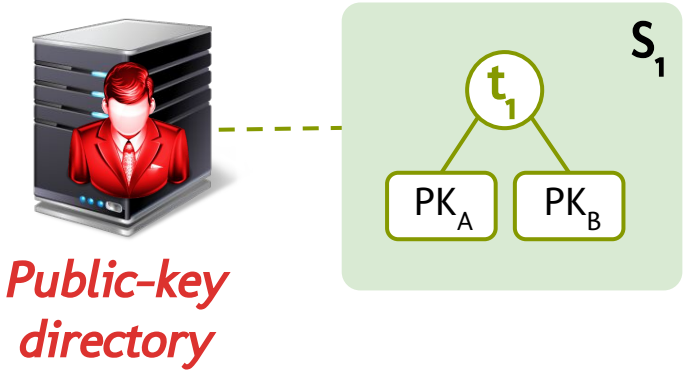


Including statements that are

incorrect at the application-layer

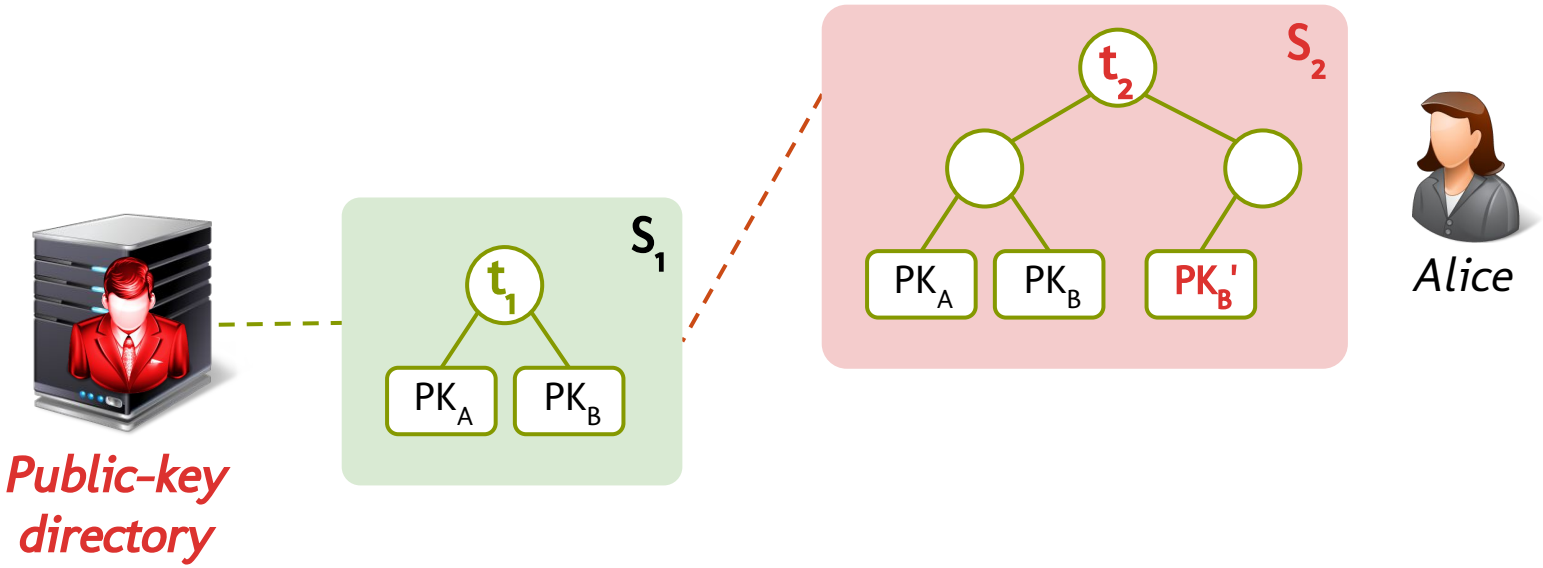
What is equivocation?

- At time **2**, malicious **server** publishes s_2 and s_2'



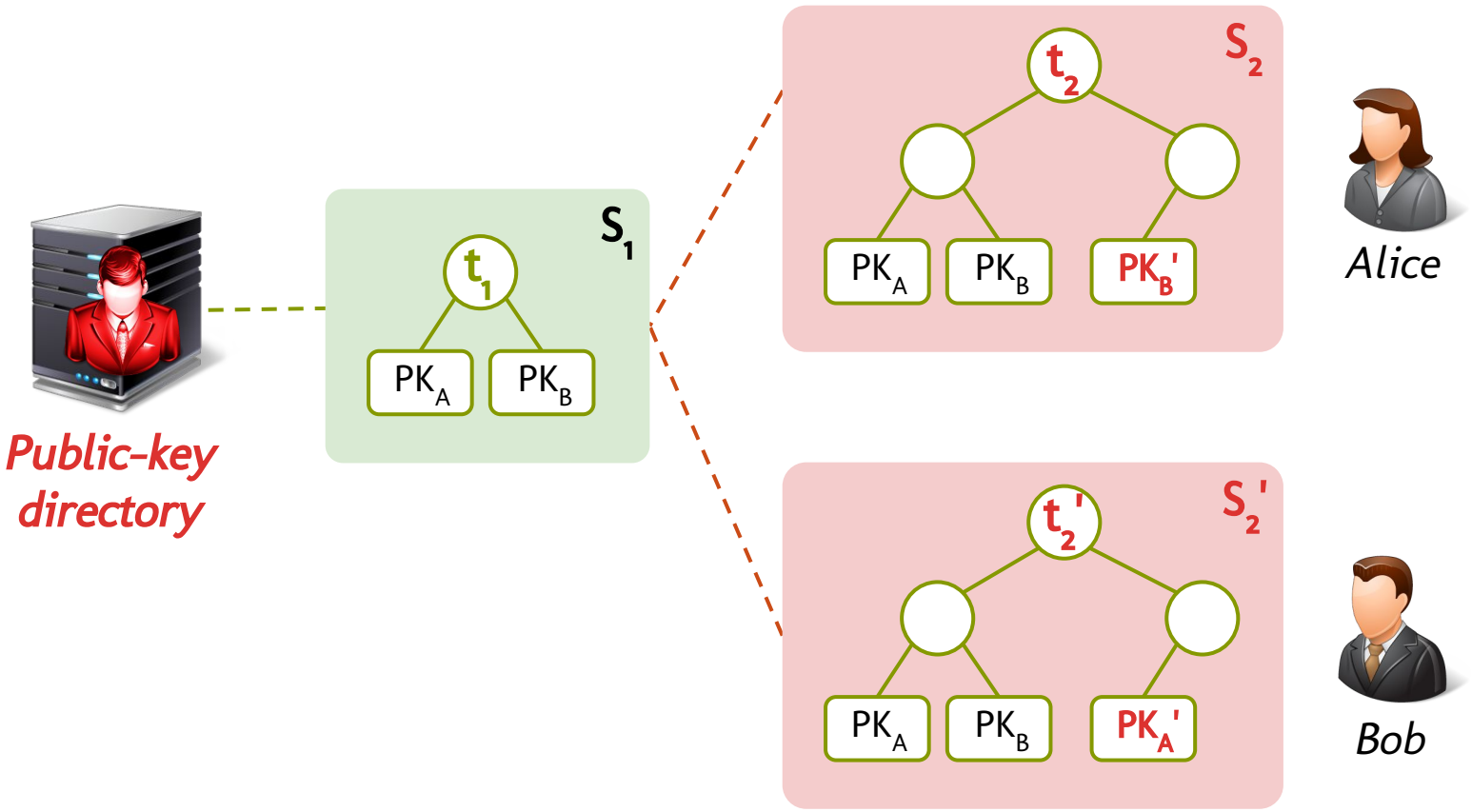
What is equivocation?

- s_2: Leave Alice's key intact, add fake PK_B' for Bob



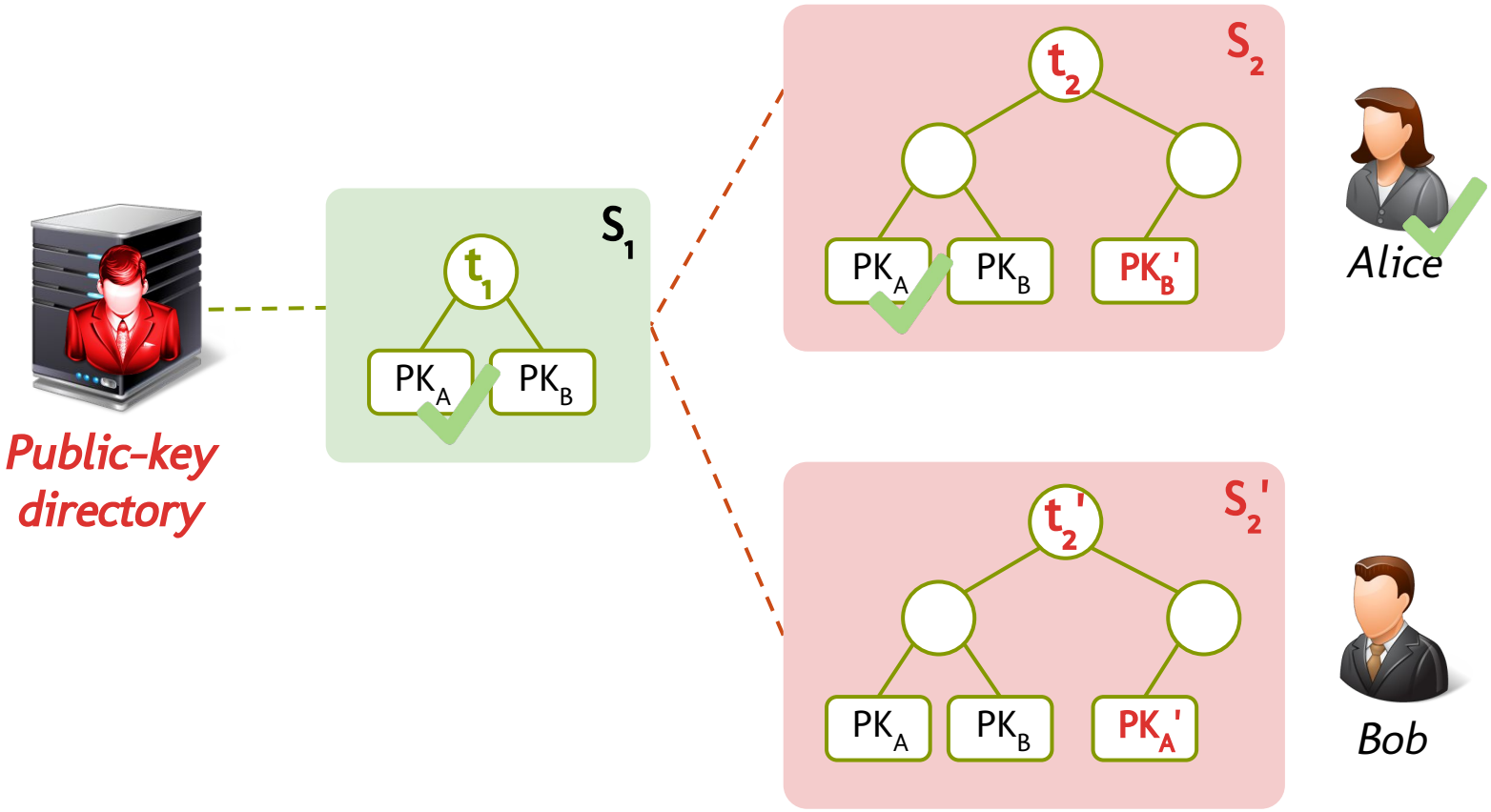
What is equivocation?

- s_2': Leave Bob's key intact, add fake PK_A' for Alice



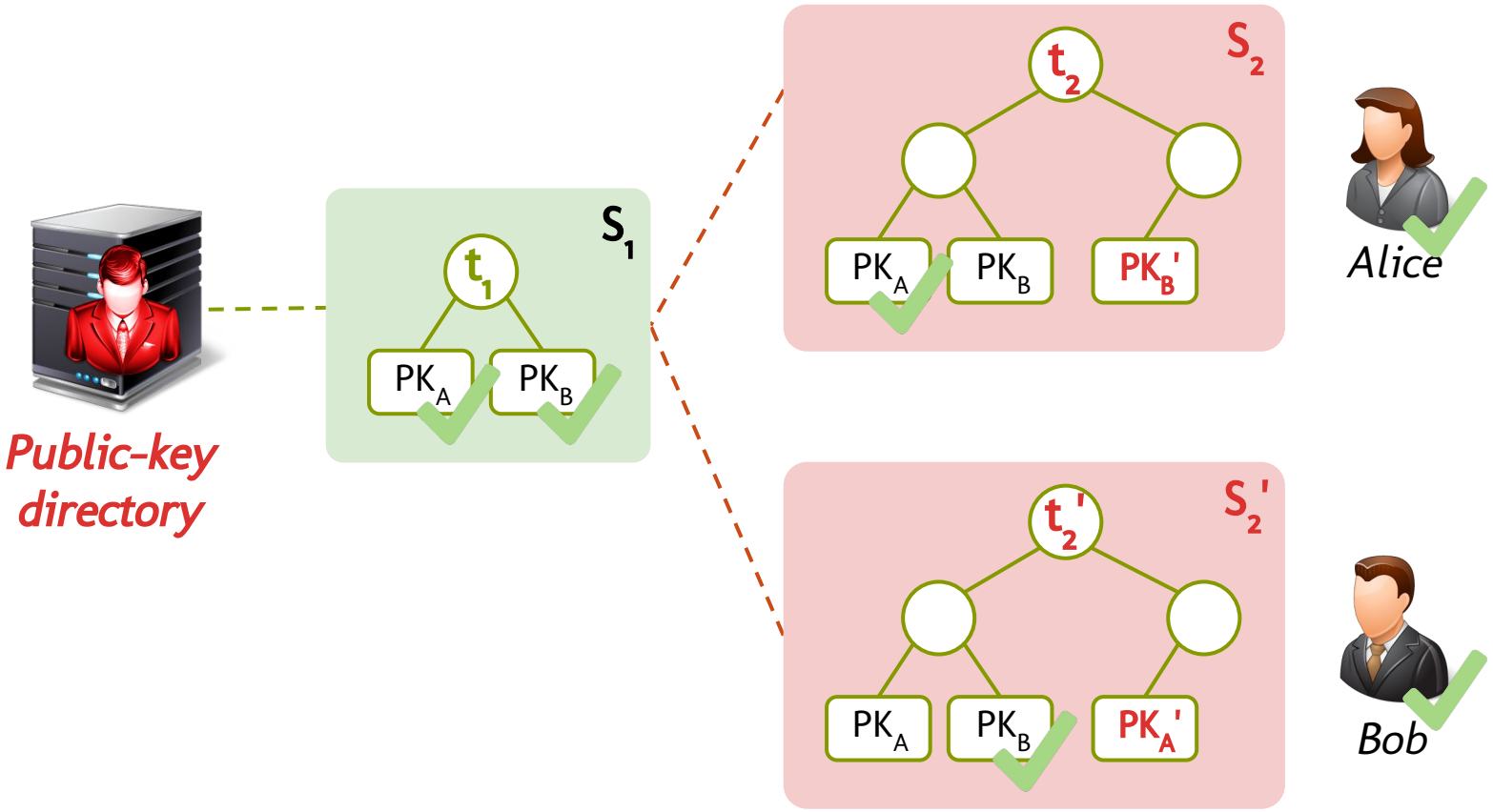
What is equivocation?

- Alice not impersonated in her view, but Bob is.



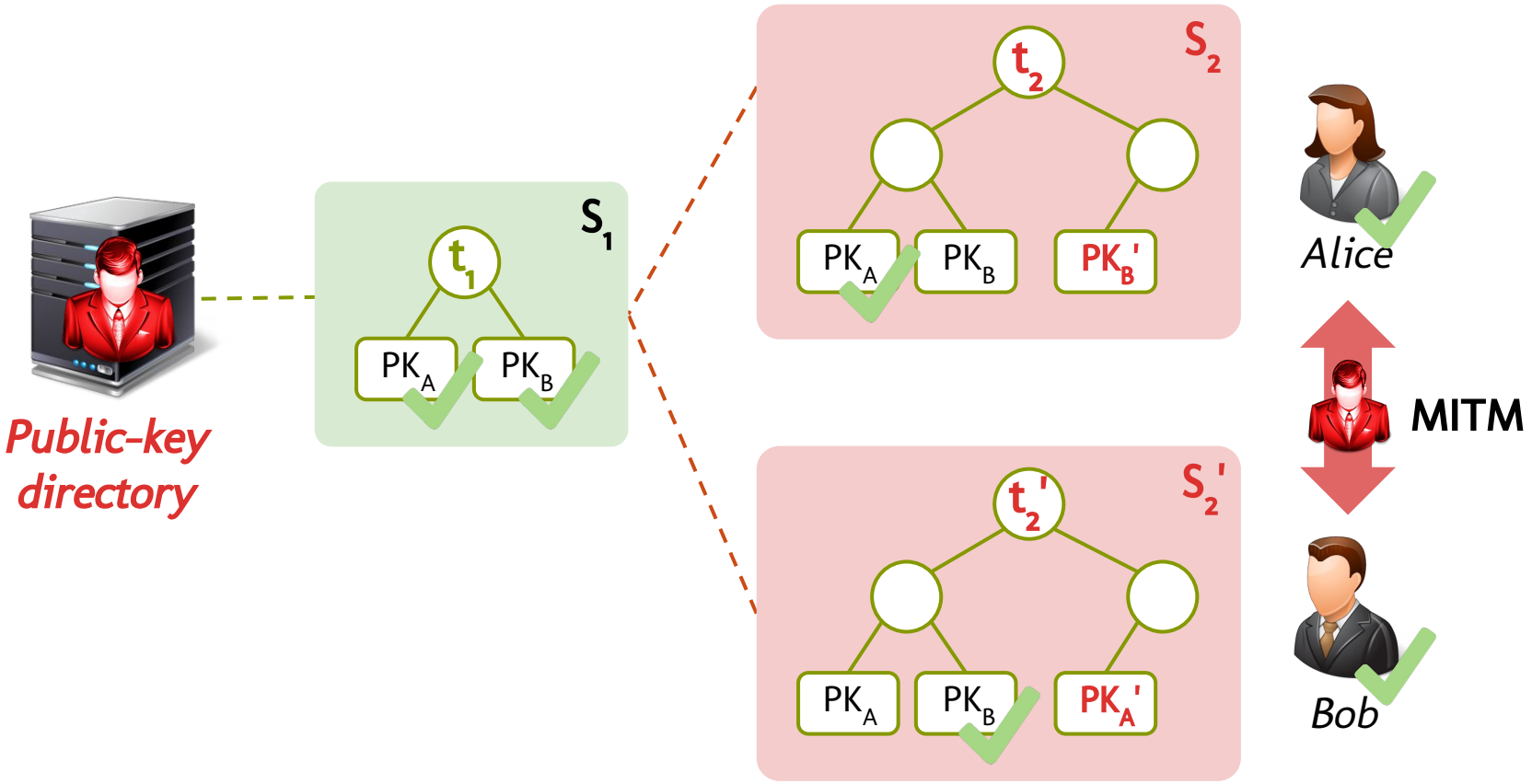
What is equivocation?

- Bob not impersonated in his view, but Alice is.



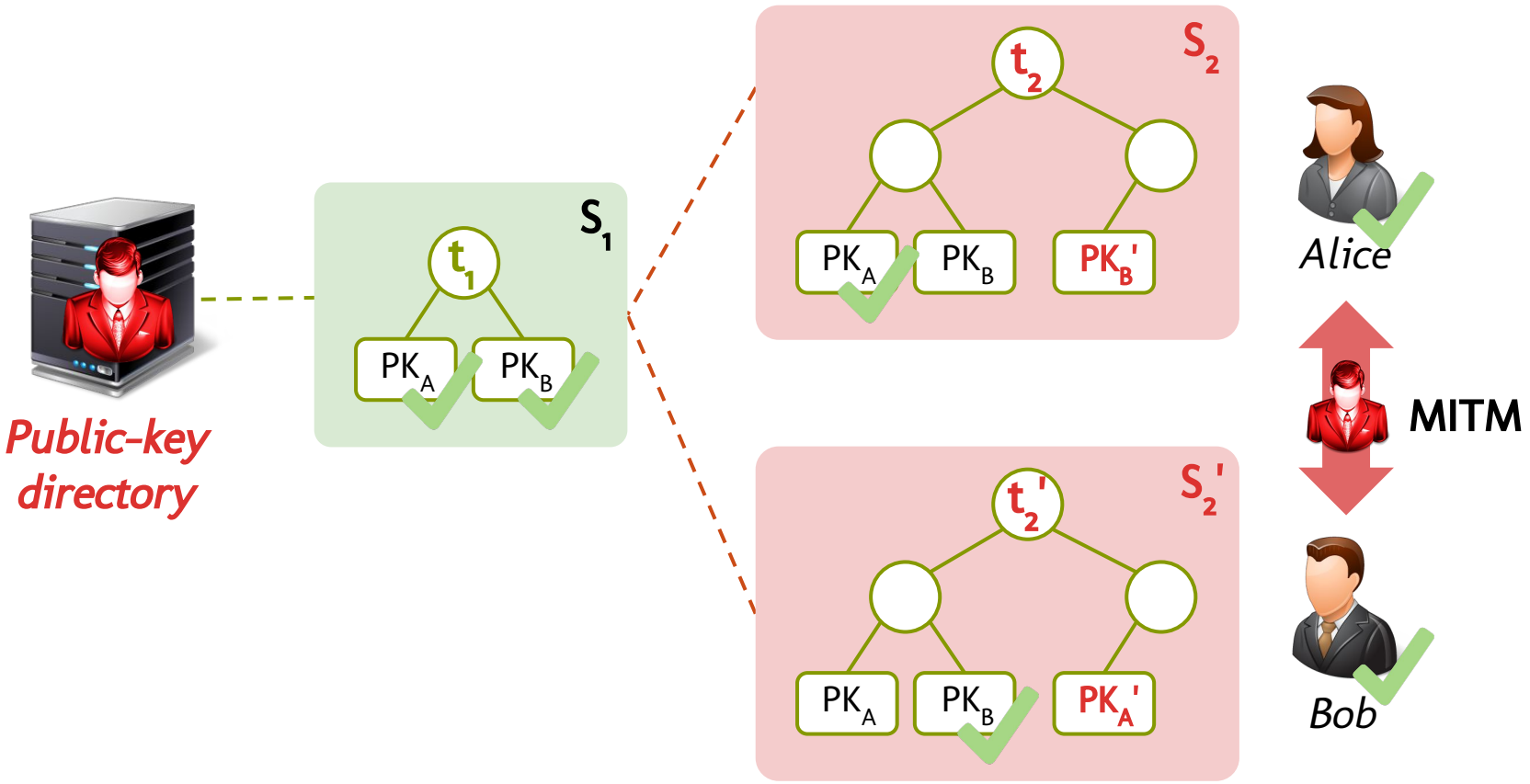
What is equivocation?

- Obtain fake keys for each other ⇒ MITM



What is equivocation?

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



Where is non-equivocation necessary?

Public-key distribution (PKD)

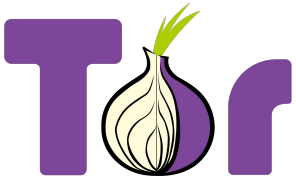
- HTTPS
- Secure messaging
- *"We assume a PKI."*

Where is non-equivocation necessary?

Public-key distribution (PKD)

- HTTPS
- Secure messaging
- *"We assume a PKI."*

Tor Directory Servers

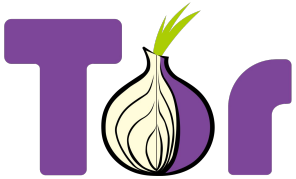


Where is non-equivocation necessary?

Public-key distribution (PKD)

- HTTPS
- Secure messaging
- *"We assume a PKI."*

Tor Directory Servers



Software transparency schemes

- Attacks on Bitcoin binaries

0.13.0 Binary Safety Warning

17 August 2016

Summary

Bitcoin.org has reason to suspect that the binaries for the upcoming Bitcoin Core release will likely be targeted by state sponsored attackers. As a website, Bitcoin.org does not have the technical resources to guarantee that we can defend ourselves from attackers of this calibre. We ask the Bitcoin community, and in particular the Chinese Bitcoin community to be extra vigilant when downloading binaries from our website.

Contributions

Contributions

- Bitcoin-based append-only log,

Contributions

- Bitcoin-based append-only log,
- ...as hard-to-fork as the Bitcoin blockchain
  - Want to fork? Do some work!

Contributions

- Bitcoin-based append-only log,
- ...as hard-to-fork as the Bitcoin blockchain
  - Want to fork? Do some work!
- ...but efficiently auditable

Contributions

- Bitcoin-based append-only log,
- ...as hard-to-fork as the Bitcoin blockchain
  - Want to fork? Do some work!
- ...but efficiently auditable
  - 600 bytes / statement (e.g., PKD digests)
  - 80 bytes / Bitcoin block

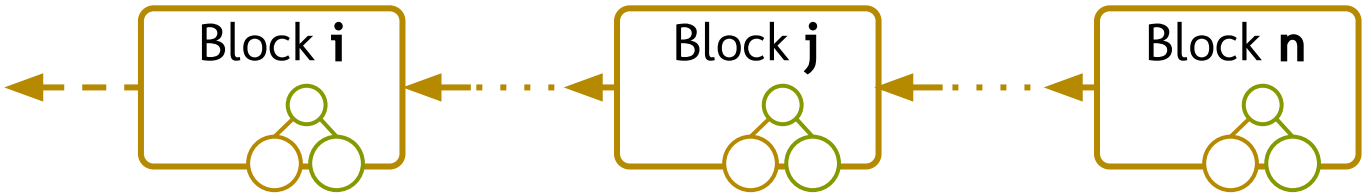
Contributions

- Bitcoin-based append-only log,
- ...as hard-to-fork as the Bitcoin blockchain
  - Want to fork? Do some work!
- ...but efficiently auditable
  - 600 bytes / statement (e.g., PKD digests)
  - 80 bytes / Bitcoin block
- Java implementation (3500 SLOC)

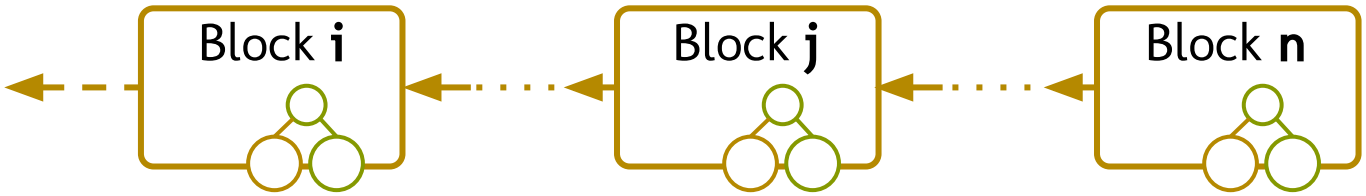
Outline

1. Bitcoin background
2. Previous work
3. Catena design
4. Catena scalability

Bitcoin blockchain

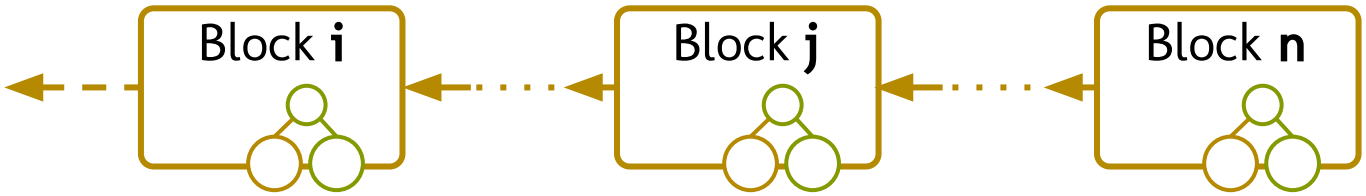


Bitcoin blockchain



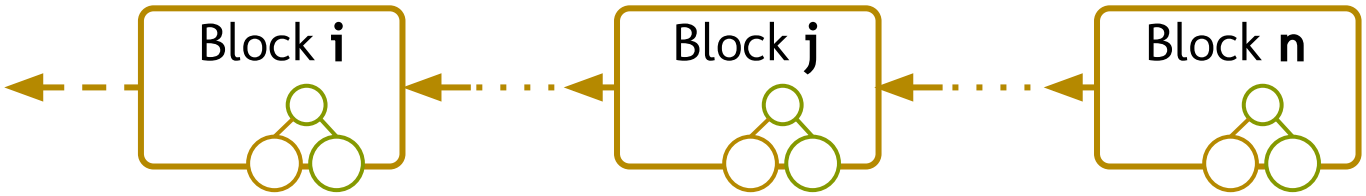
- Hash chain of blocks

Bitcoin blockchain



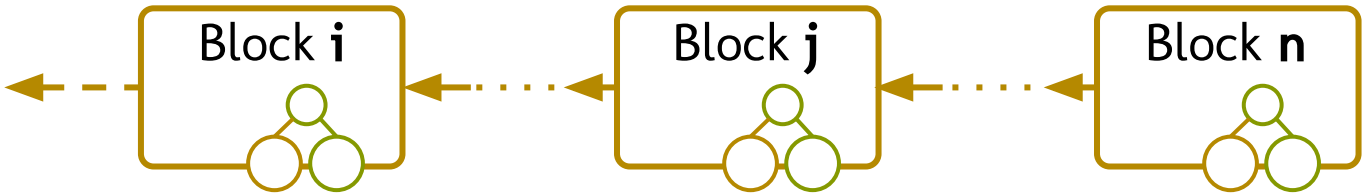
- Hash chain of blocks
  - Arrows are *hash pointers*

Bitcoin blockchain



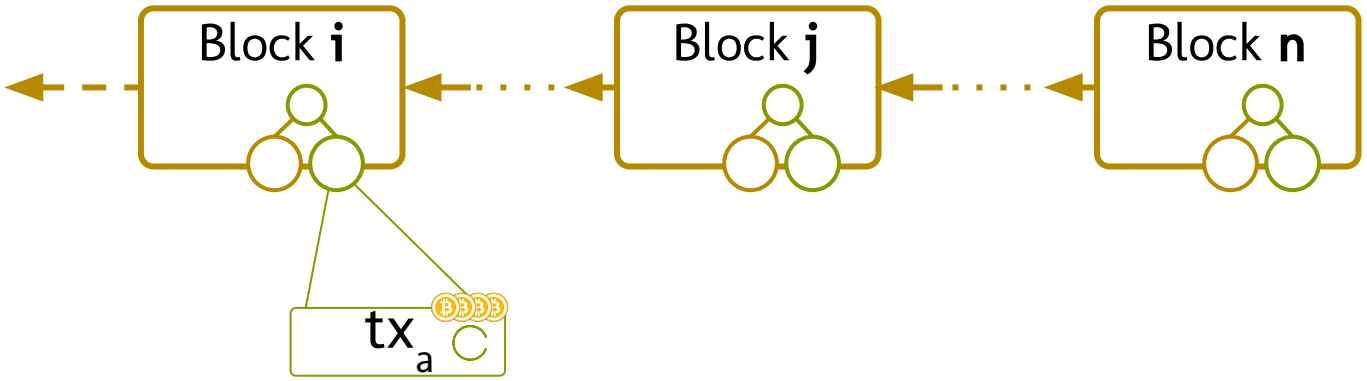
- Hash chain of blocks
  - Arrows are *hash pointers*
- Merkle tree of TXNs in each block

Bitcoin blockchain



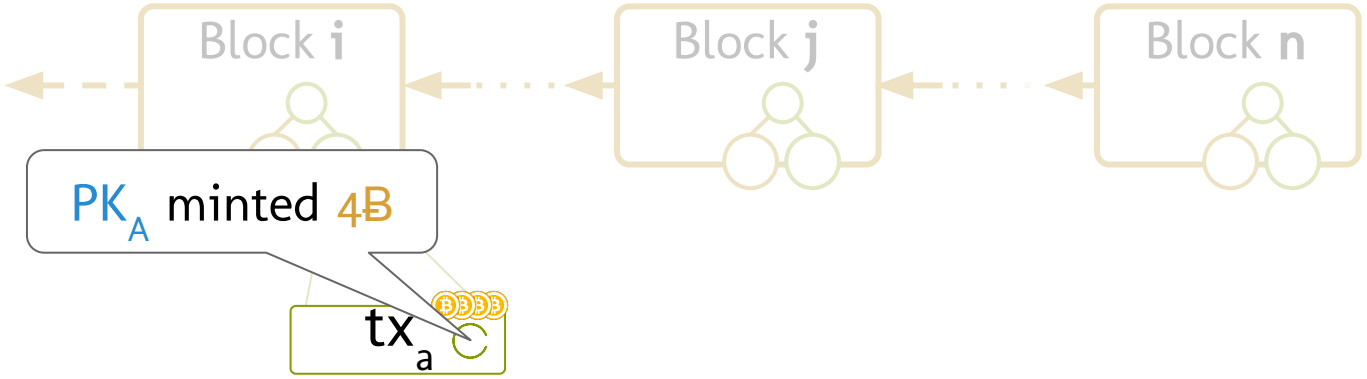
- Hash chain of blocks
  - Arrows are *hash pointers*
- Merkle tree of TXNs in each block
- Proof-of-work (PoW) consensus

Bitcoin blockchain



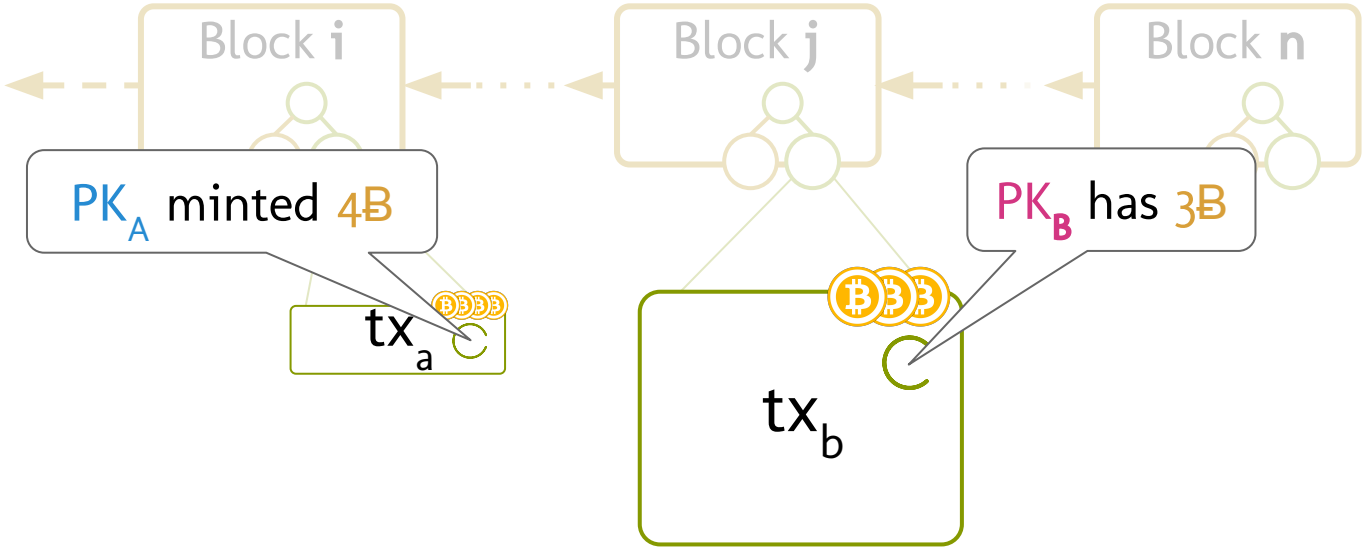
- Transactions mint coins

Bitcoin blockchain



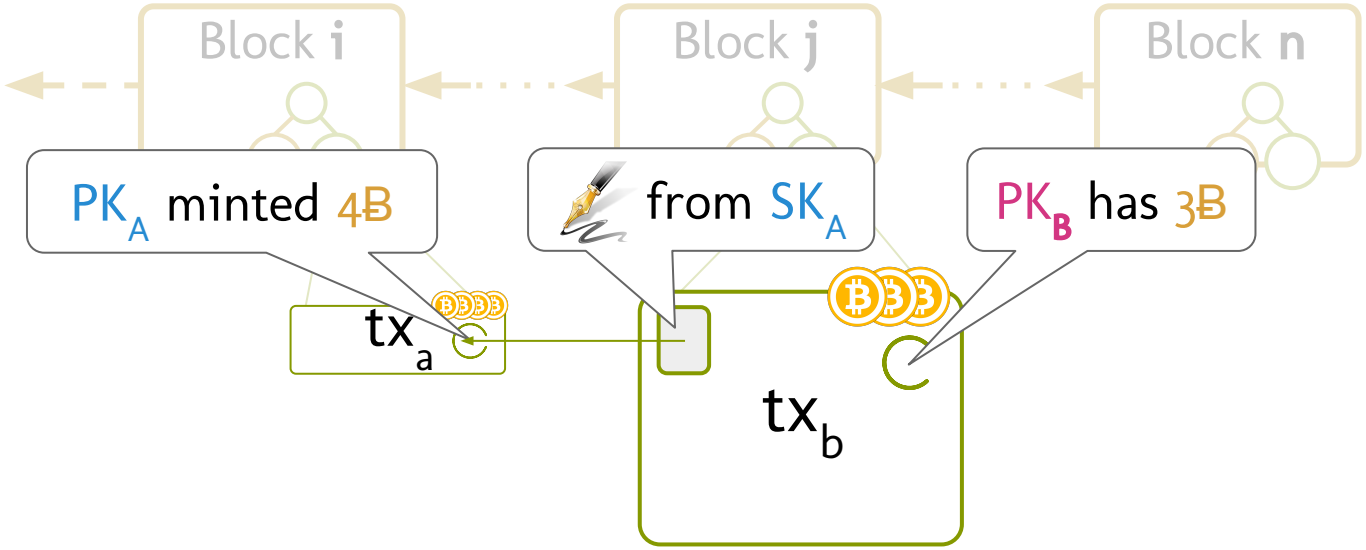
- Transactions mint coins
- Output = # of coins and owner's PK

Bitcoin blockchain



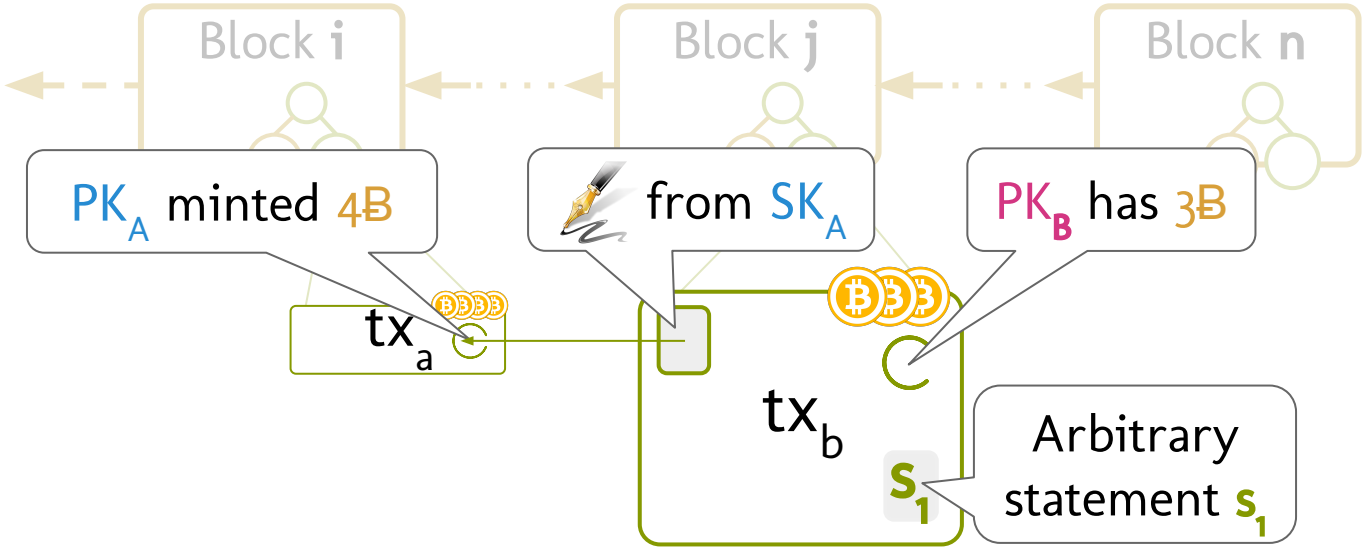
- Transactions mint coins
- Output = # of coins and owner's PK
- Transactions transfer coins (and pay fees)

Bitcoin blockchain



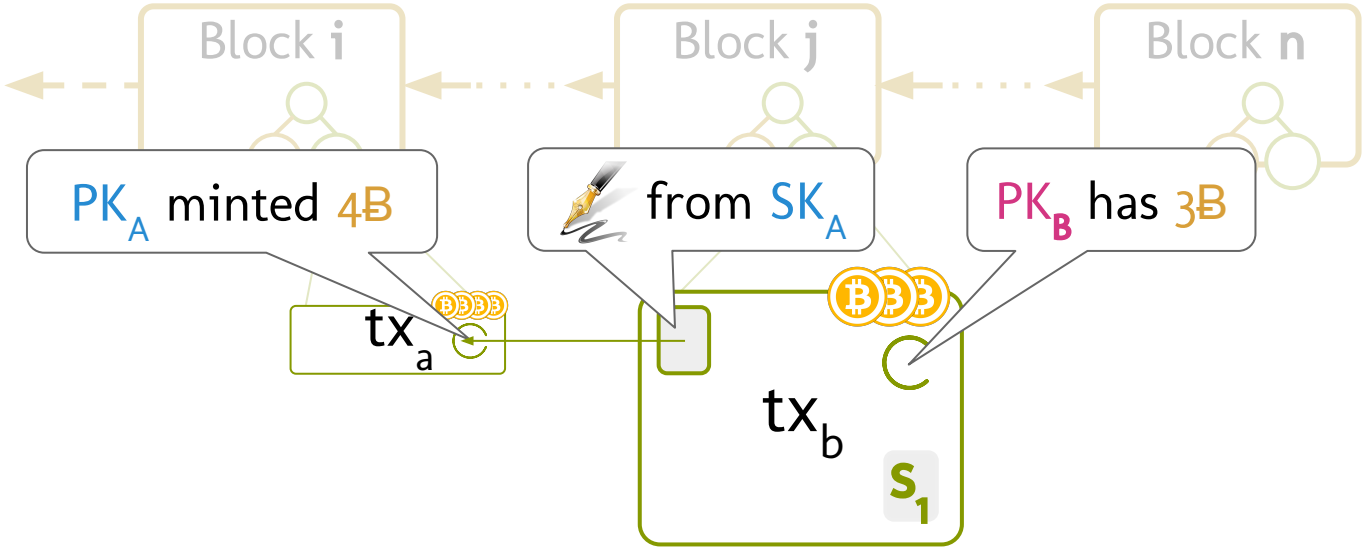
- Transactions mint coins
- Output = # of coins and owner's PK
- Transactions transfer coins (and pay fees)
- Input = hash pointer to output & digital signature

Bitcoin blockchain



Data can be embedded in TXNs.

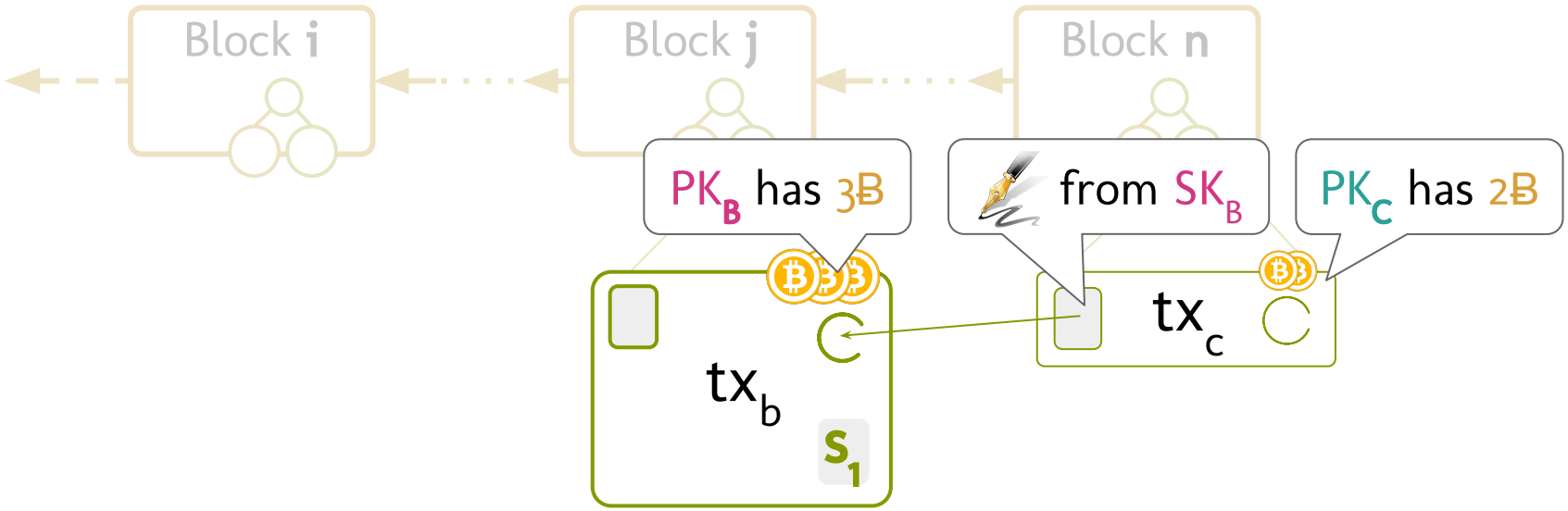
Bitcoin blockchain



Alice gives Bob 3Ƀ,

Bitcoin *miners* collected 1Ƀ as a *fee.*

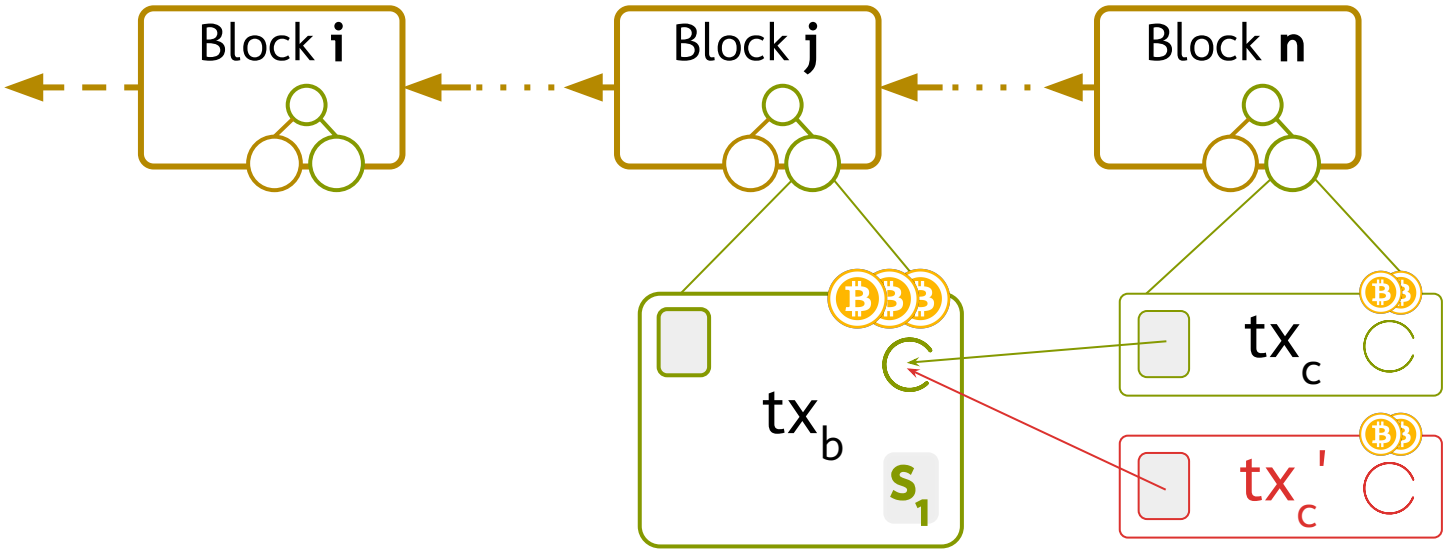
Bitcoin blockchain



Bob gives Carol 2Ƀ,

Bitcoin *miners* collected another Ƀ as a fee.

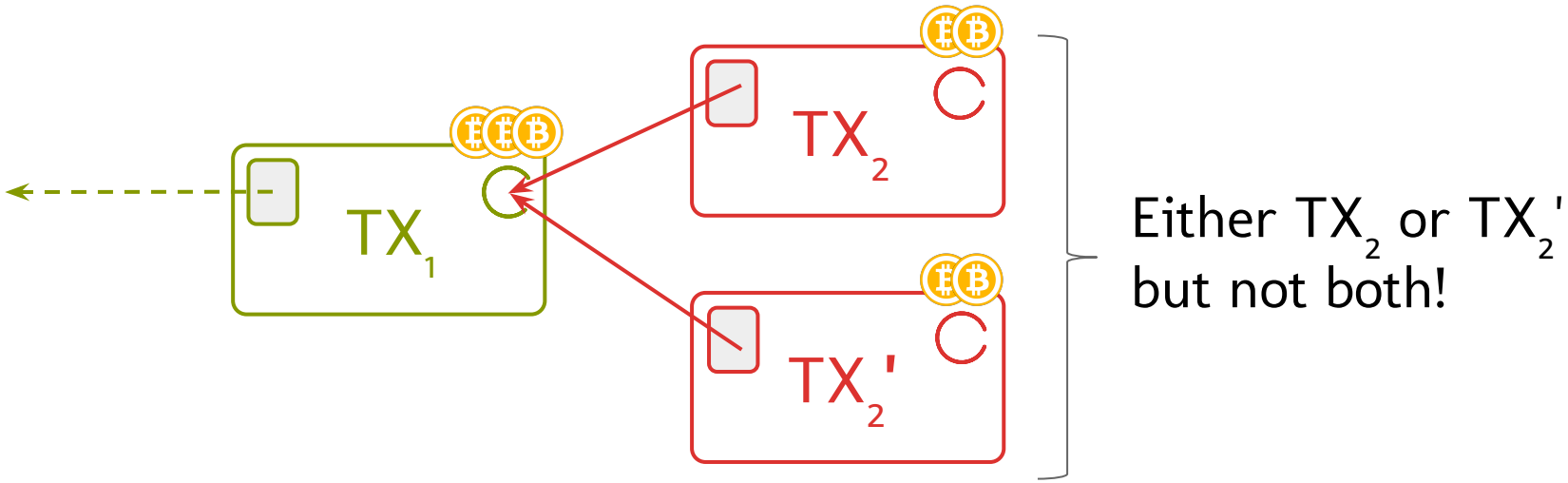
Bitcoin blockchain



No double-spent coins: A TXN output can only be referred to by a single TXN input.

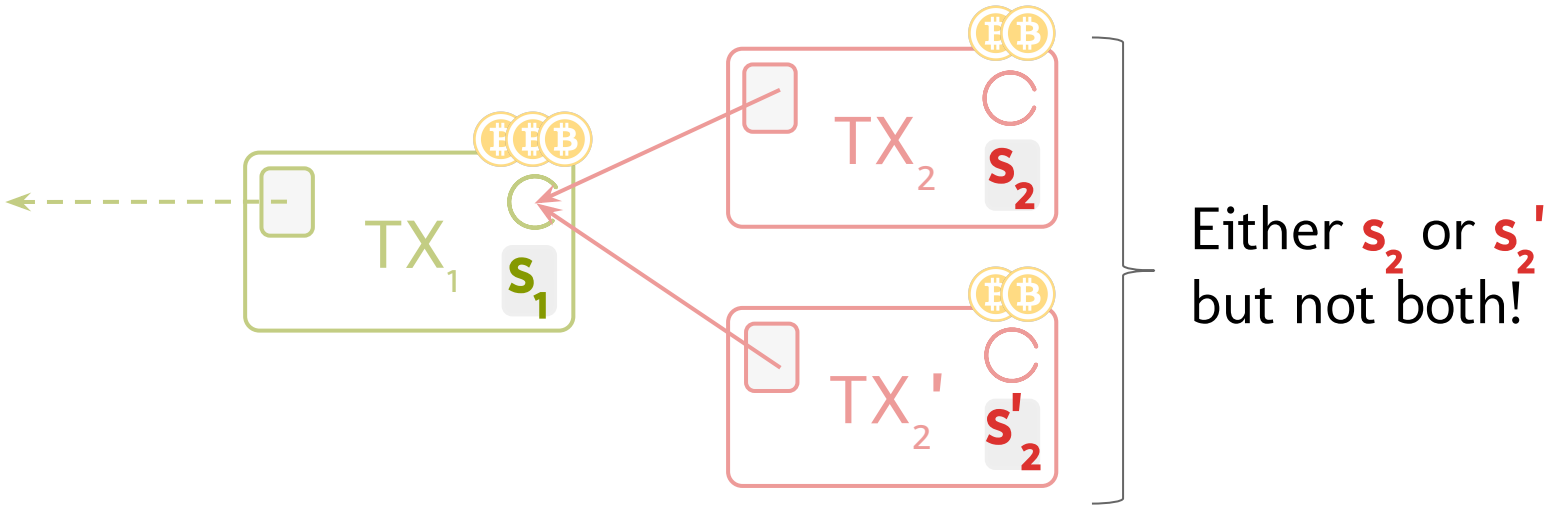
Moral of the story

Proof-of-work (PoW) consensus ⇒ No double spends



Moral of the story

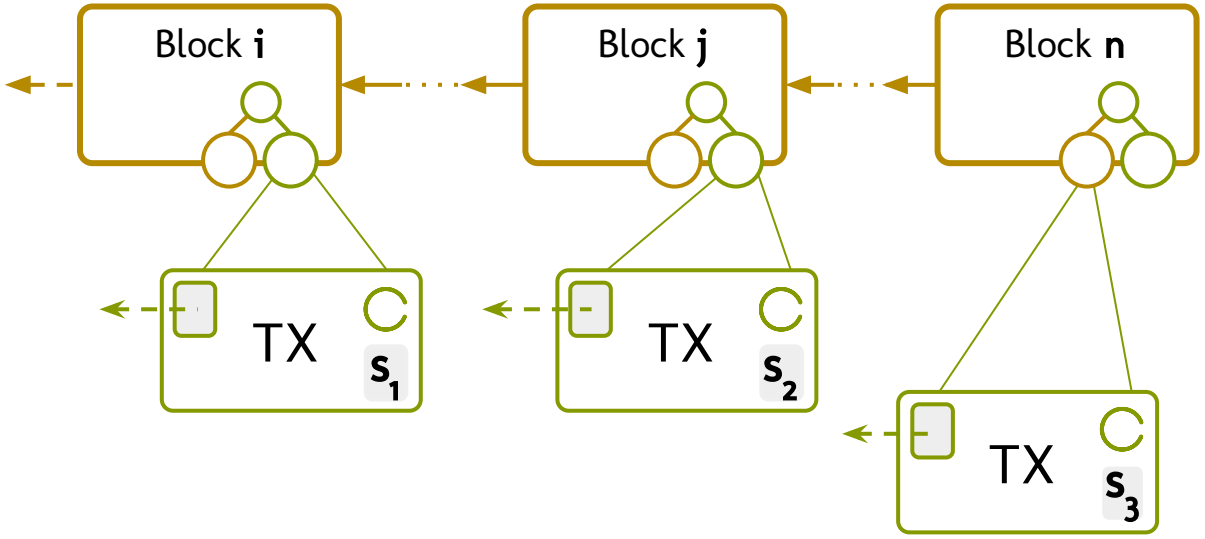
Proof-of-work (PoW) consensus ⇒ No double spends



Outline

1. Bitcoin background
2. **Previous work**
3. Catena design
4. Catena scalability

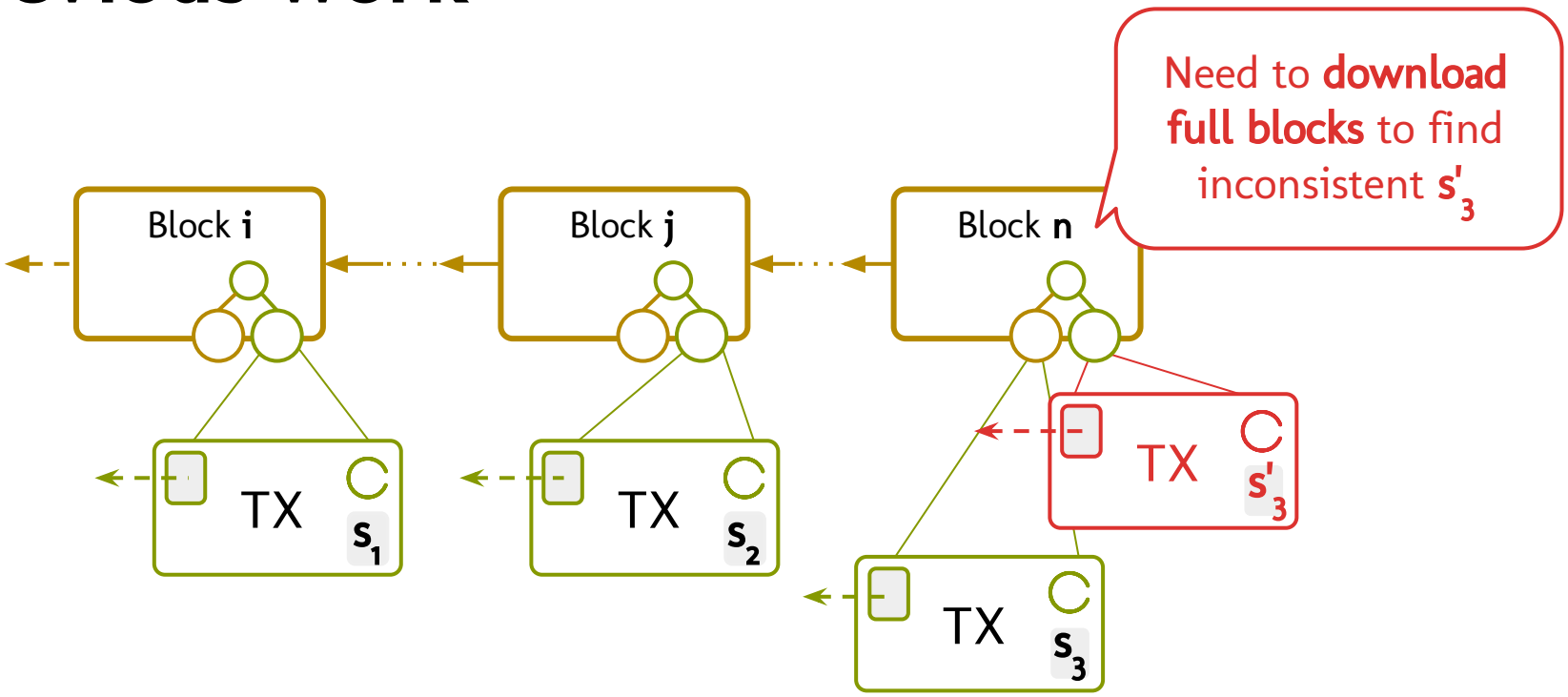
Previous work



blockstack

Keybase

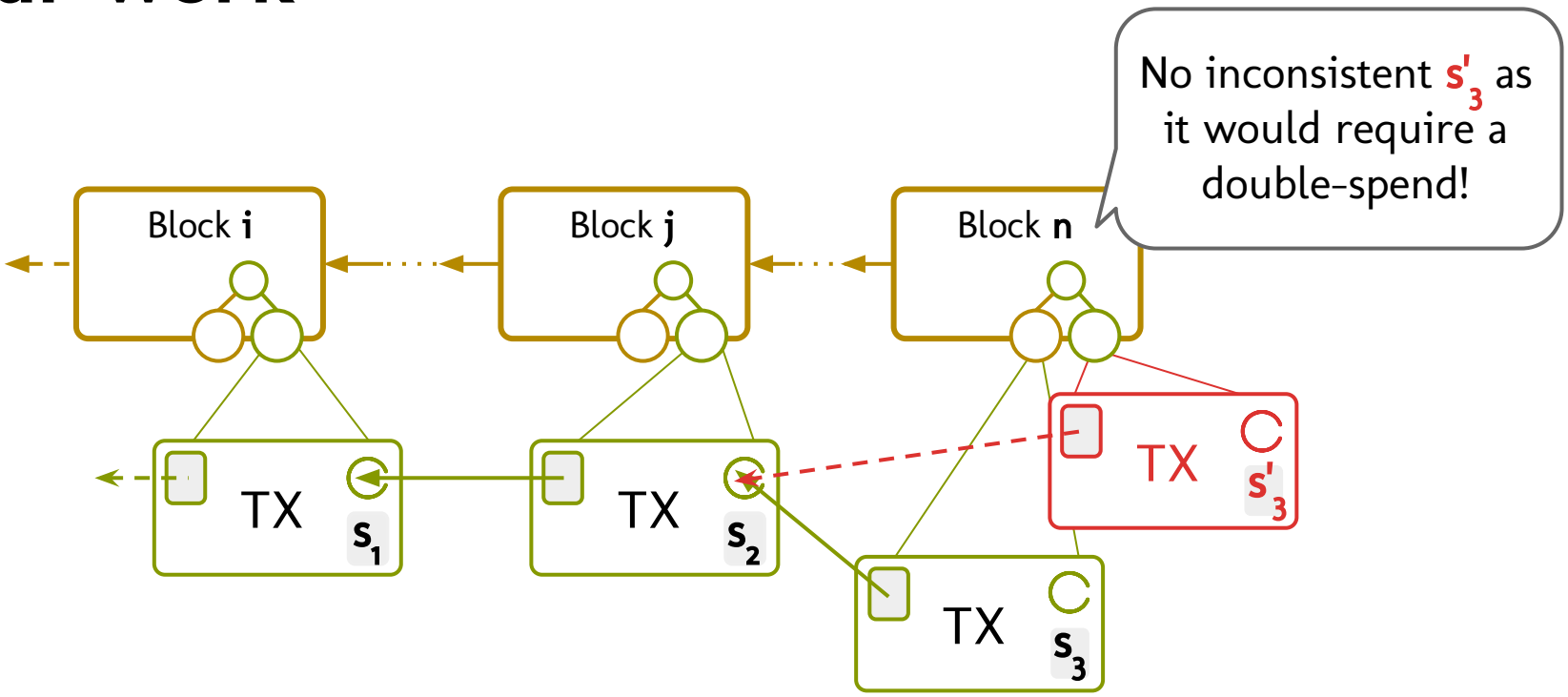
Previous work



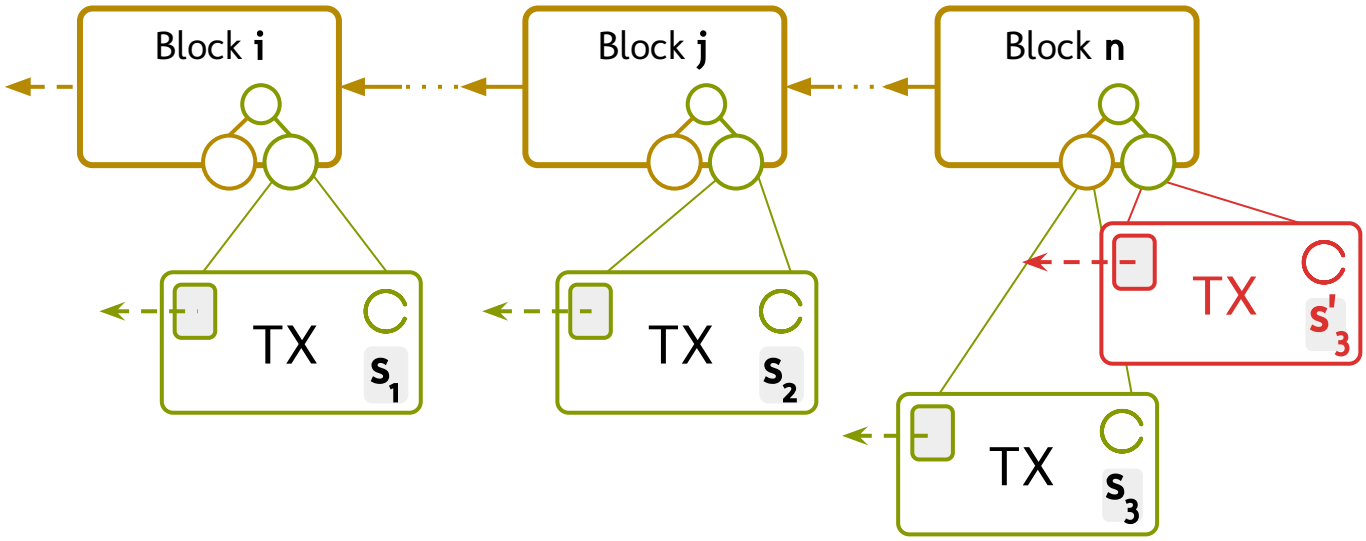
blockstack

Keybase

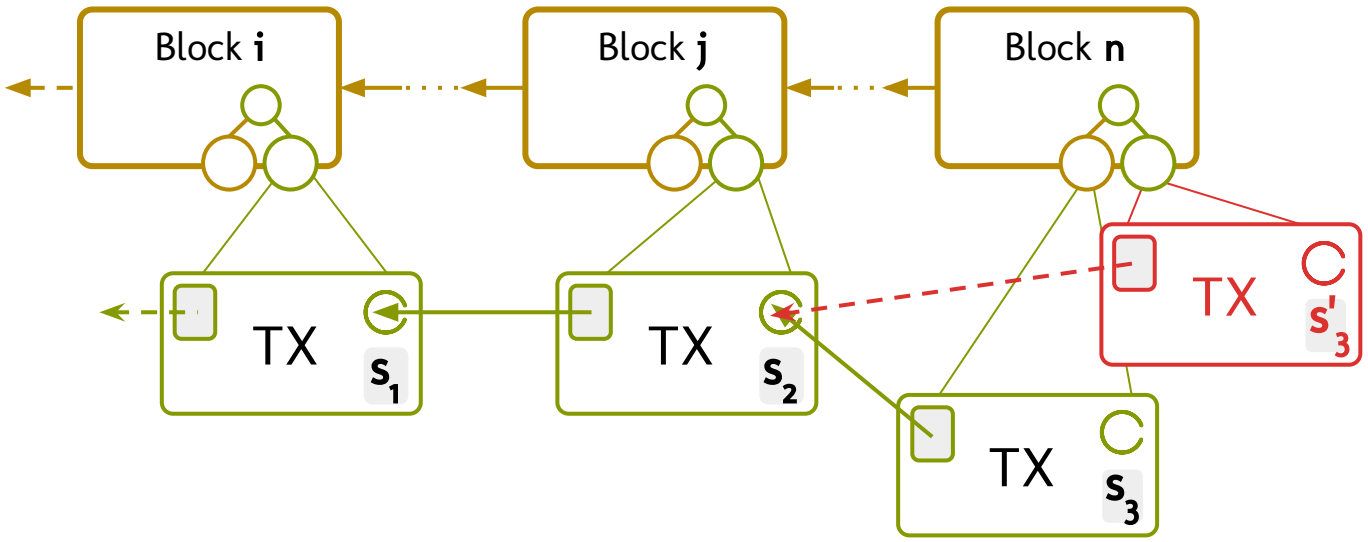
Our work


No inconsistent s'_3 as it would require a double-spend!
Block i
Block j
Block n
TX
s_1
TX
s_2
TX
s_3
TX
s'_3

Previous work



Our work



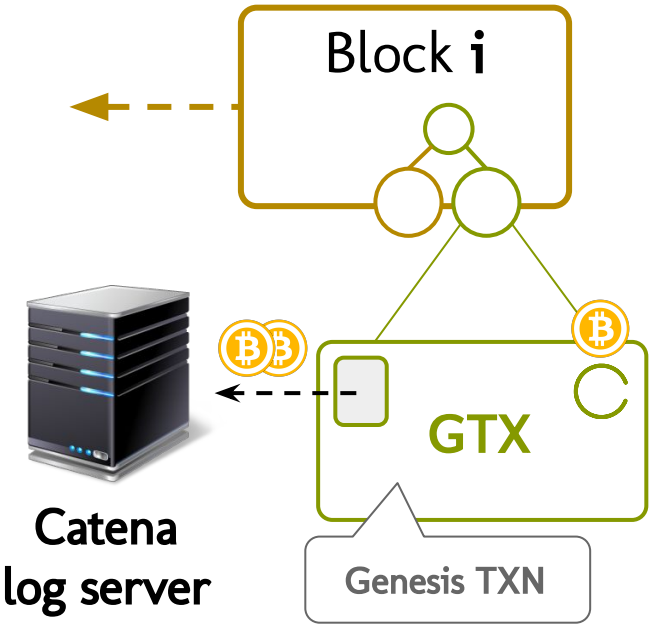
Outline

1. Bitcoin background
2. Previous work
3. **Catena design**
4. Catena scalability

Starting a Catena log

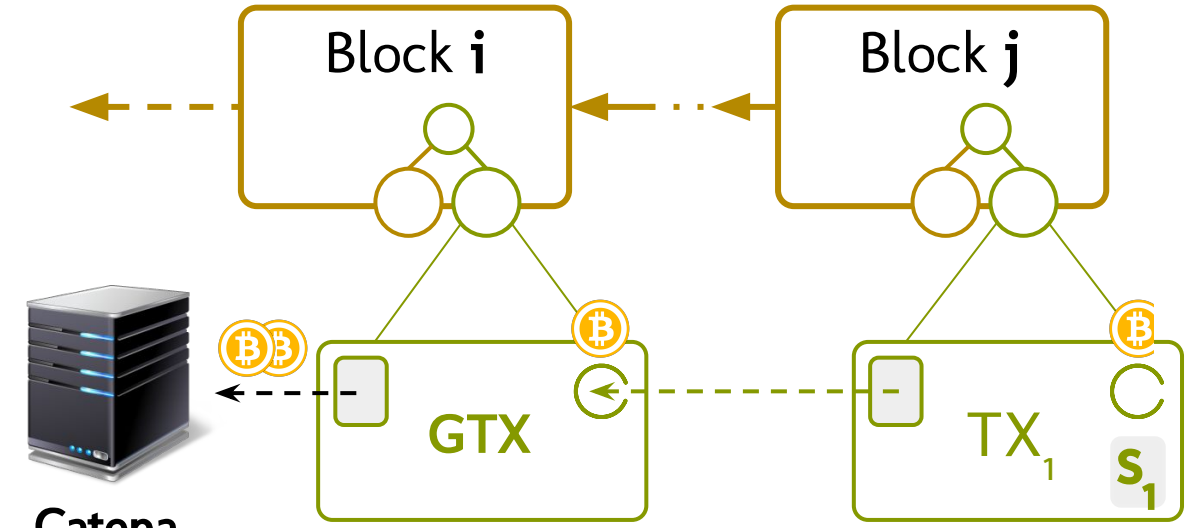
Catena
log server

Starting a Catena log



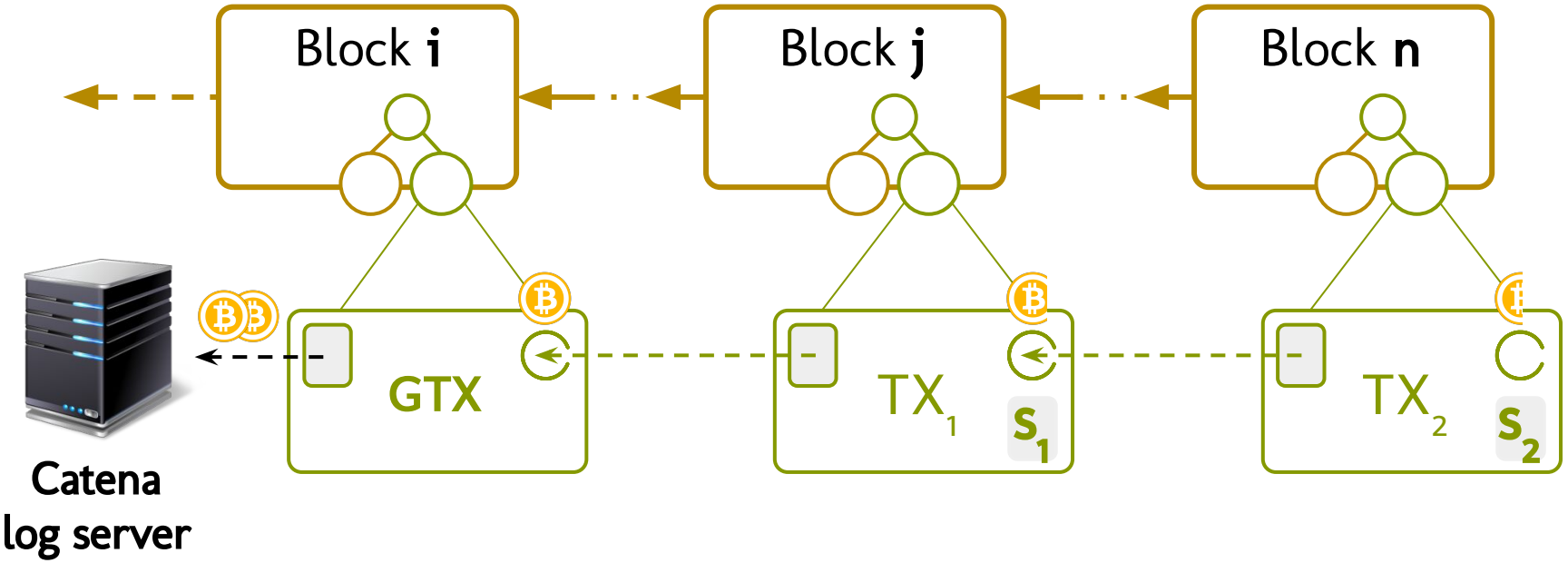
- *Genesis TXN* (GTX) = log's "public key"
- Coins from server back to server (minus fees)

Appending to a Catena log



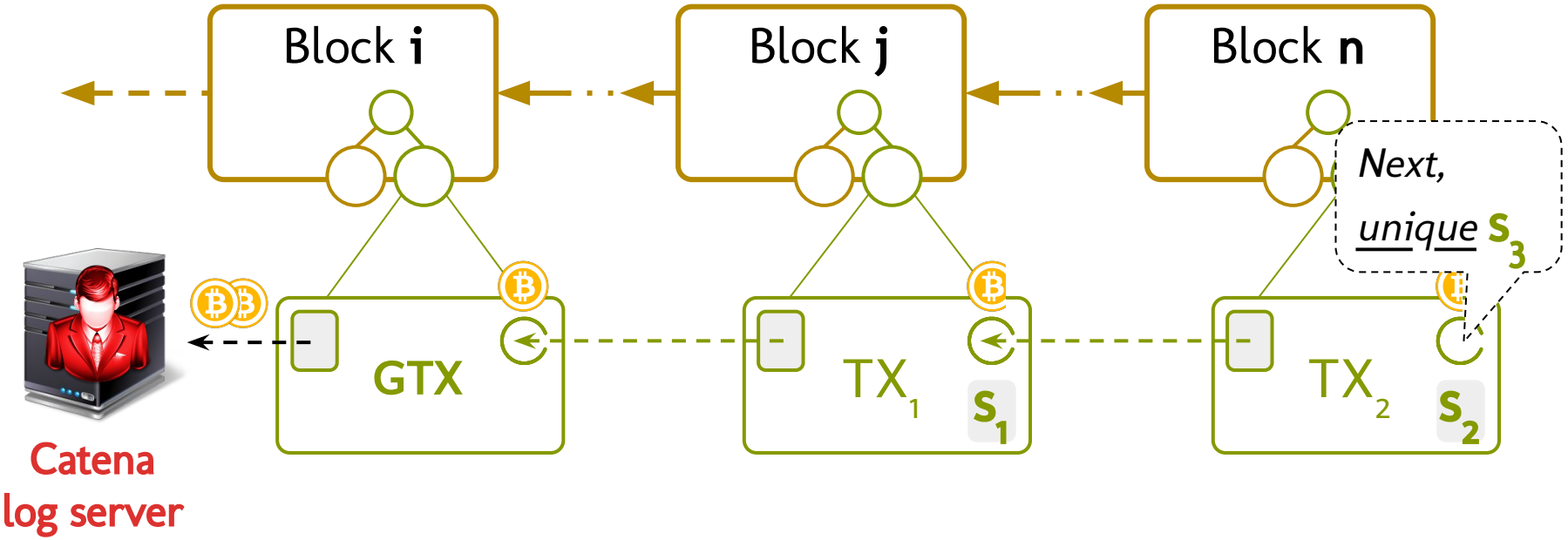
- TX_1 "spends" GTX's output, publishes s_1
- Coins from server back to server (minus fees)
- Inconsistent s_1' would require a double-spend

Appending to a Catena log



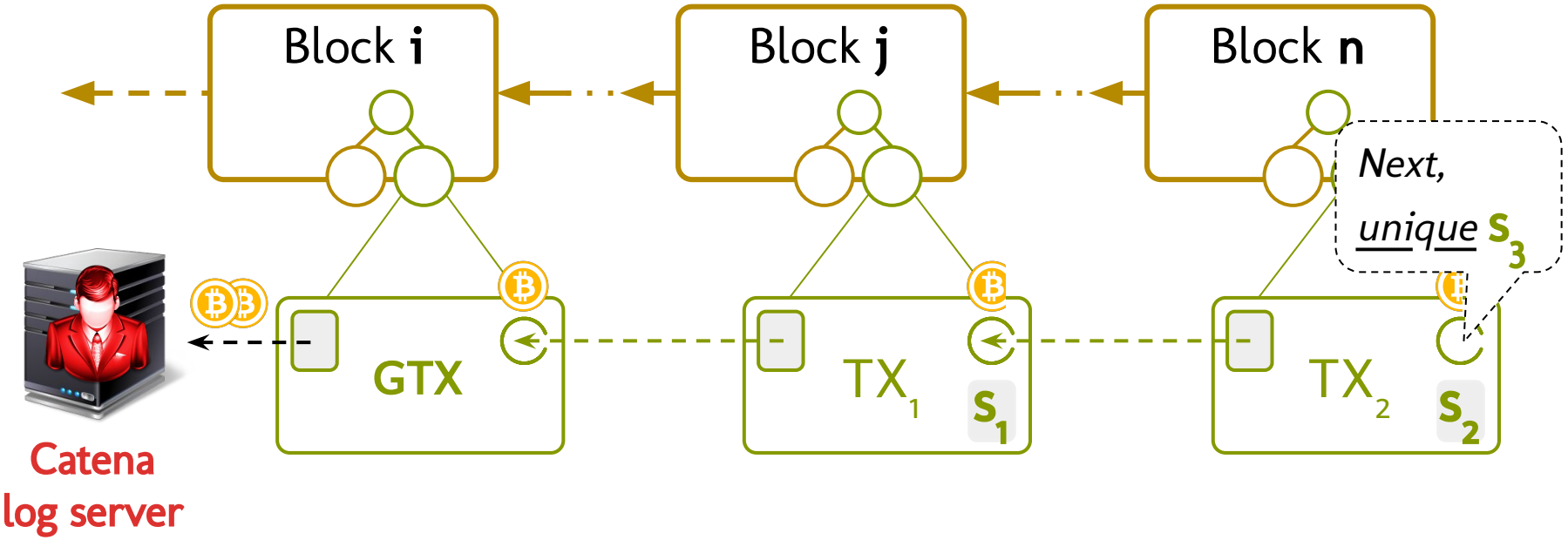
- TX_2 "spends" TX_1's output, publishes s_2
- Coins from server back to server (minus fees)
- Inconsistent s_2' would require a double-spend

Appending to a Catena log



- Server is compromised, still cannot equivocate.

Appending to a Catena log

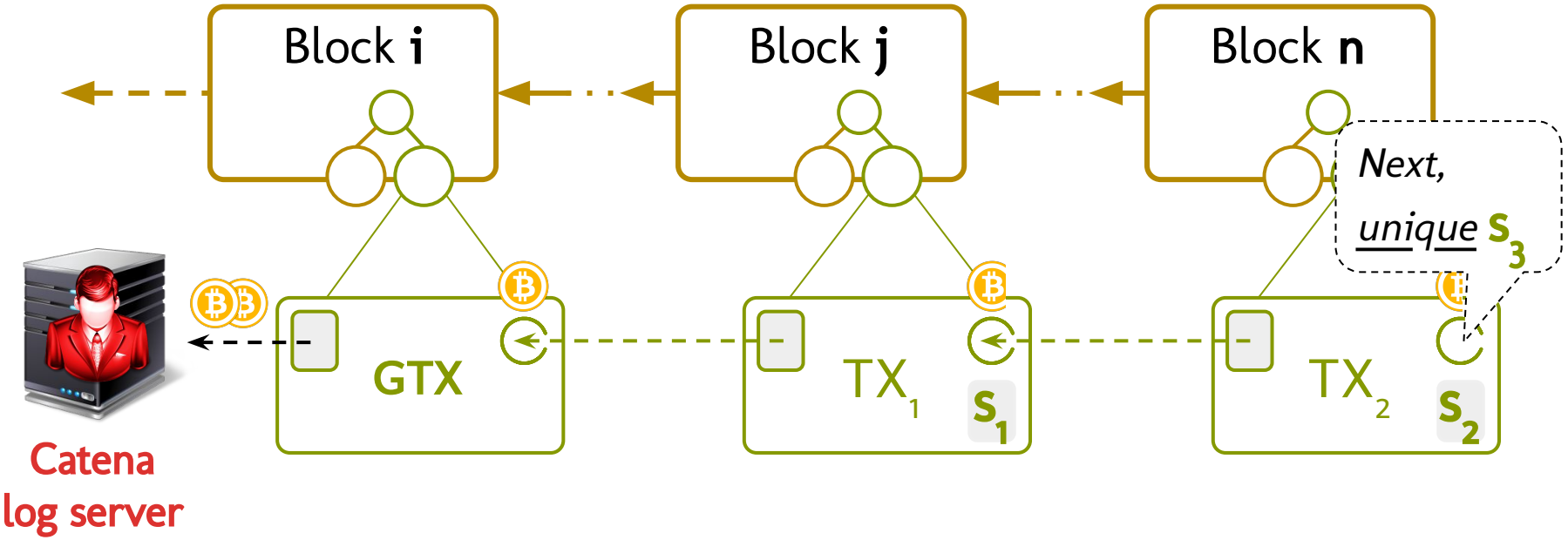


Advantages:

(1) Hard to fork

(2) Efficient to verify

Appending to a Catena log



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

Disadvantages:
(1) 6-block confirmation delay
(2) 1 statement every 10 minutes
(3) Must pay Bitcoin TXN fees

Efficient auditing

Efficient auditing

Catena
client

Catena
log server

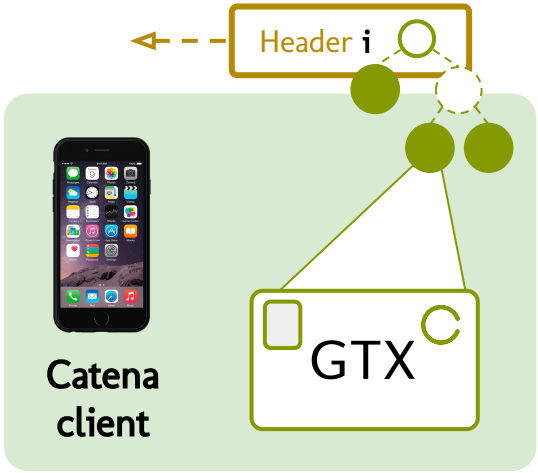
Efficient auditing

Catena
client

Bitcoin P2P
(7000 nodes)

Catena
log server

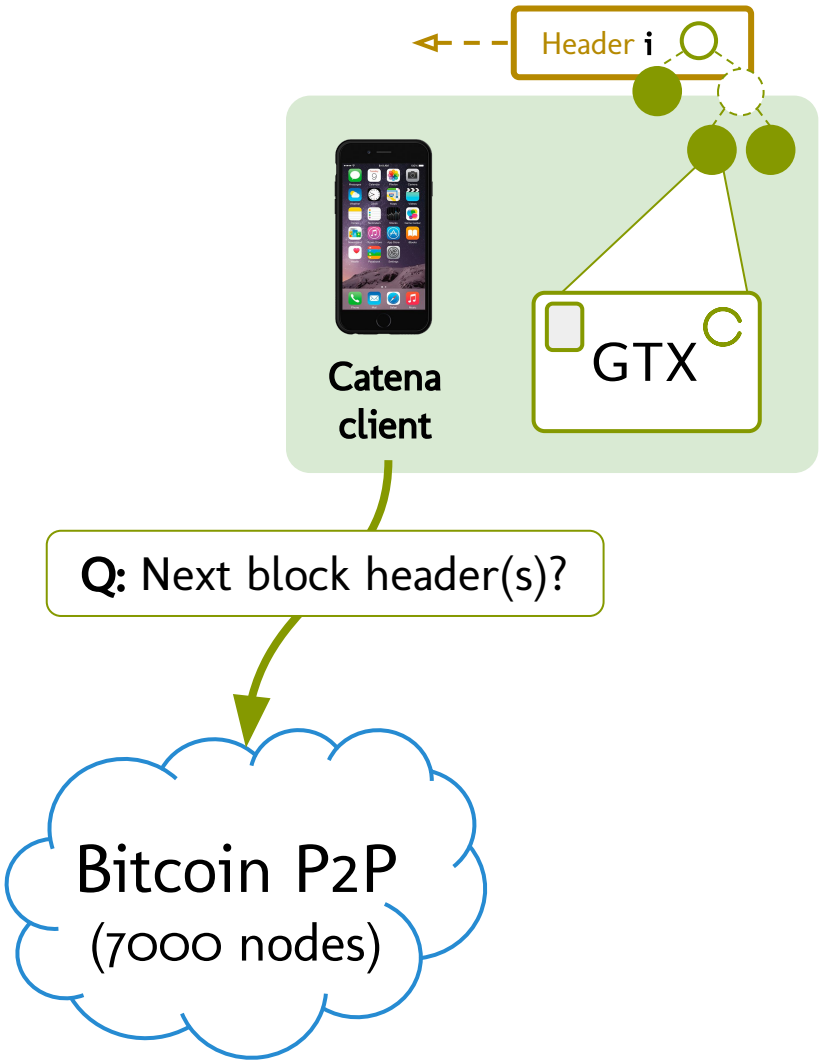
Efficient auditing



Bitcoin P2P
(7000 nodes)

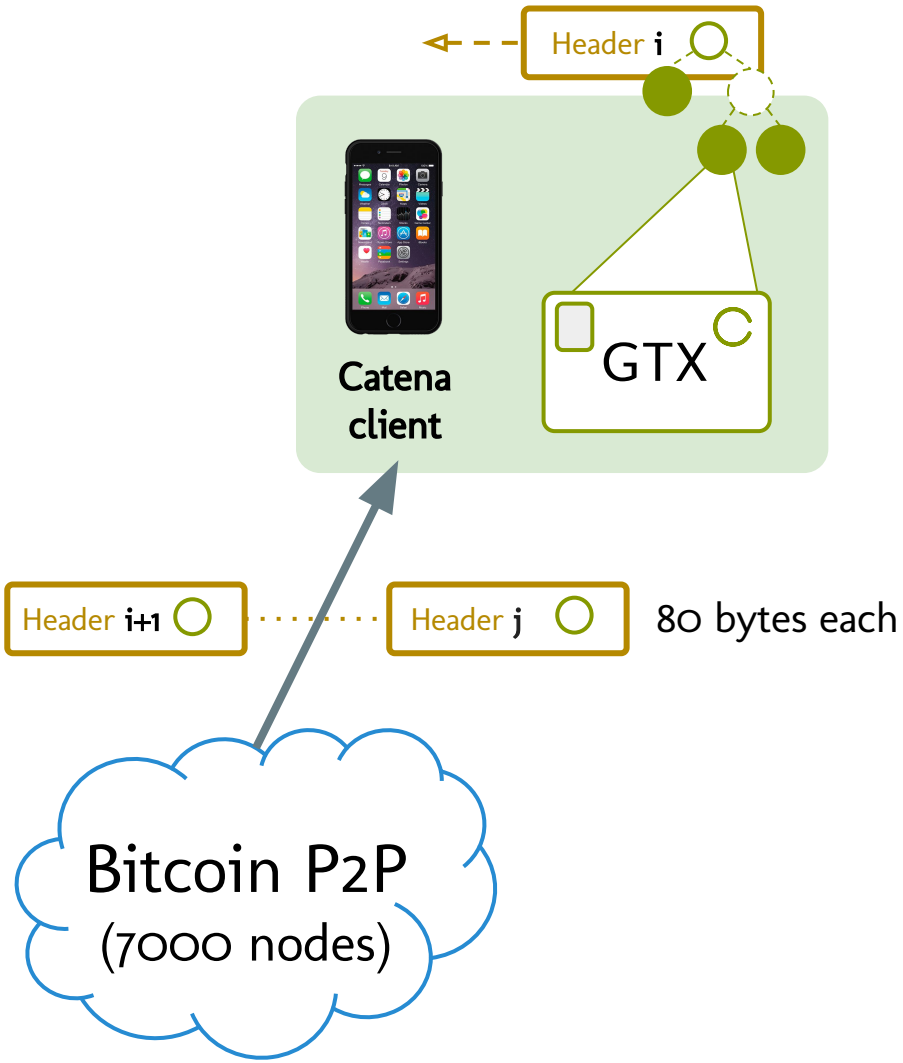
Catena
log server

Efficient auditing



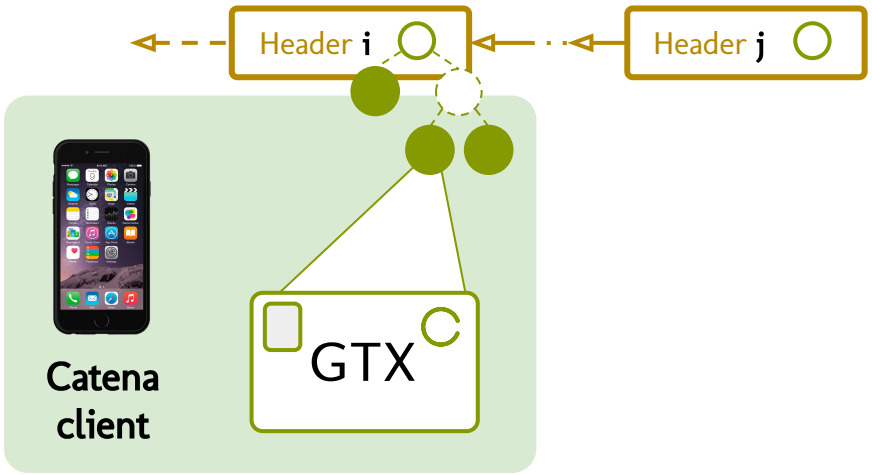
Catena log server

Efficient auditing



Catena log server

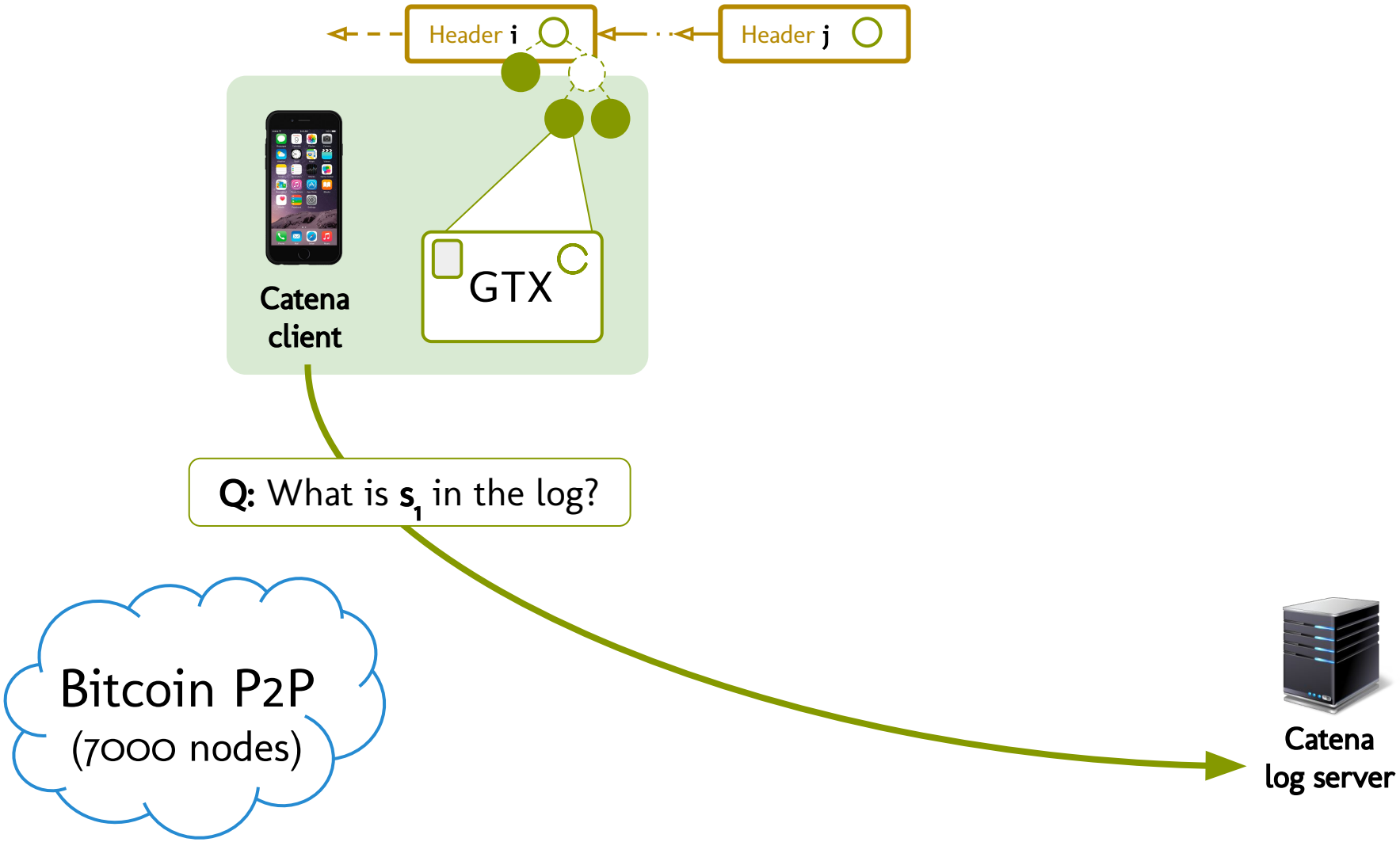
Efficient auditing



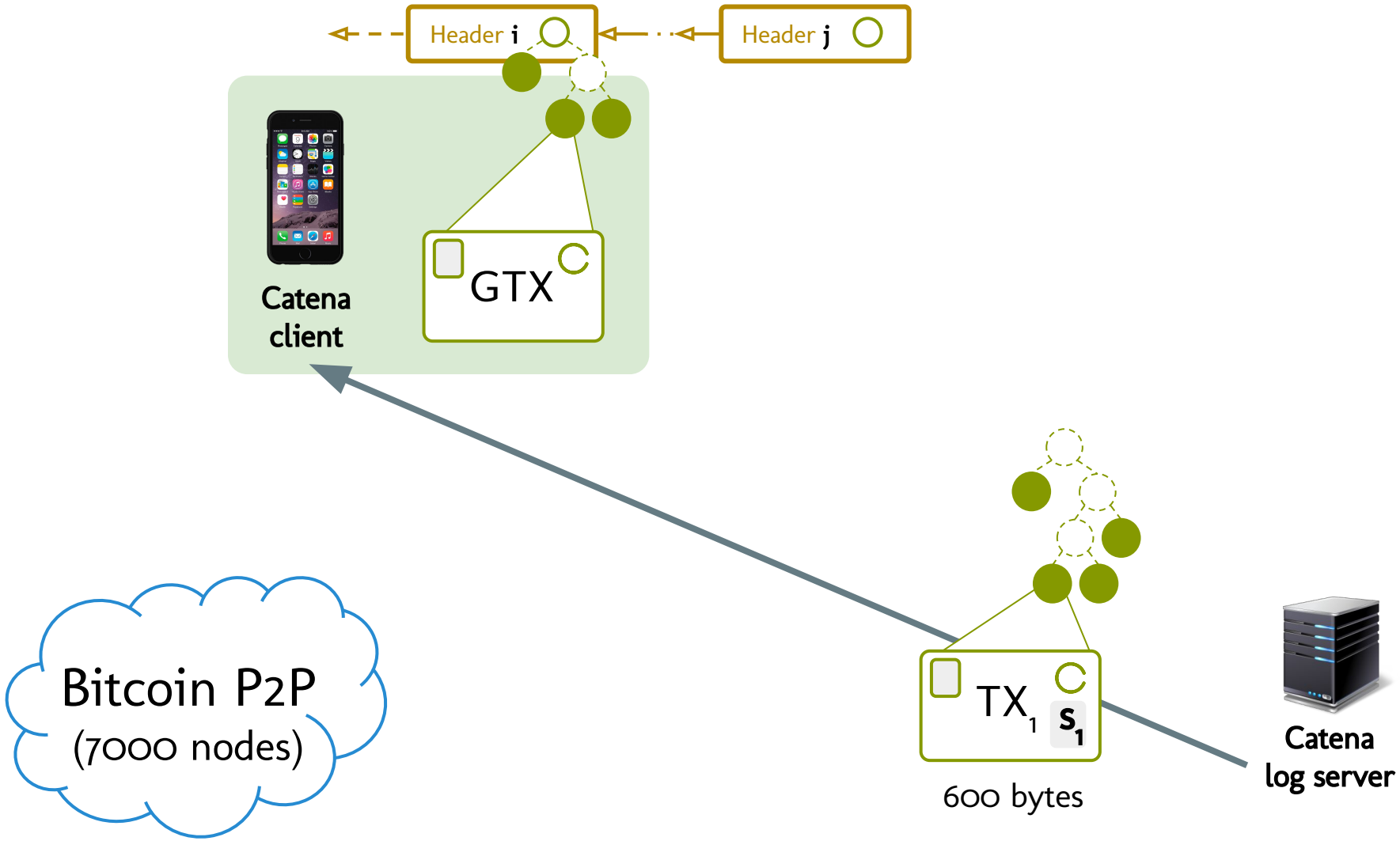
Bitcoin P2P
(7000 nodes)

Catena log server

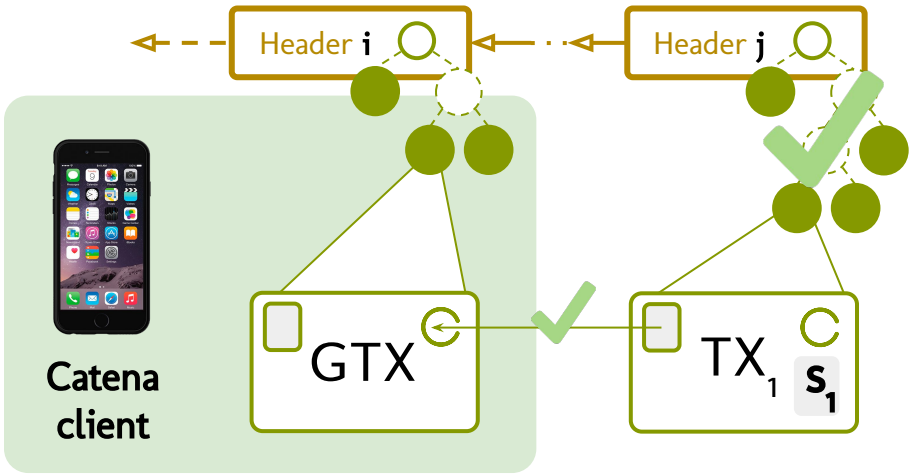
Efficient auditing



Efficient auditing



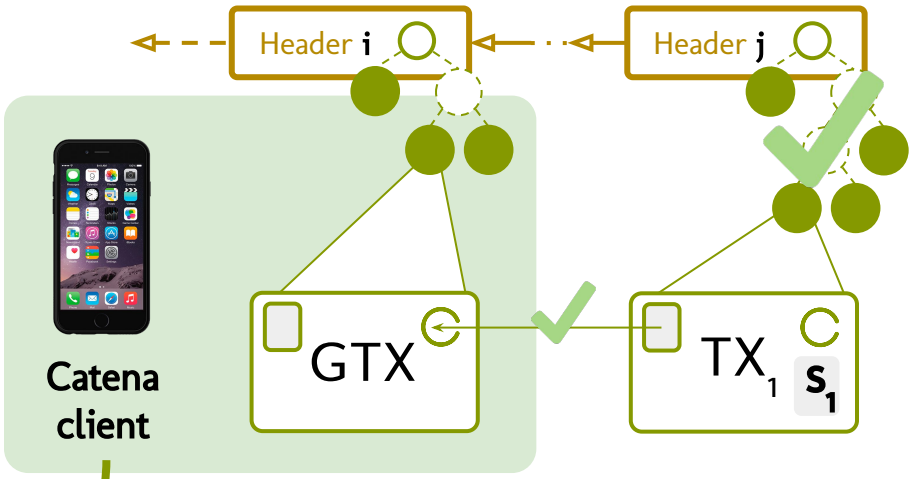
Efficient auditing



Bitcoin P2P
(7000 nodes)

Catena log server

Efficient auditing

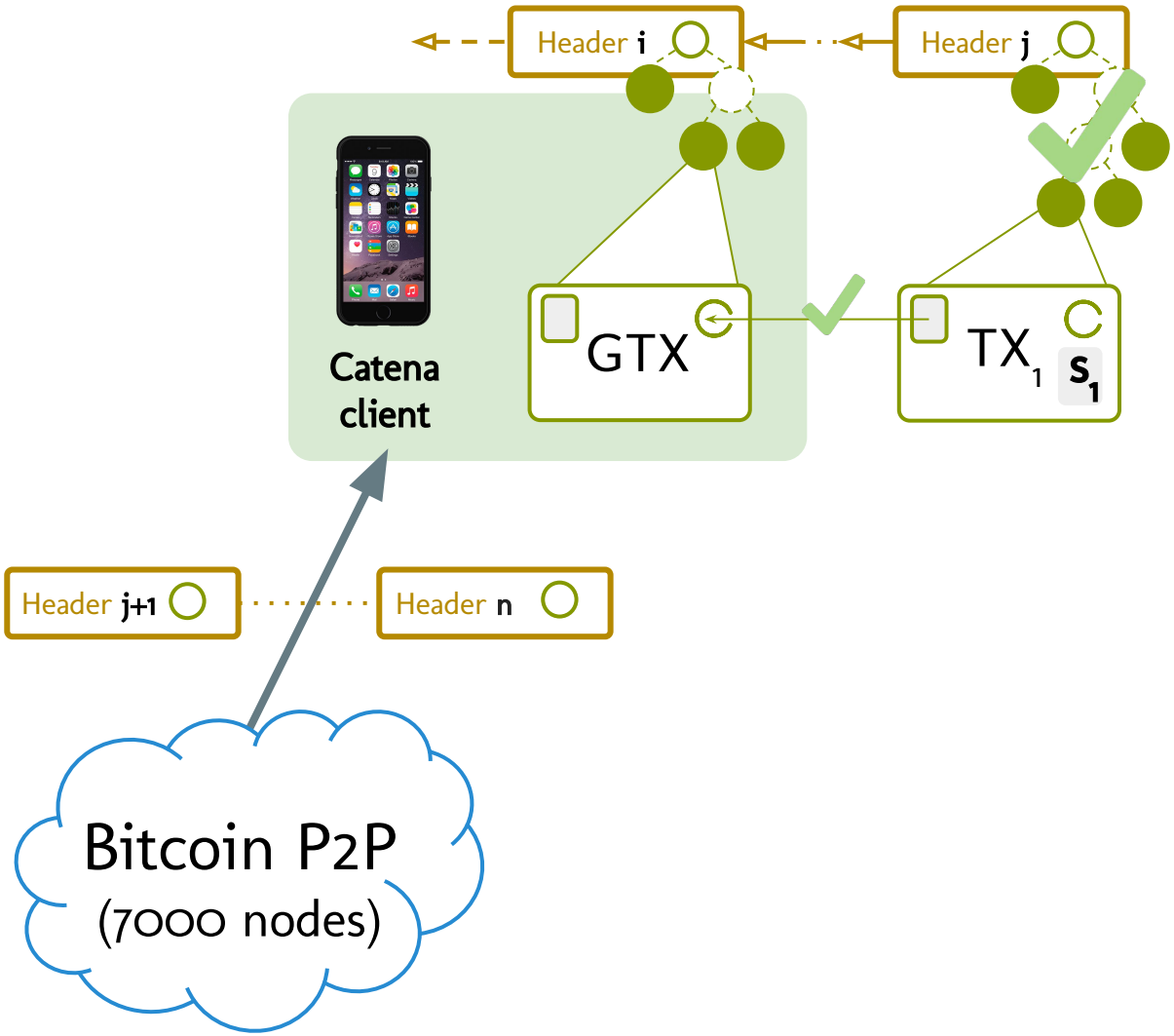


Q: Next block header(s)?

Bitcoin P2P
(7000 nodes)

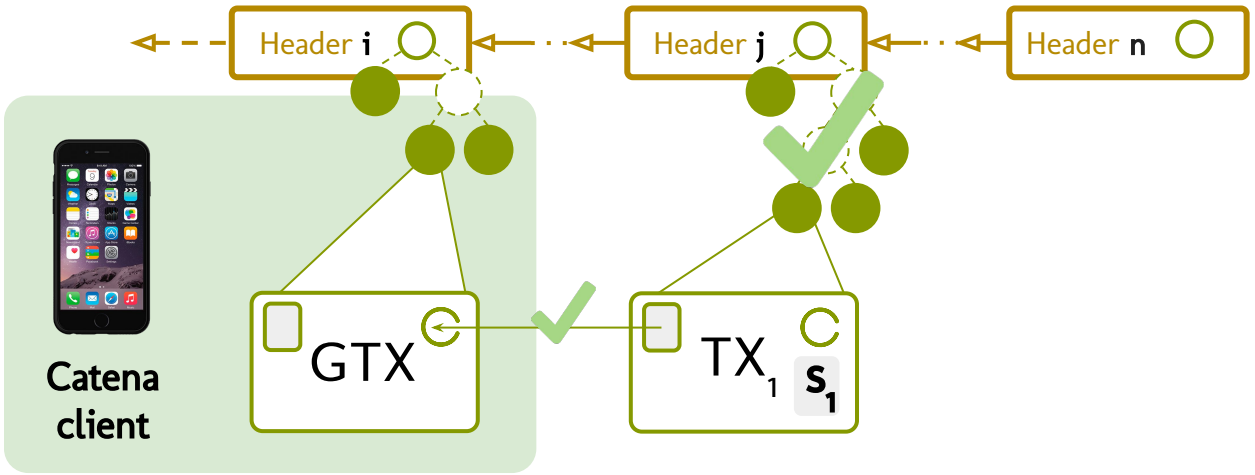
Catena log server

Efficient auditing



Catena log server

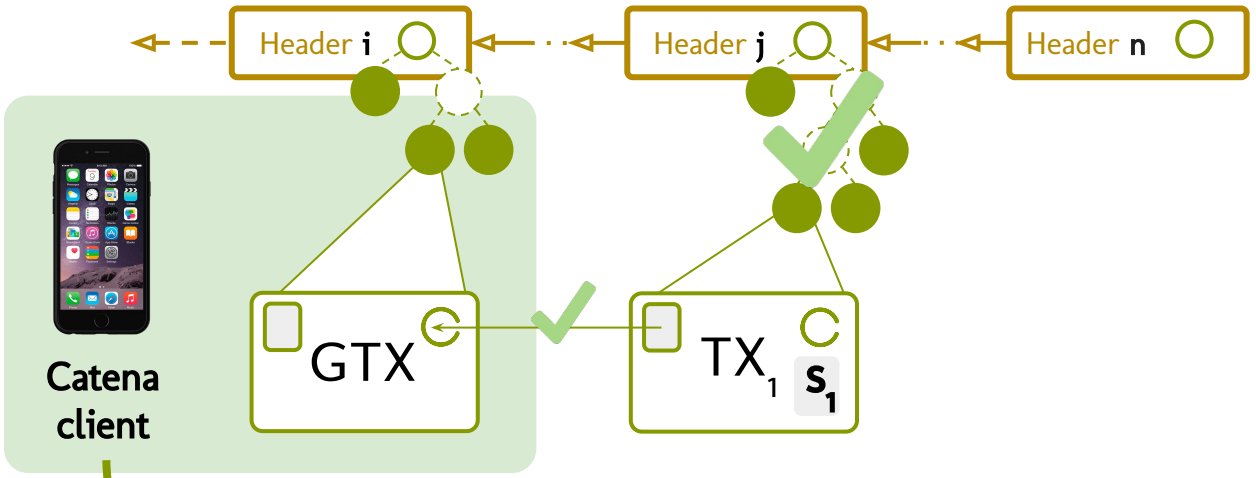
Efficient auditing



Bitcoin P2P
(7000 nodes)

Catena log server

Efficient auditing

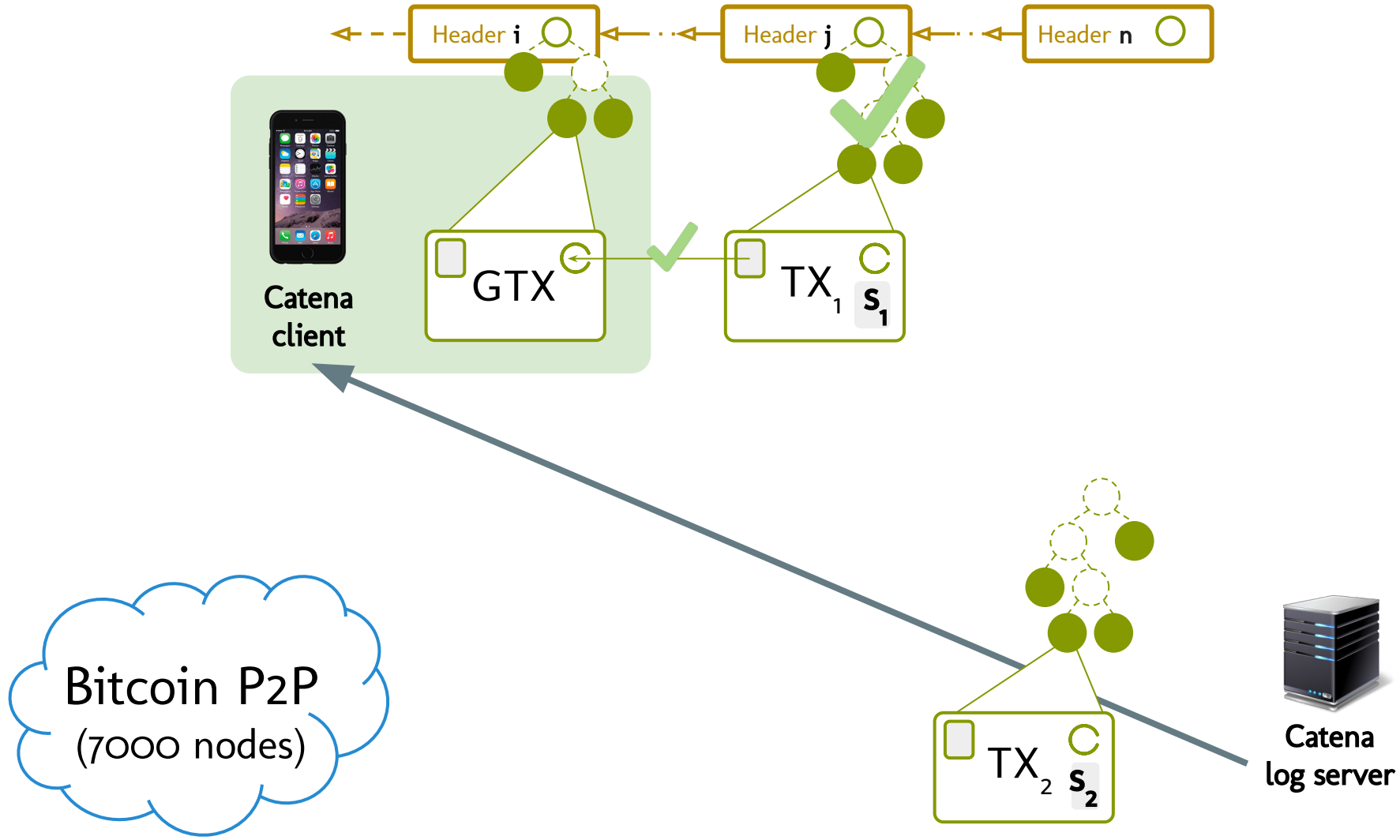


Q: What is s_2 in the log?

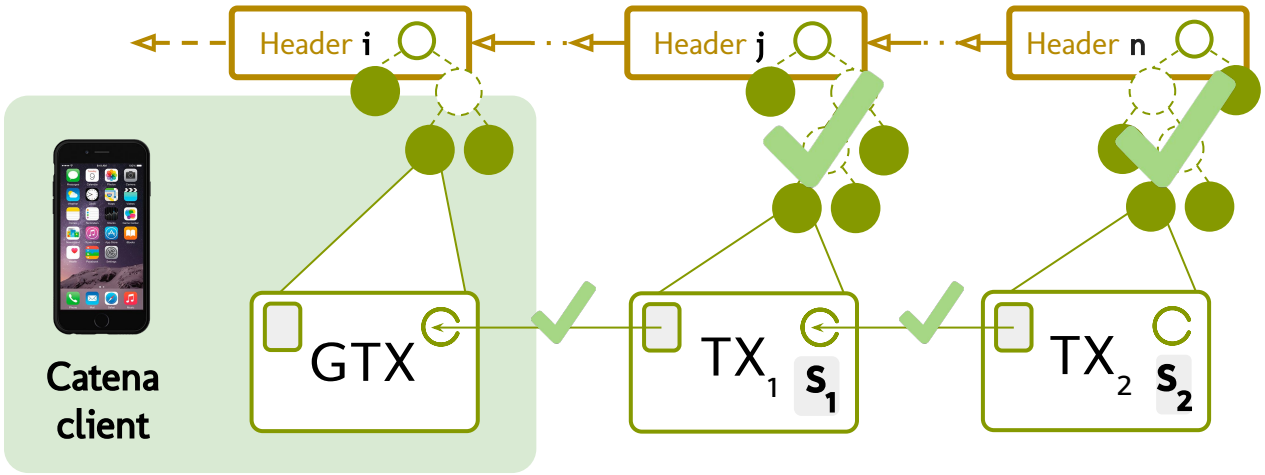
Bitcoin P2P
(7000 nodes)

Catena log server

Efficient auditing



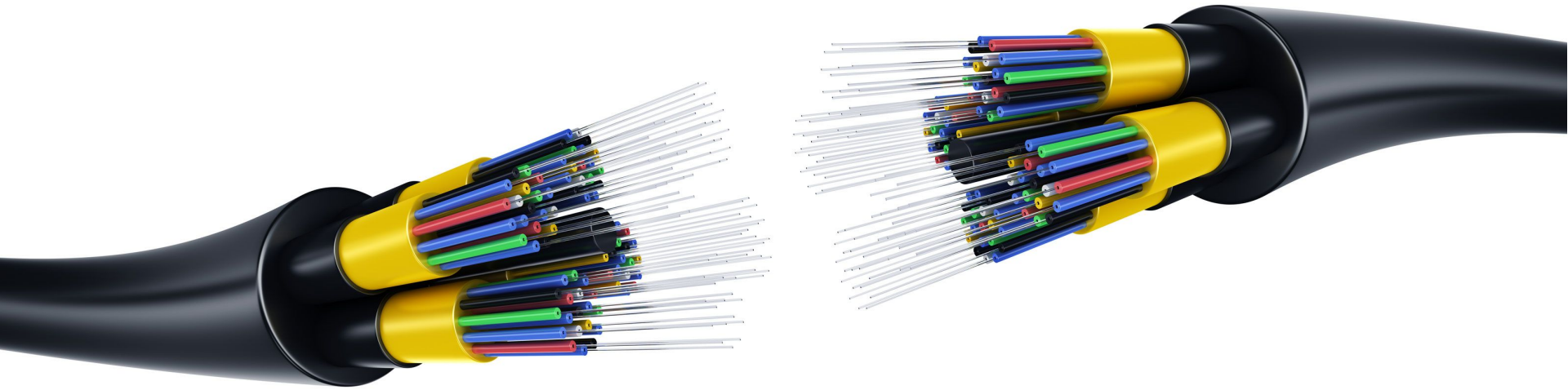
Efficient auditing



Bitcoin P2P
(7000 nodes)

Catena log server

Auditing bandwidth



e.g., **460K** block headers + **10K** statements = **~41** MB

(80 bytes each) (around 600 bytes each)

Outline

1. Bitcoin background
2. Previous work
3. Catena design
4. **Catena scalability**

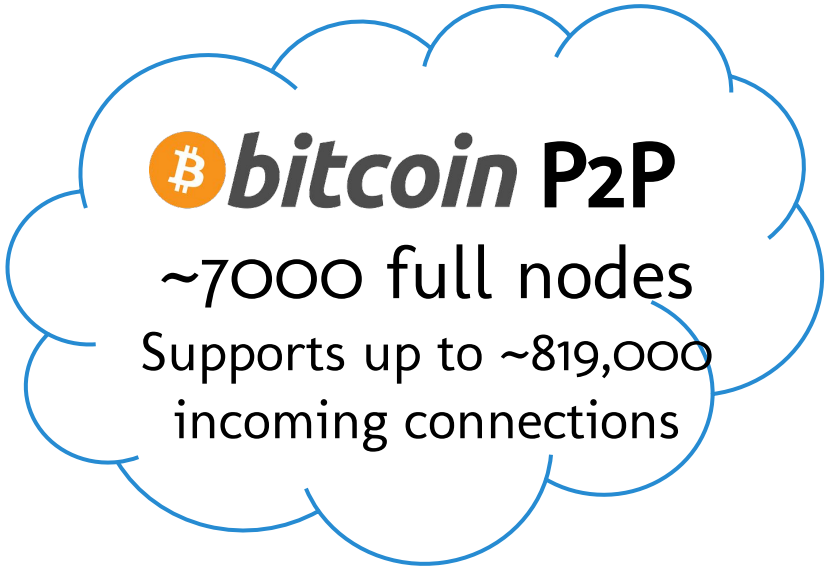
Catena scalability

Catena client 1

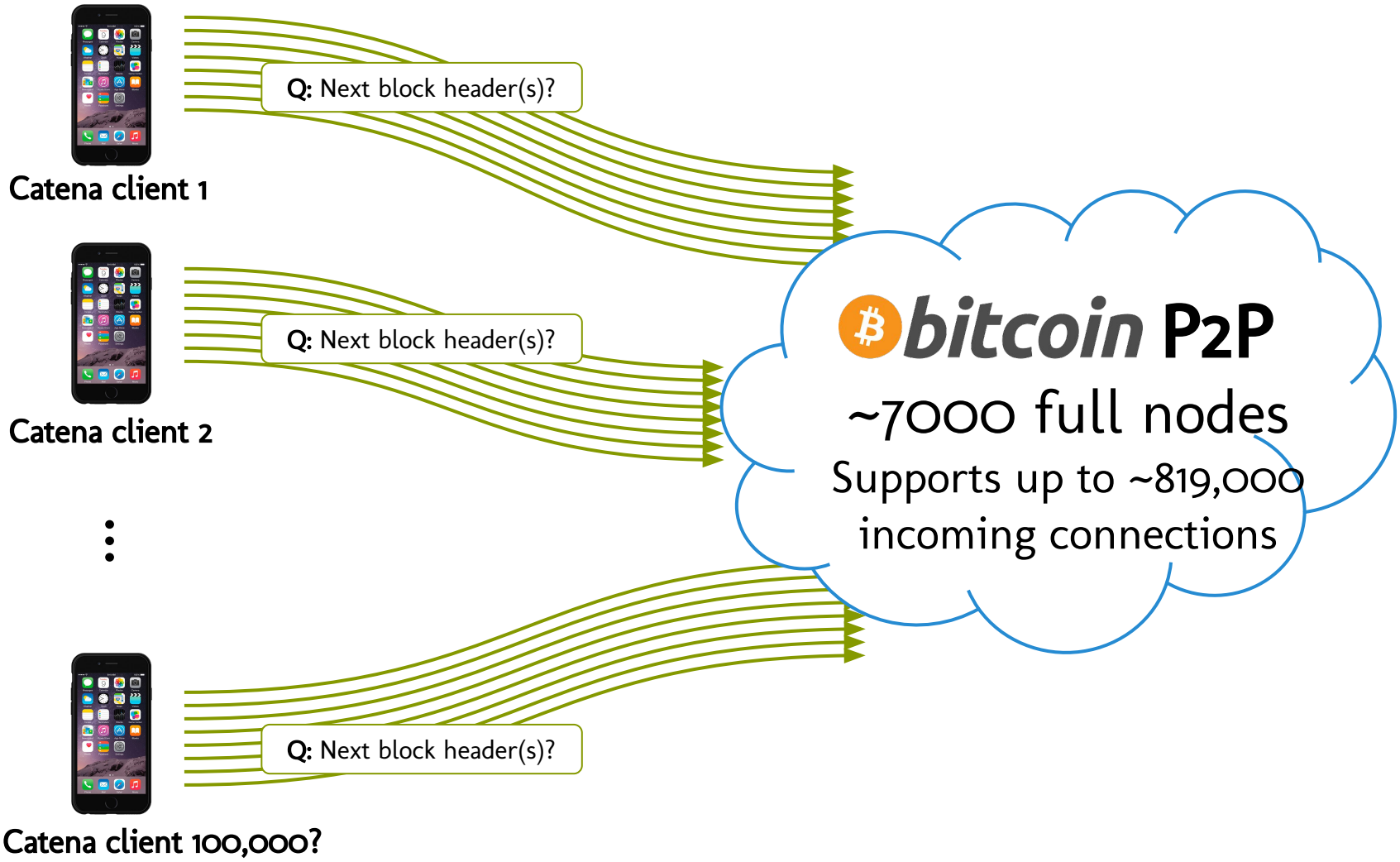
Catena client 2

⋮

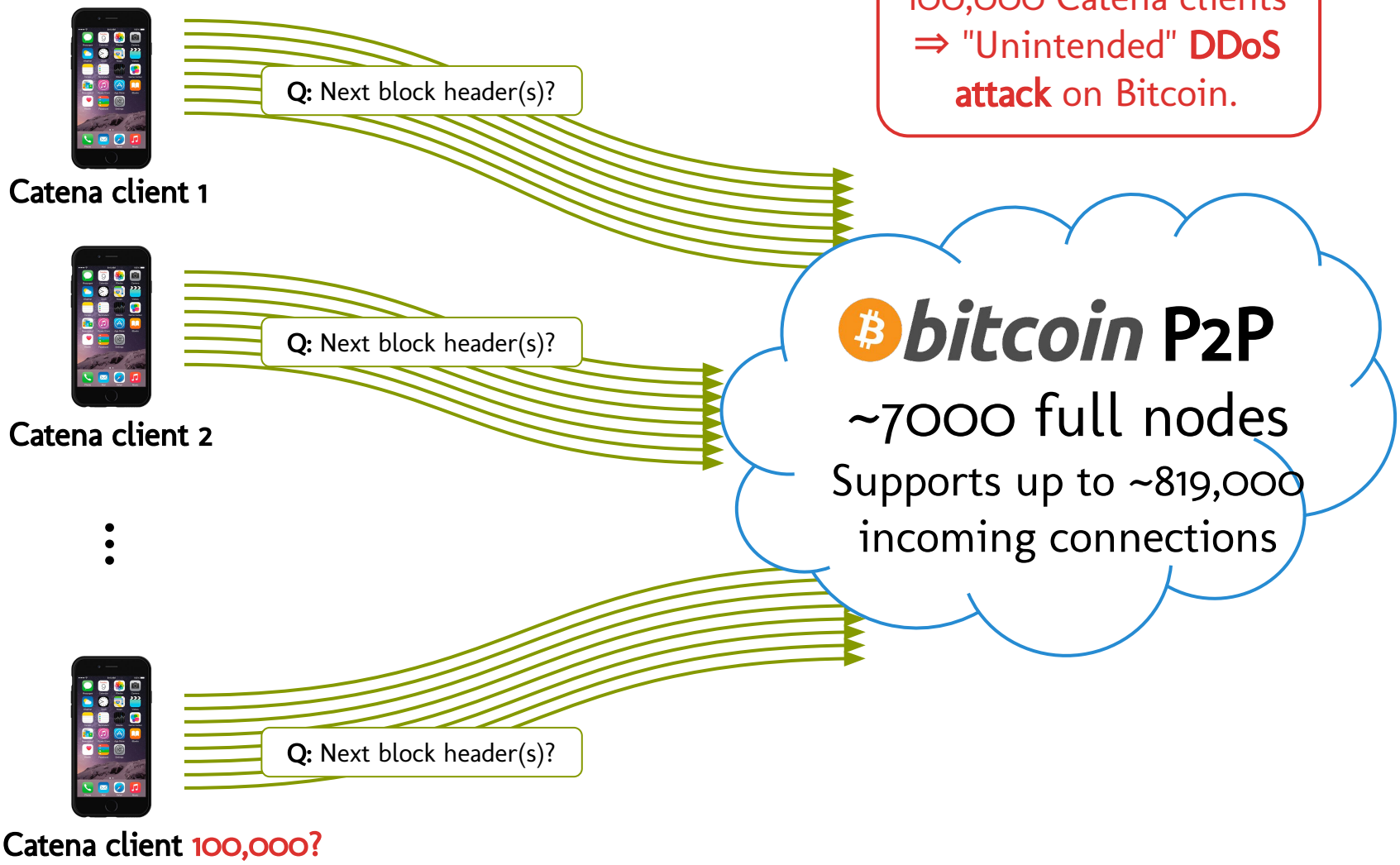
Catena client 100,000?



Catena scalability



Catena scalability



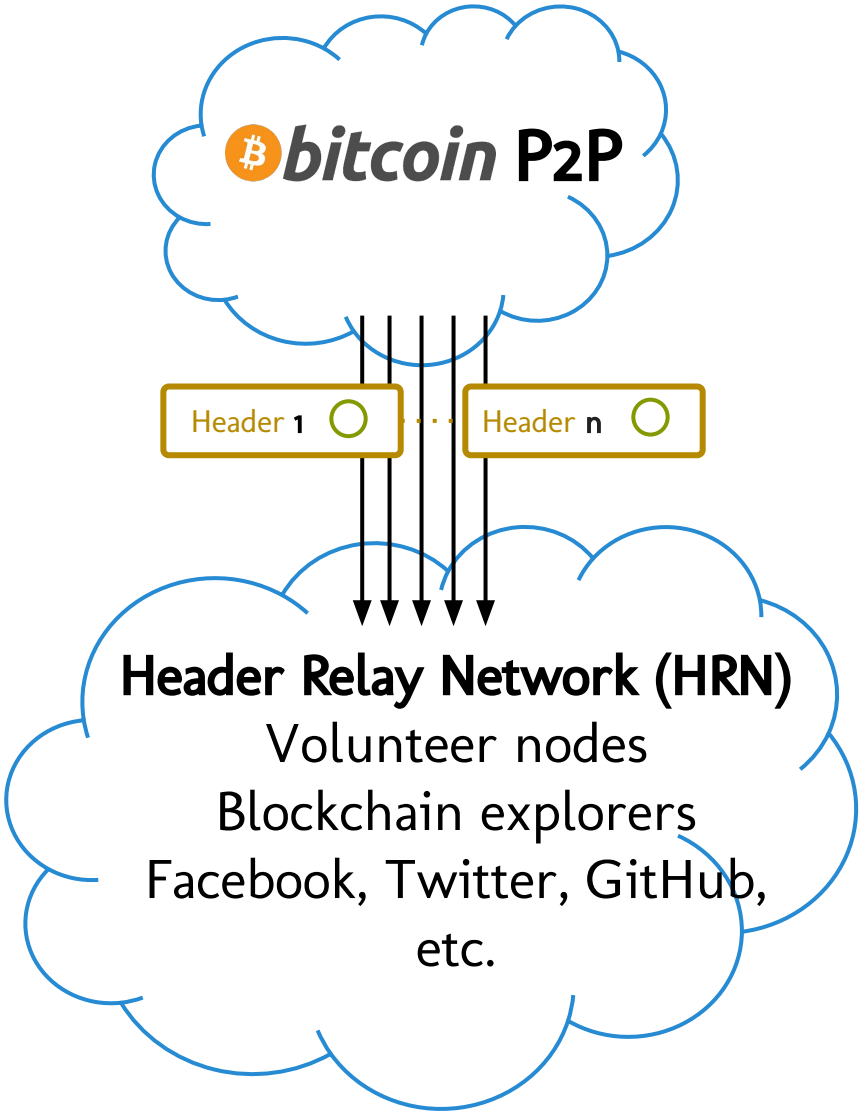
Catena scalability

Catena client 1

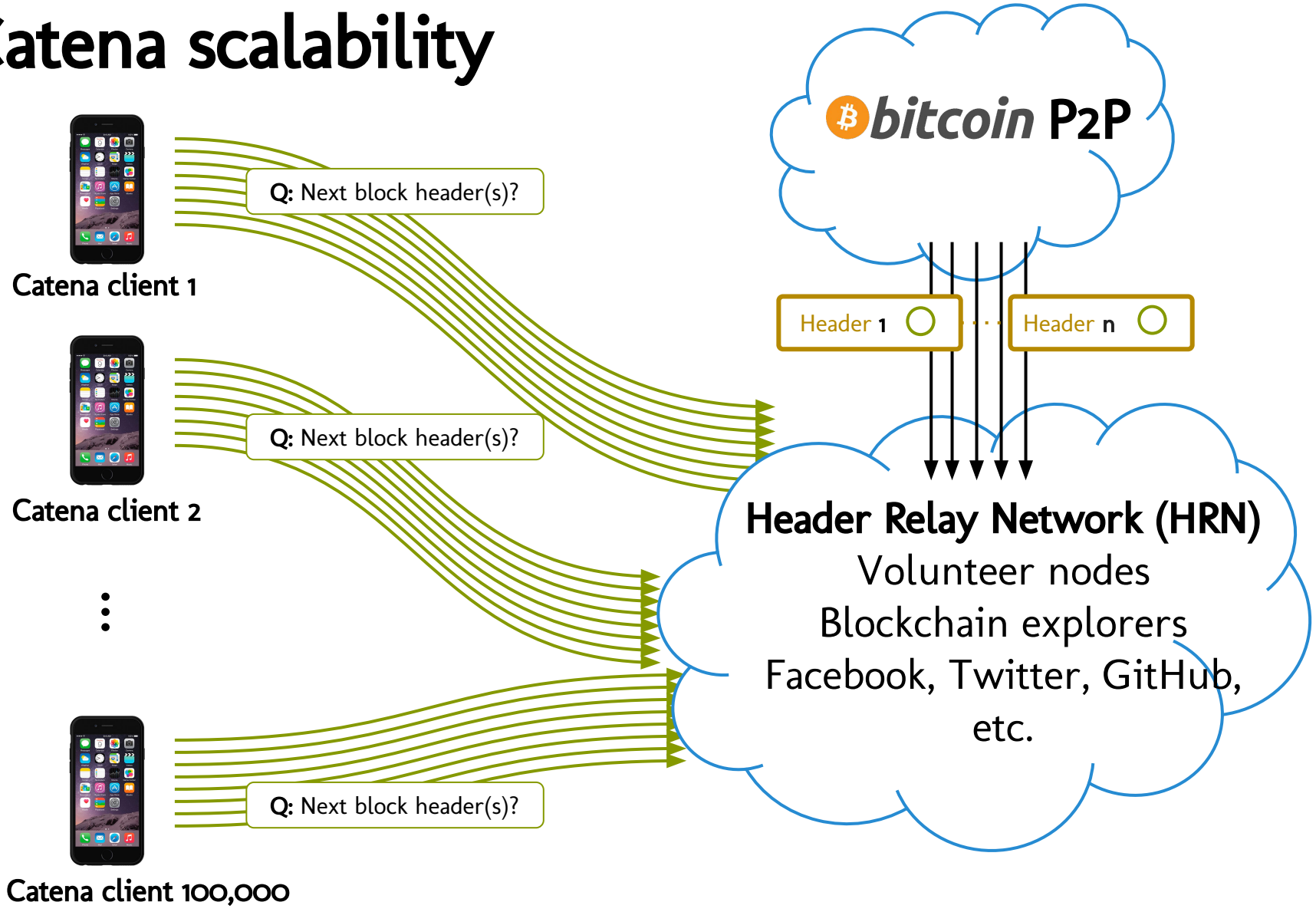
Catena client 2

⋮

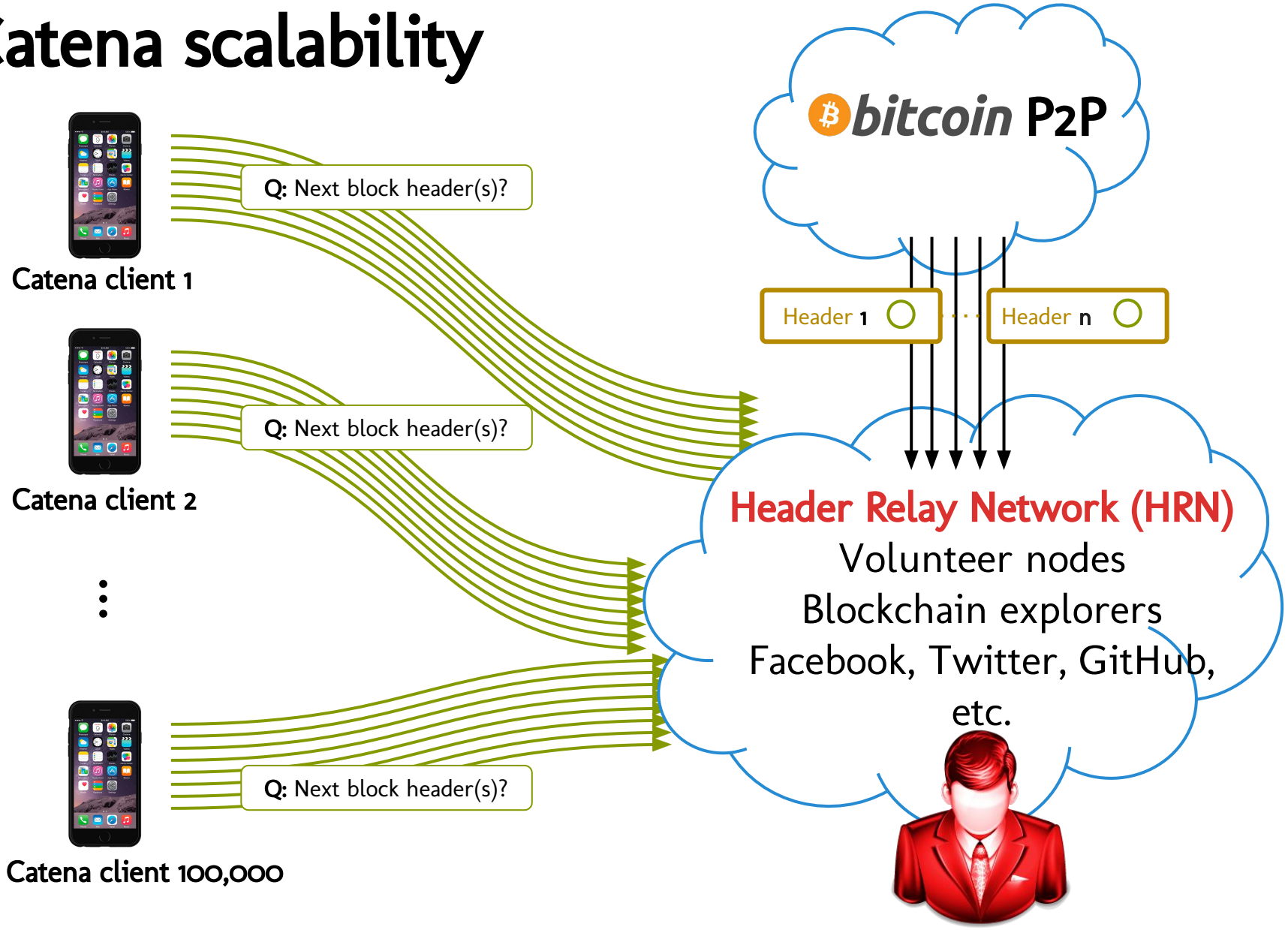
Catena client 100,000



Catena scalability



Catena scalability



Conclusions

Conclusions

What we did:

- Enabled applications to efficiently leverage Bitcoin's publicly-verifiable consensus
  - Download transactions selectively rather than full blockchain
  - ~41 MB instead of gigabytes of bandwidth

Conclusions

What we did:

- Enabled applications to efficiently leverage Bitcoin's publicly-verifiable consensus
  - Download transactions selectively rather than full blockchain
  - ~41 MB instead of gigabytes of bandwidth

Why it matters:

- Public-key directories for HTTPS and secure messaging
- Tor Consensus Transparency
- Software transparency schemes
- Turn fork consistency into full consistency

Conclusions

What we did:

- Enabled applications to efficiently leverage Bitcoin's publicly-verifiable consensus
  - Download transactions selectively rather than full blockchain
  - ~41 MB instead of gigabytes of bandwidth

Why it matters:

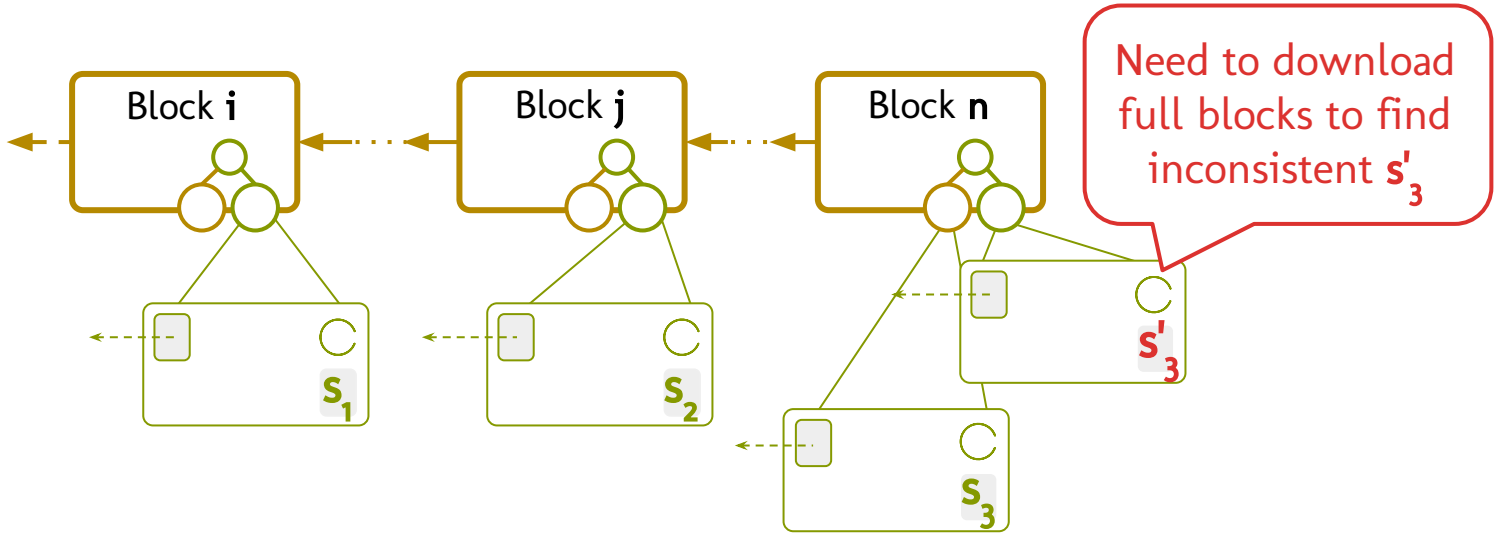
- Public-key directories for HTTPS and secure messaging
- Tor Consensus Transparency
- Software transparency schemes
- Turn fork consistency into full consistency

For more, read our paper!

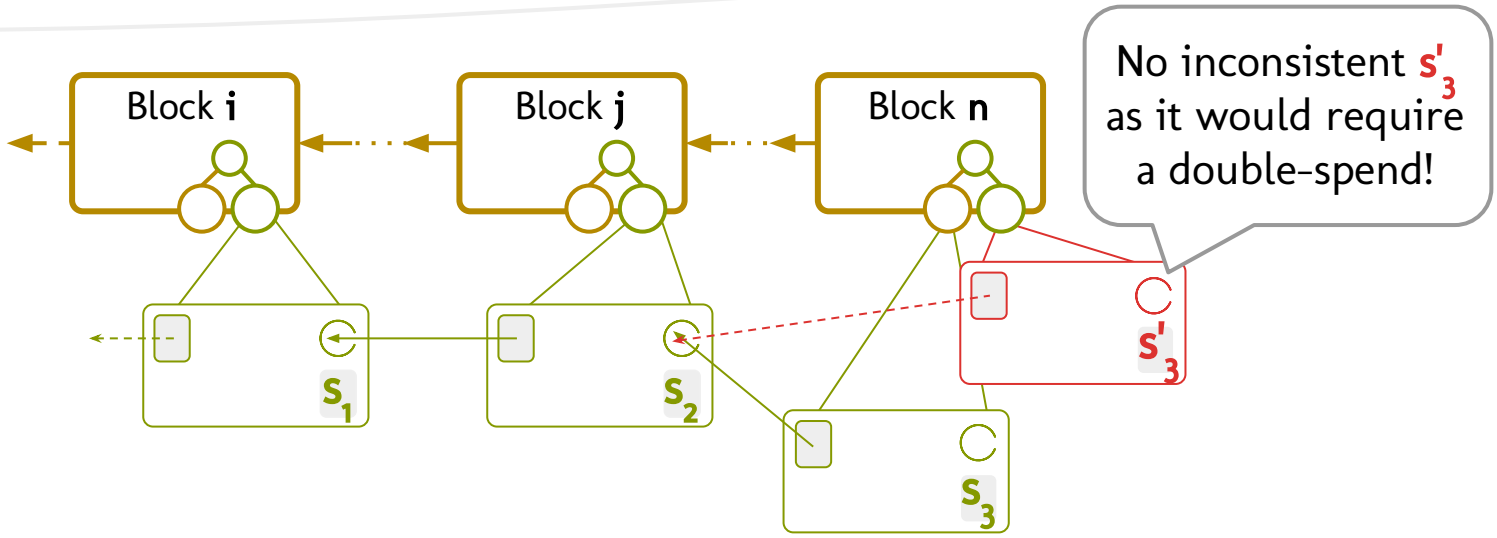
Ask me questions!

https://github.com/alinush/catena-java

Previous work

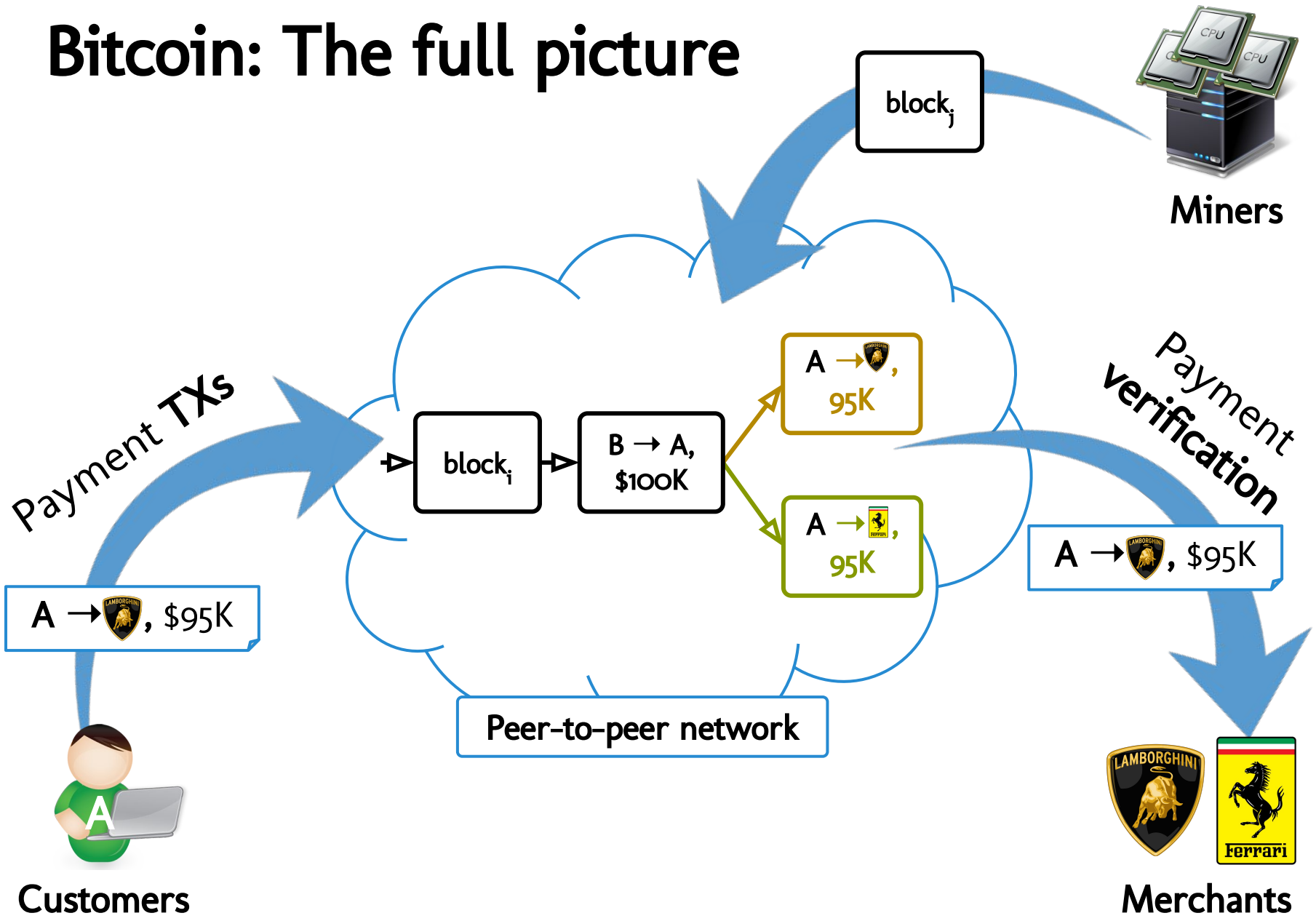


Catena



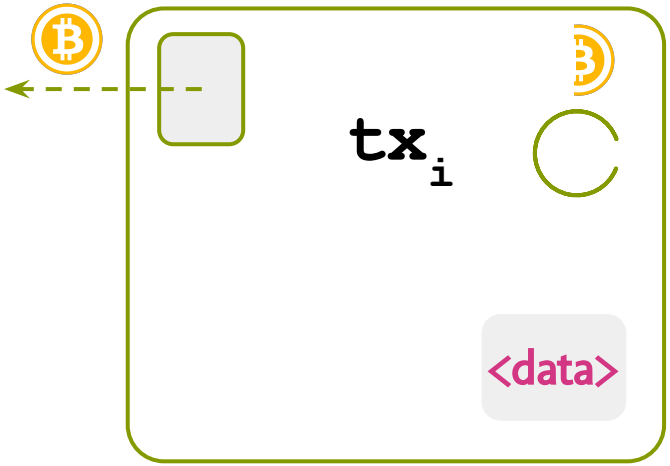
Extra slides

Bitcoin: The full picture



Catena transaction format

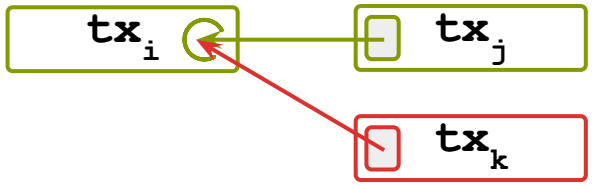
Coins from server for paying TX fees (digital signature)



"Change" coins back to server (public key)

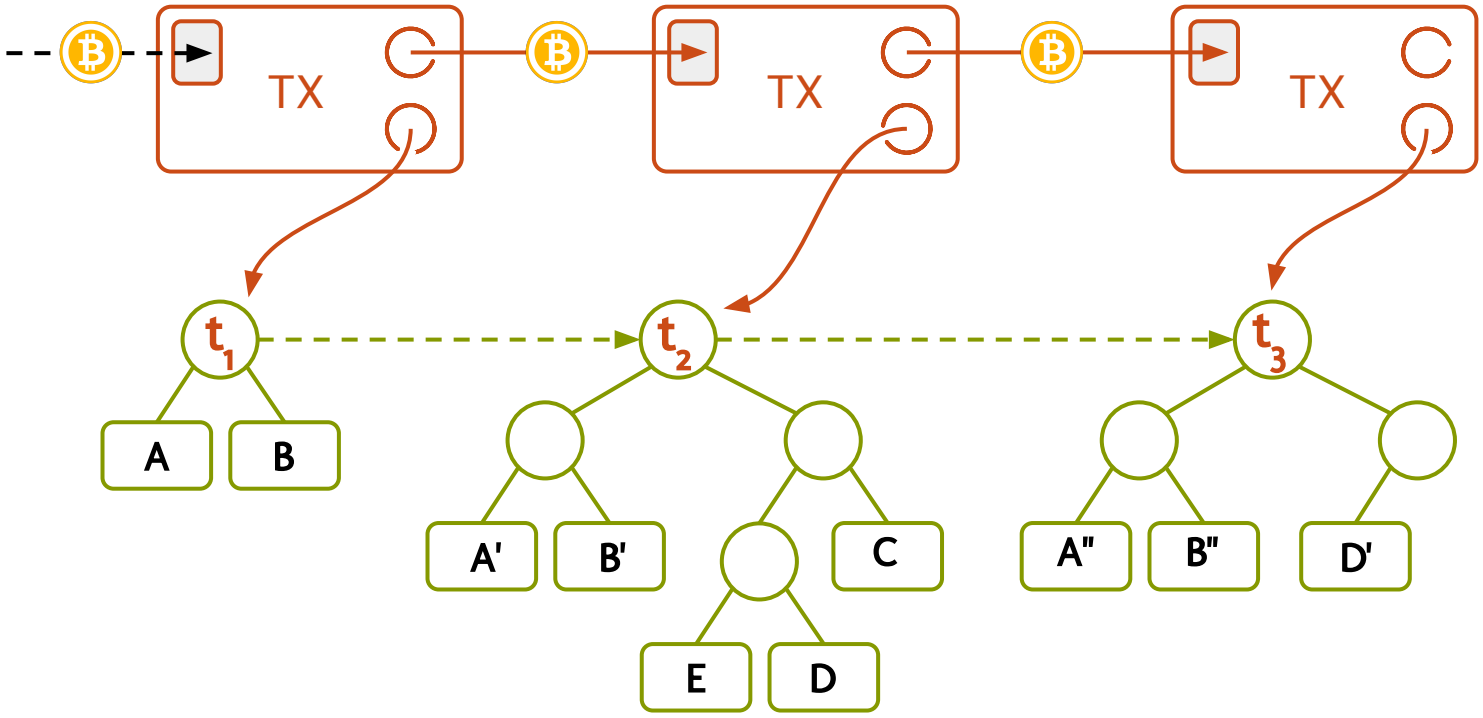
Unspendable `OP_RETURN` output with arbitrary data

A **single spendable output** ⇒ No forks



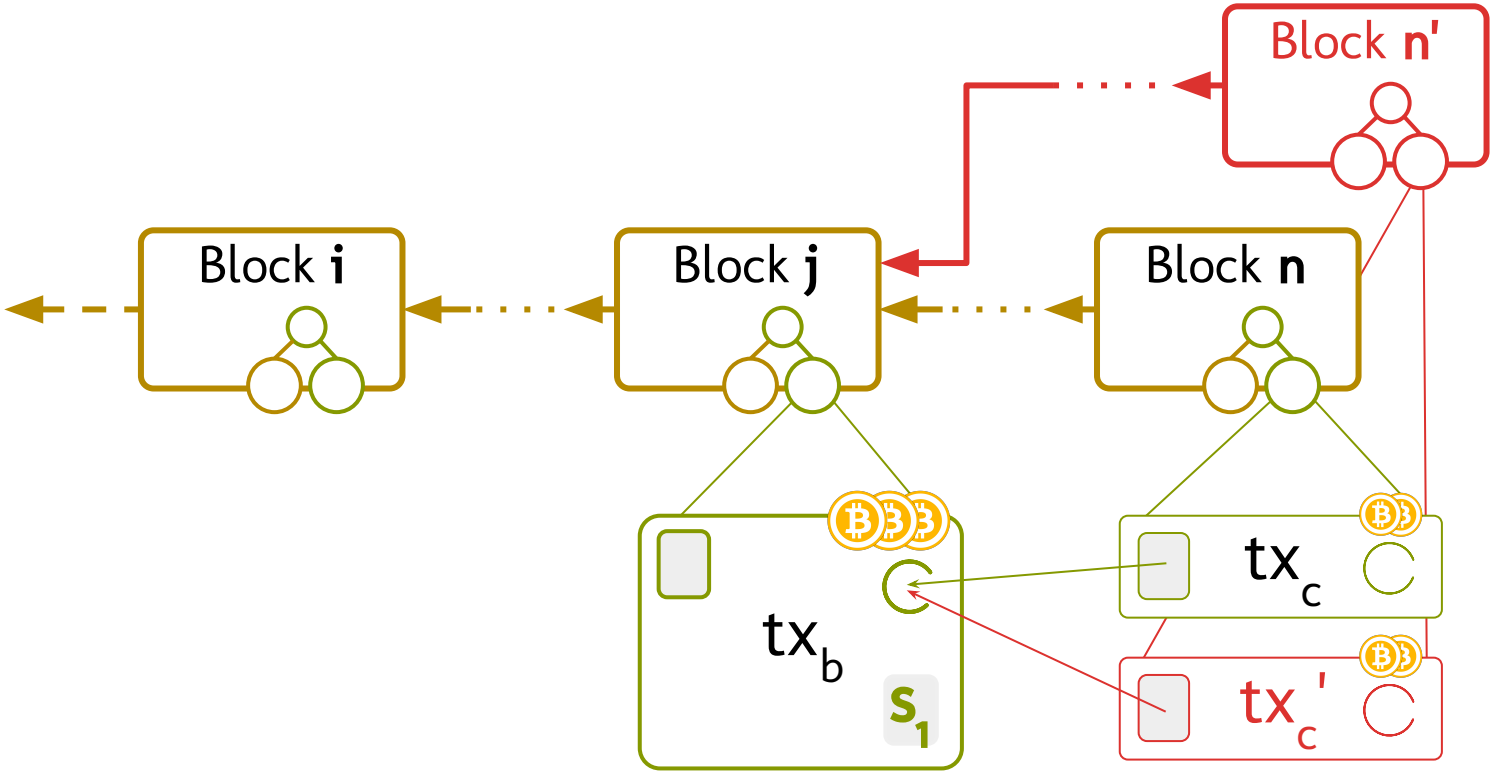
BKD: A Bitcoin-backed PKD

Catena: Hard-to-fork, append-only log (Bitcoin-backed)



BKD: Hard-to-fork public-key directory (Catena-backed)

Bitcoin blockchain



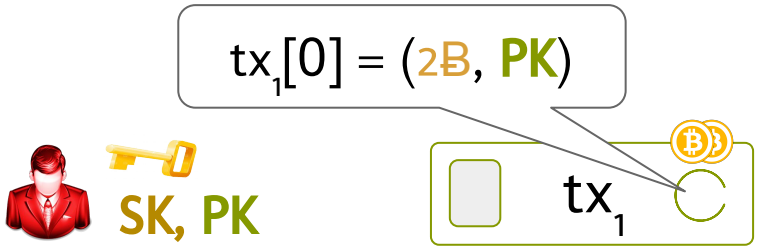
Blockchain forks ⇔ Double-spent coins

Previous work

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

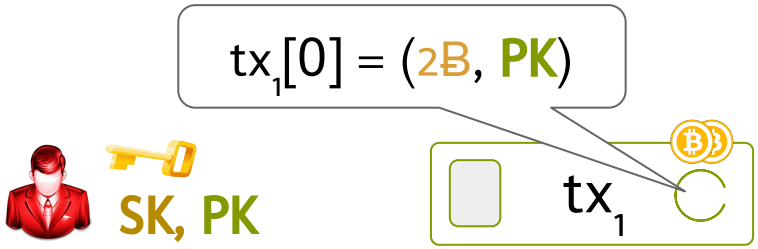
Previous work

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



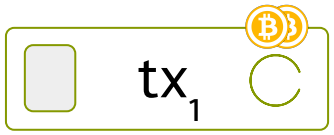
Previous work

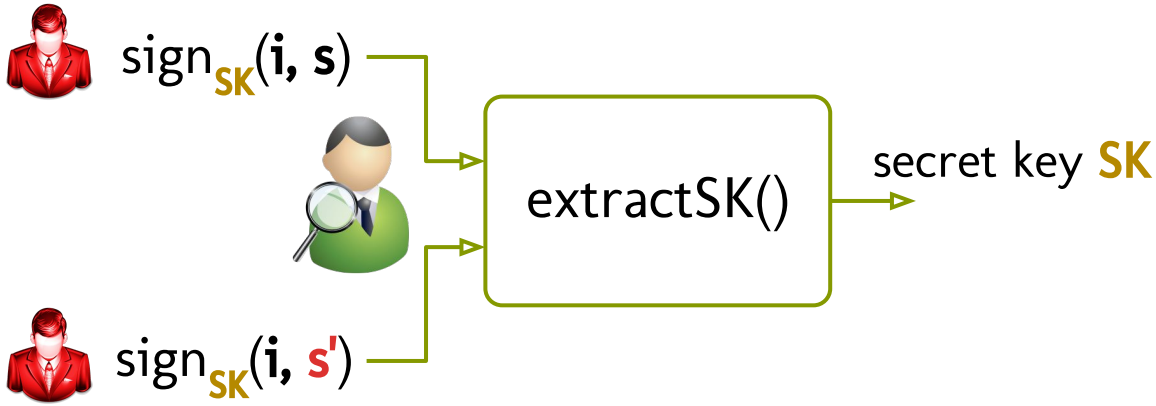
"Liar, liar, coins on fire!" (CCS '15)



Previous work

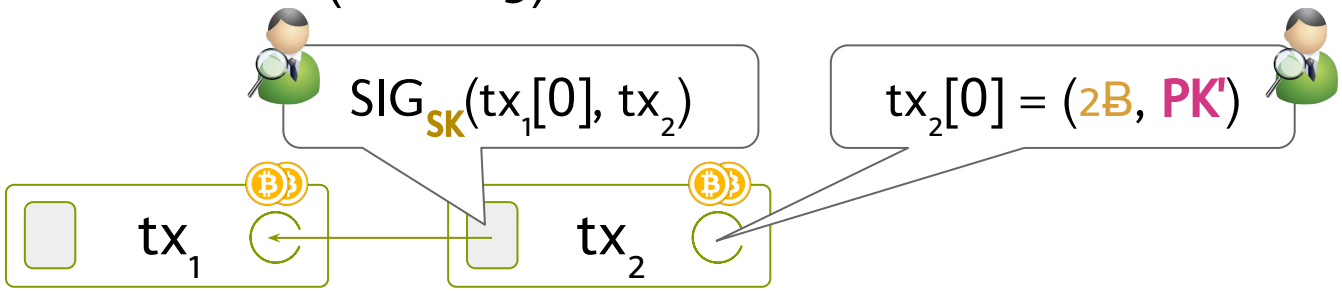
"Liar, liar, coins on fire!" (CCS '15)

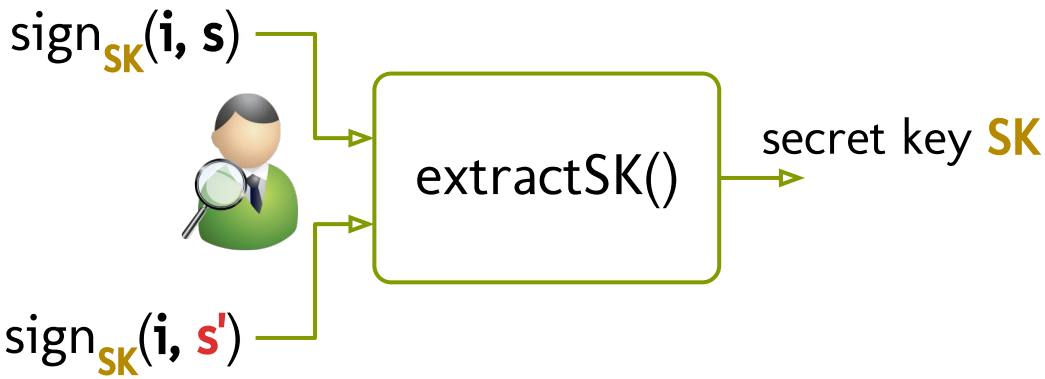




Previous work

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





Previous work

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