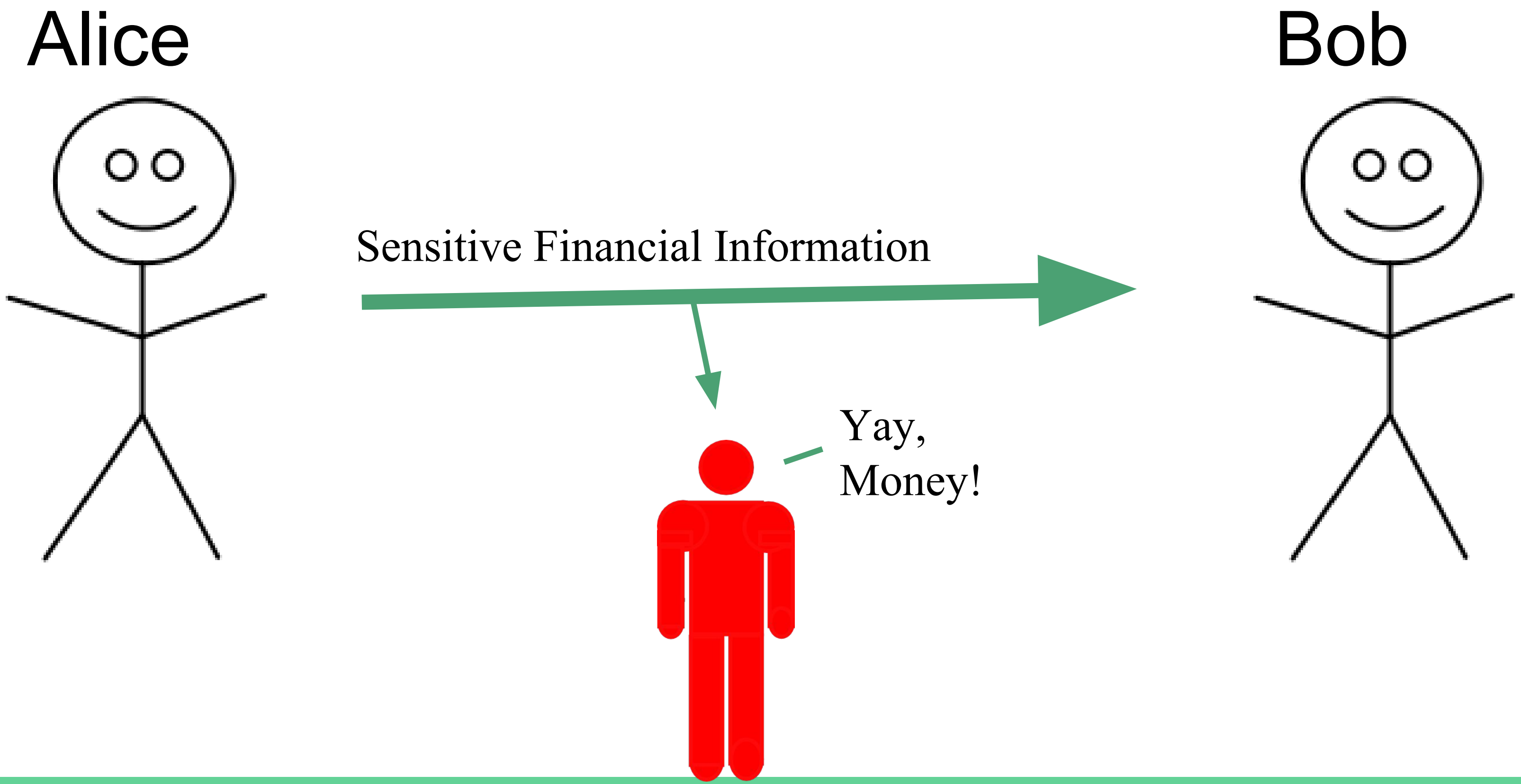
Keychat: Secure Messaging via Bitcoin

Robert Chen, John Kuszmaul, Yiming Zheng
Mentored By Alin Tomescu

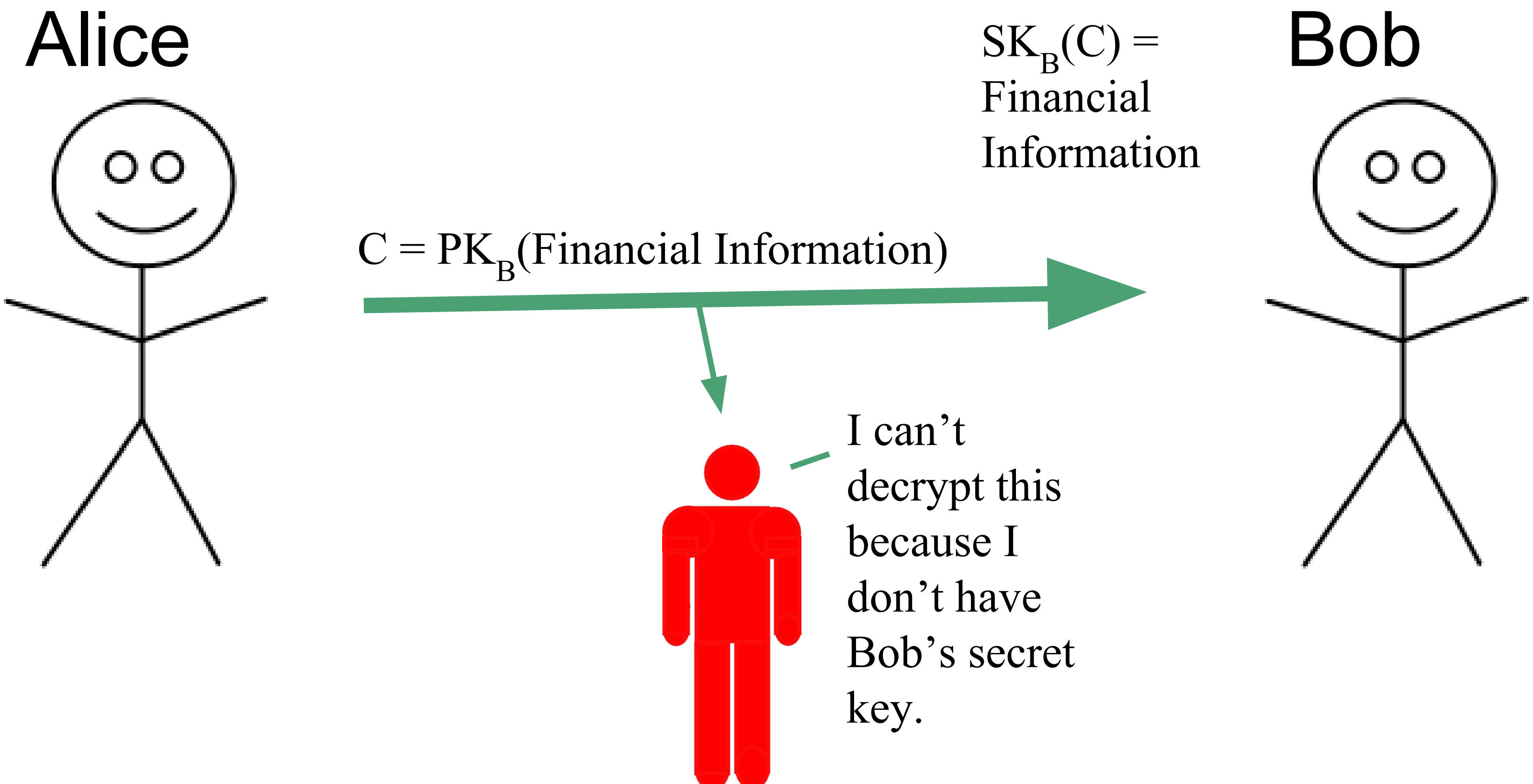
Motivation

- We want to secure communications.



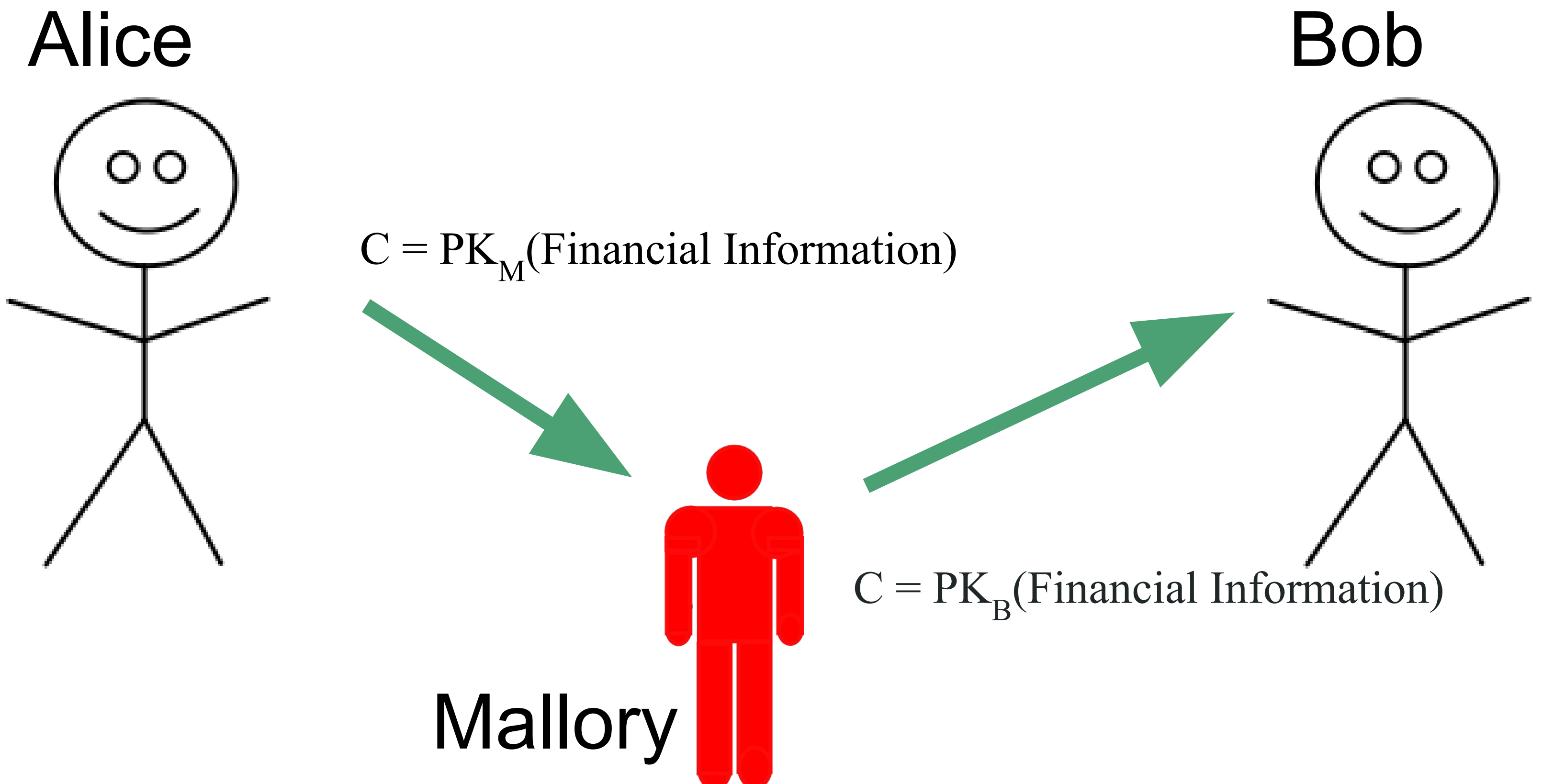
End-to-end Encryption

- People generate a key pair. They broadcast public keys.



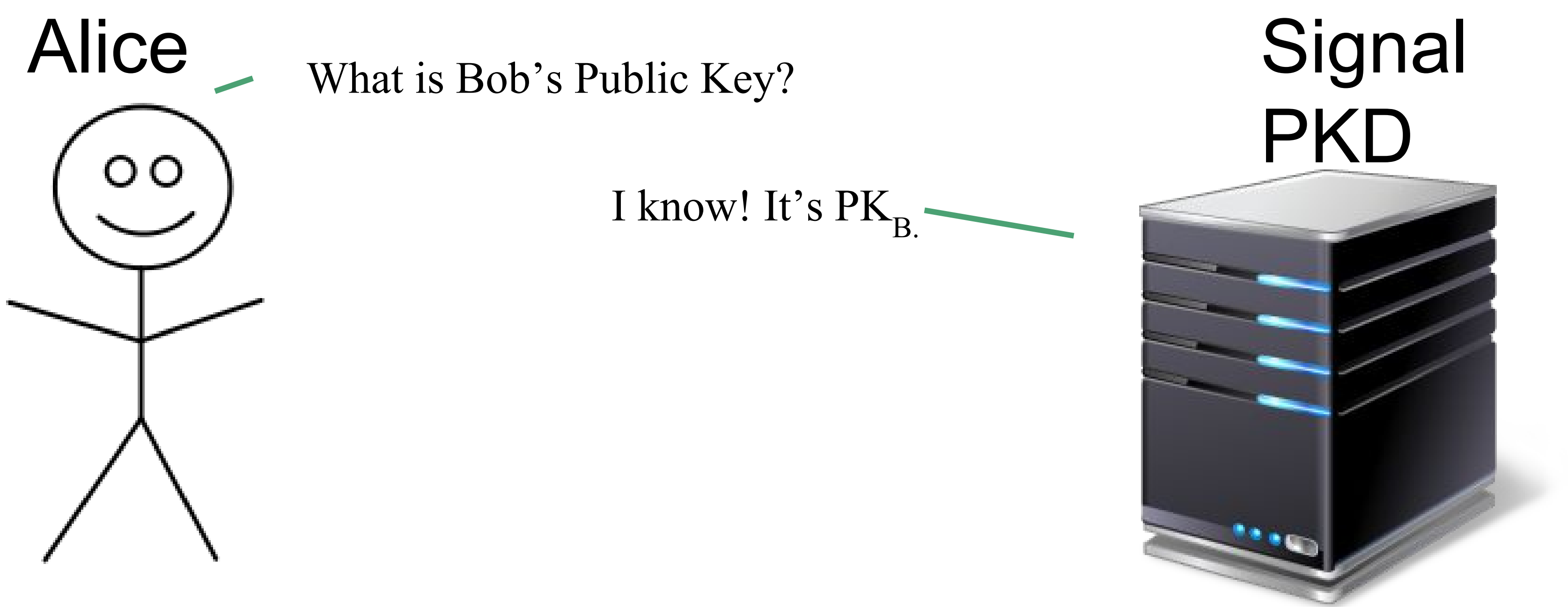
The Problem: Public Key Distribution

- People can encrypt messages, but they might be encrypting them for the wrong person.



The Current State of Common Communications Apps

- Facebook Messenger, Gmail - Do not use end-to-end encryption by default
- WhatsApp, Signal - Public keys are stored in a central directory that may be insecure.



Are append-only PKDs enough for security?

If append-only, then Bob can detect fake PK_M

What else can the malicious PKD do?

Public Key Directory

Name	Public Key
Alice	PK_A
Bob	PK_B

Our Work: Public Key Directory Equivocation

Version 1

Name	Public Key
Alice	PK_A

Version 2A

Alice's Perspective

Name	Public Key
Alice	PK_A
Bob	PK_M

Version 2B

Bob's Perspective

Name	Public Key
Alice	PK_A
Bob	PK_B

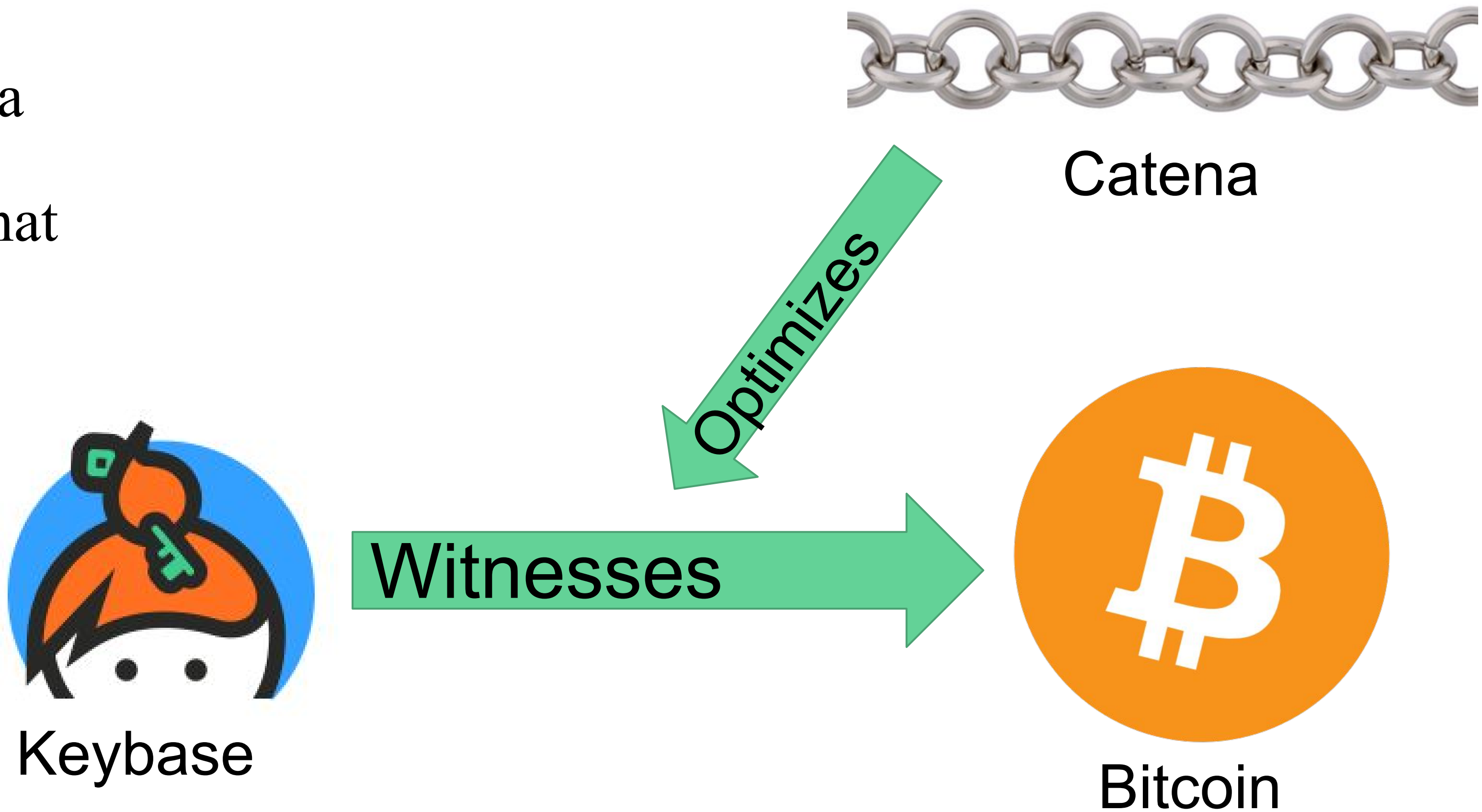
Outline

Keybase

Bitcoin

Catena

Keychat



Keybase: A public key directory

Keybase PKD Server

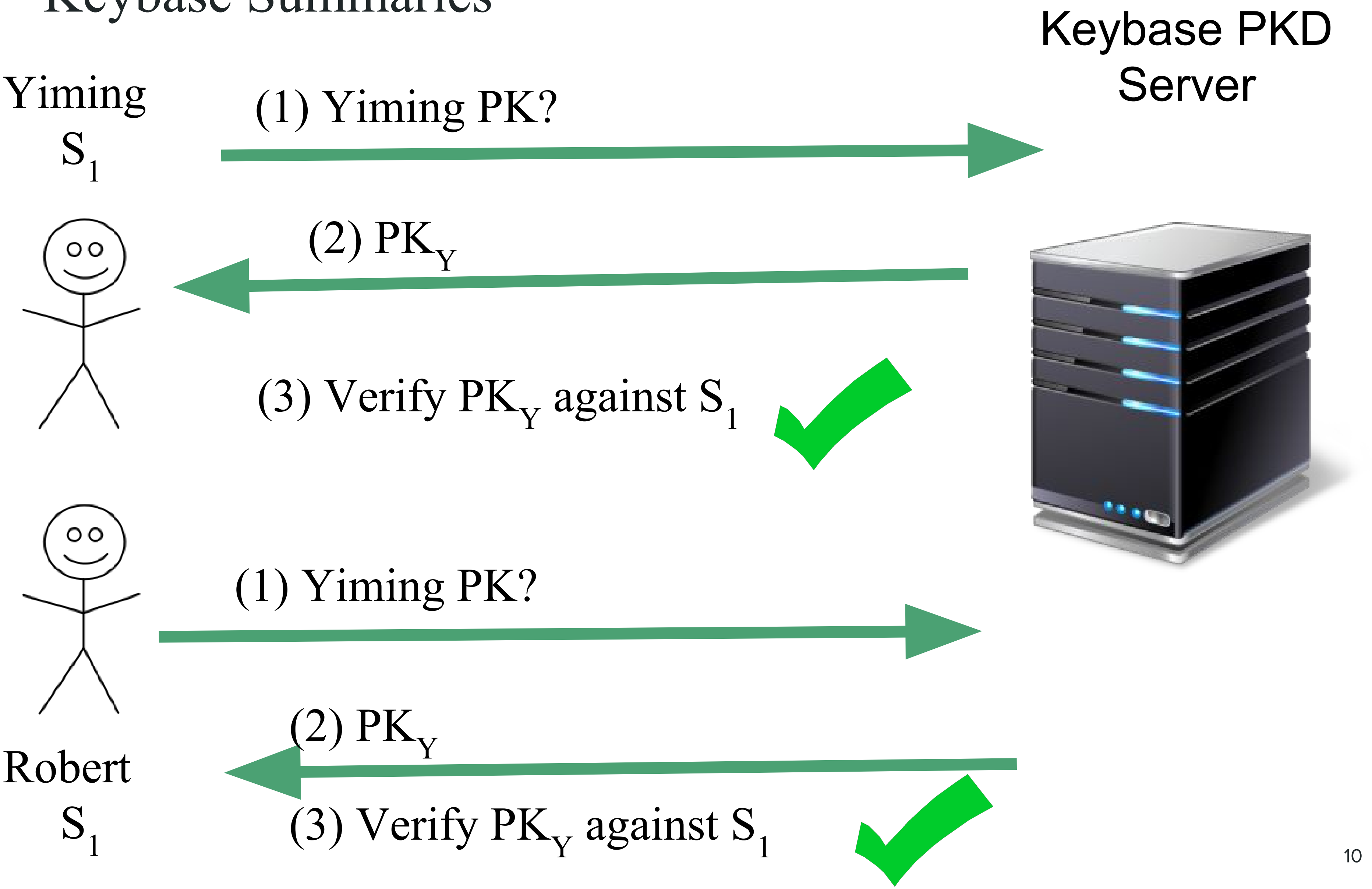
$S_1 = H(DIR_1)$

Name	Public Key
John	PK_J
Robert	PK_R

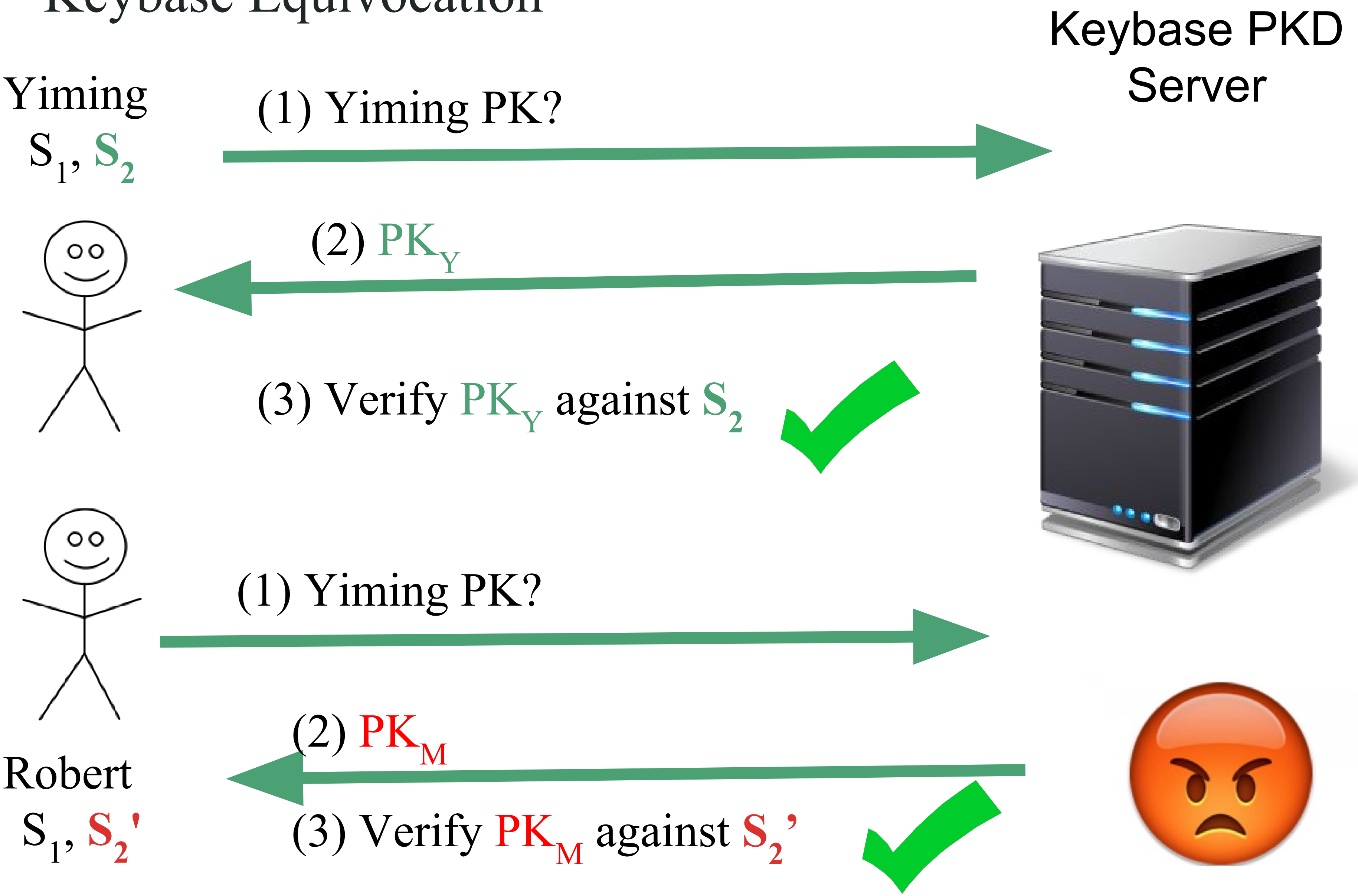
$S_2 = H(DIR_2)$

Name	Public Key
John	PK_J
Robert	PK_R
Yiming	PK_Y

Keybase Summaries

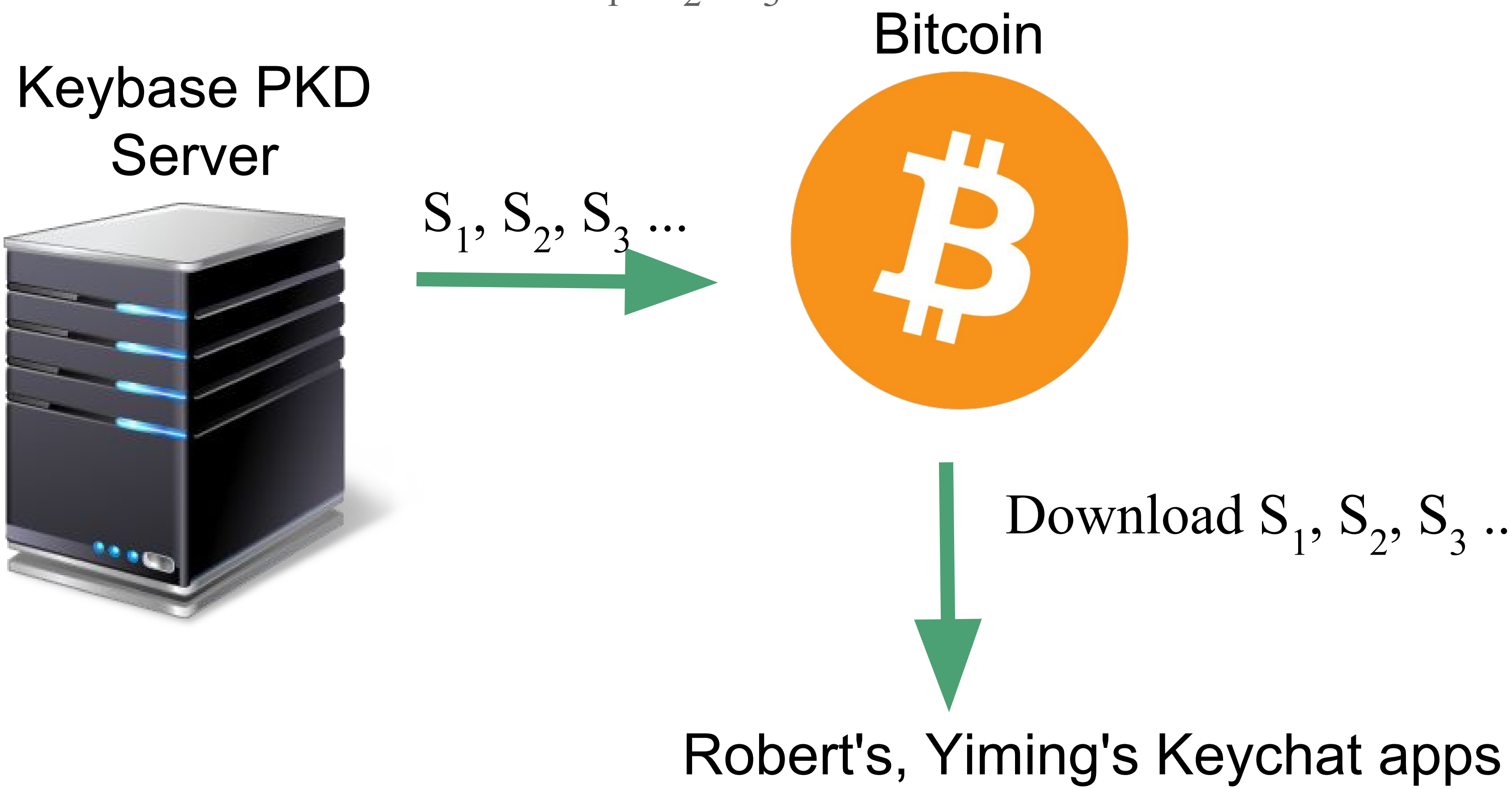


Keybase Equivocation



Keybase Non-equivocation

- Important that Yiming and Robert have the same history of hashes/summaries: S_1, S_2, S_3 ...



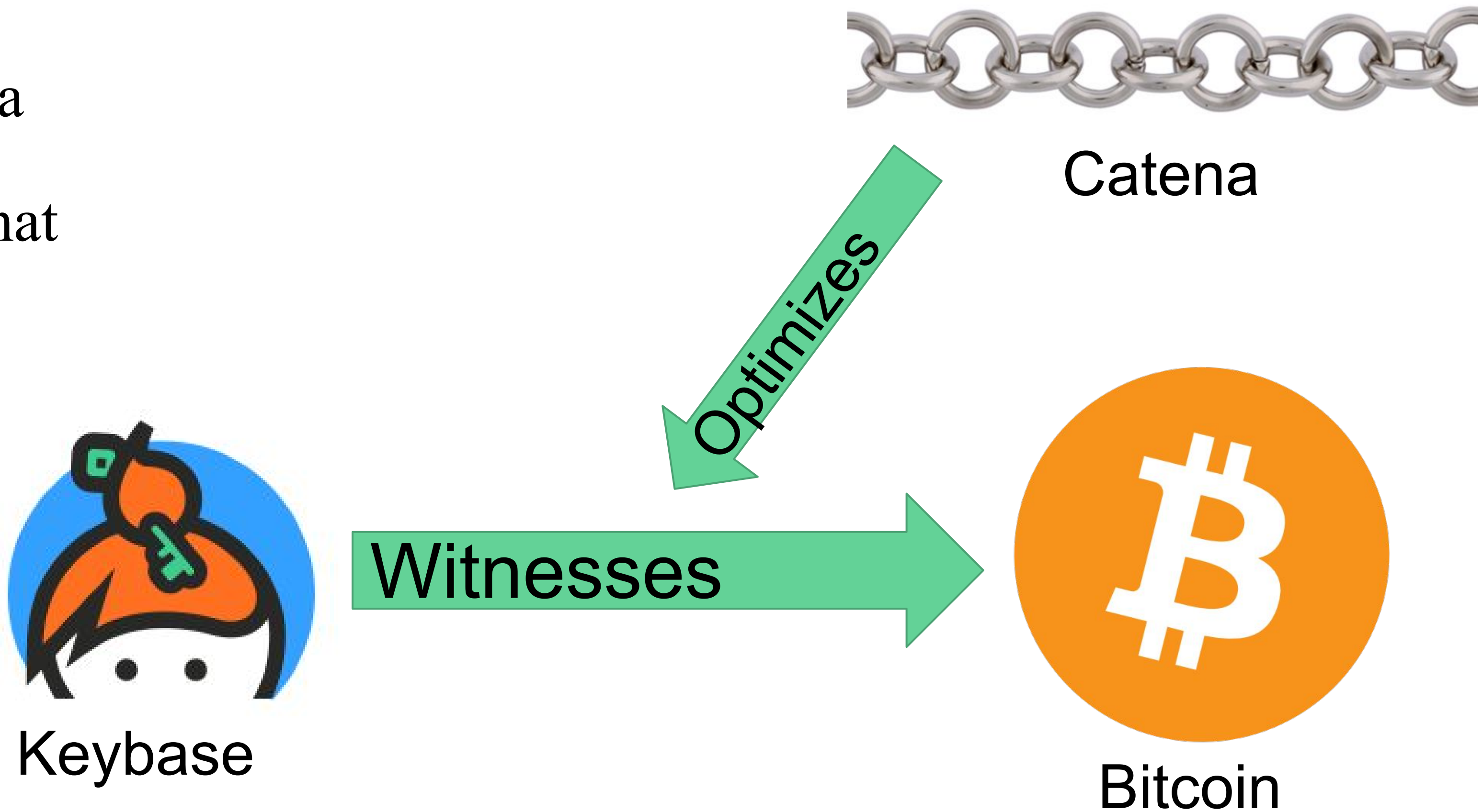
Outline

Keybase

Bitcoin

Catena

Keychat



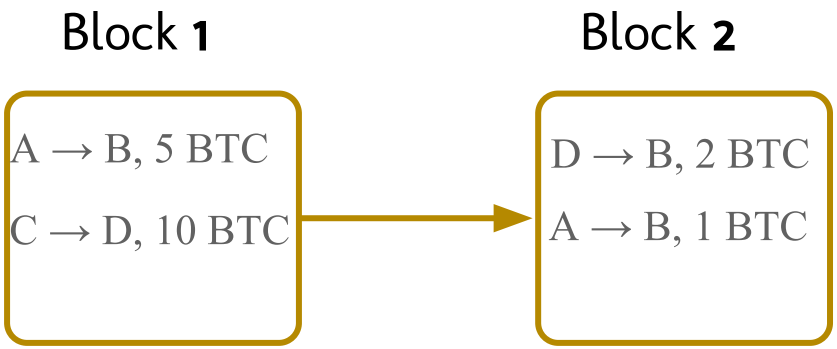
Bitcoin: Blockchain

Block **1**

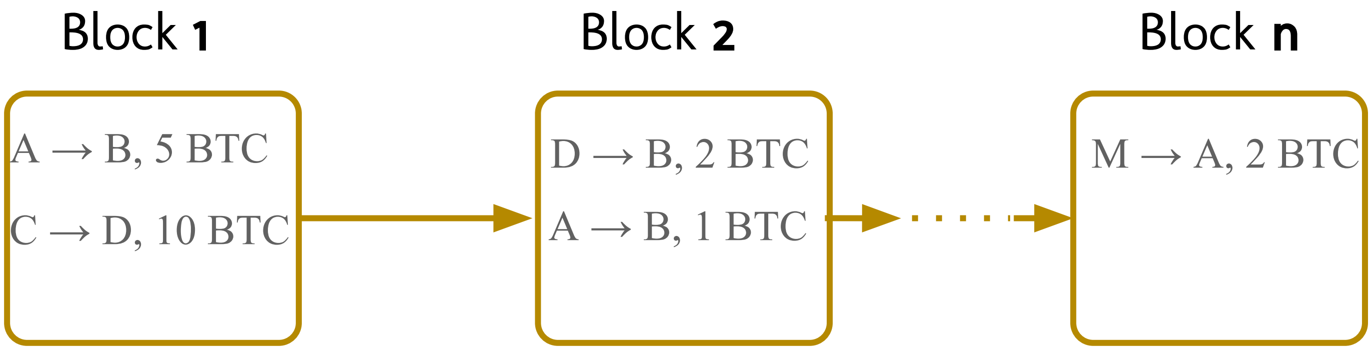
A → B, 5 BTC

C → D, 10 BTC

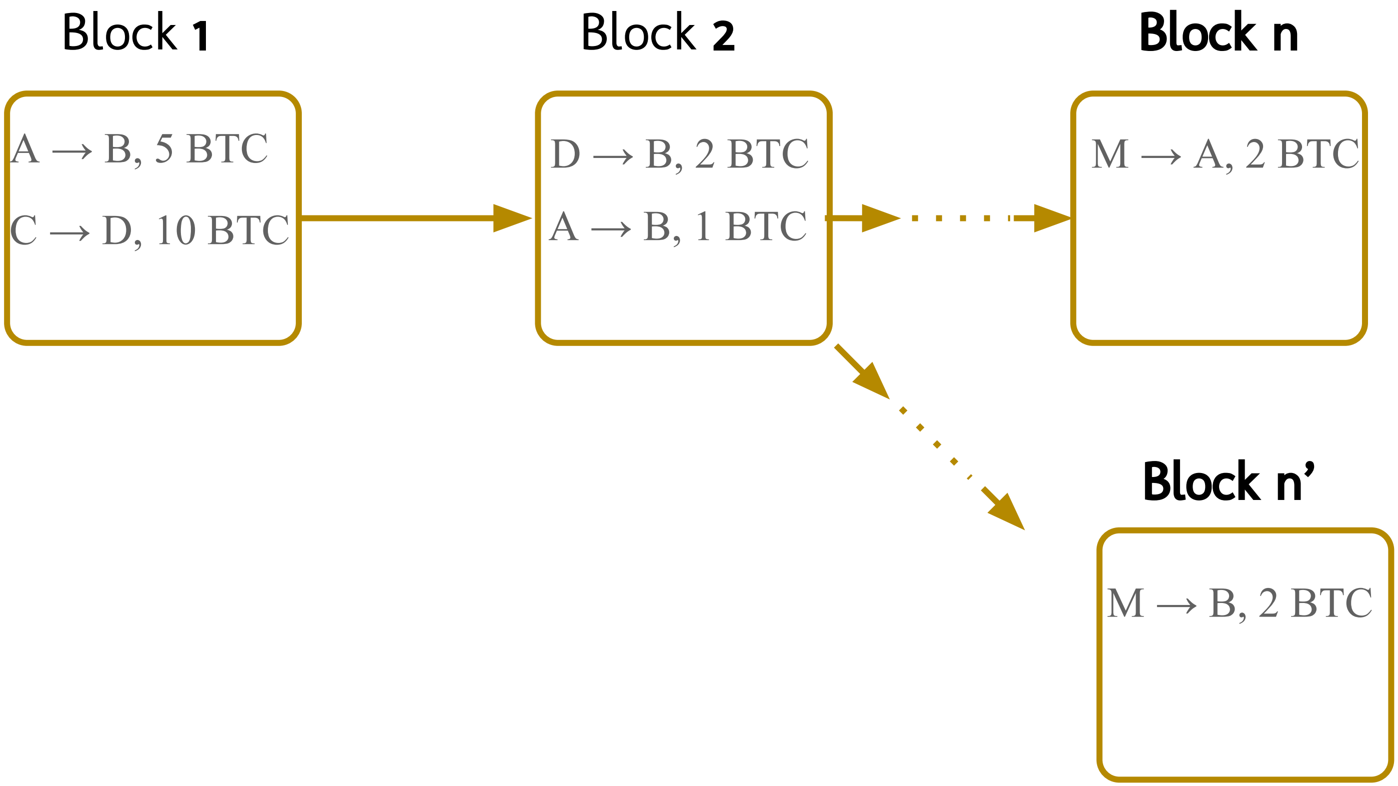
Bitcoin: Blockchain



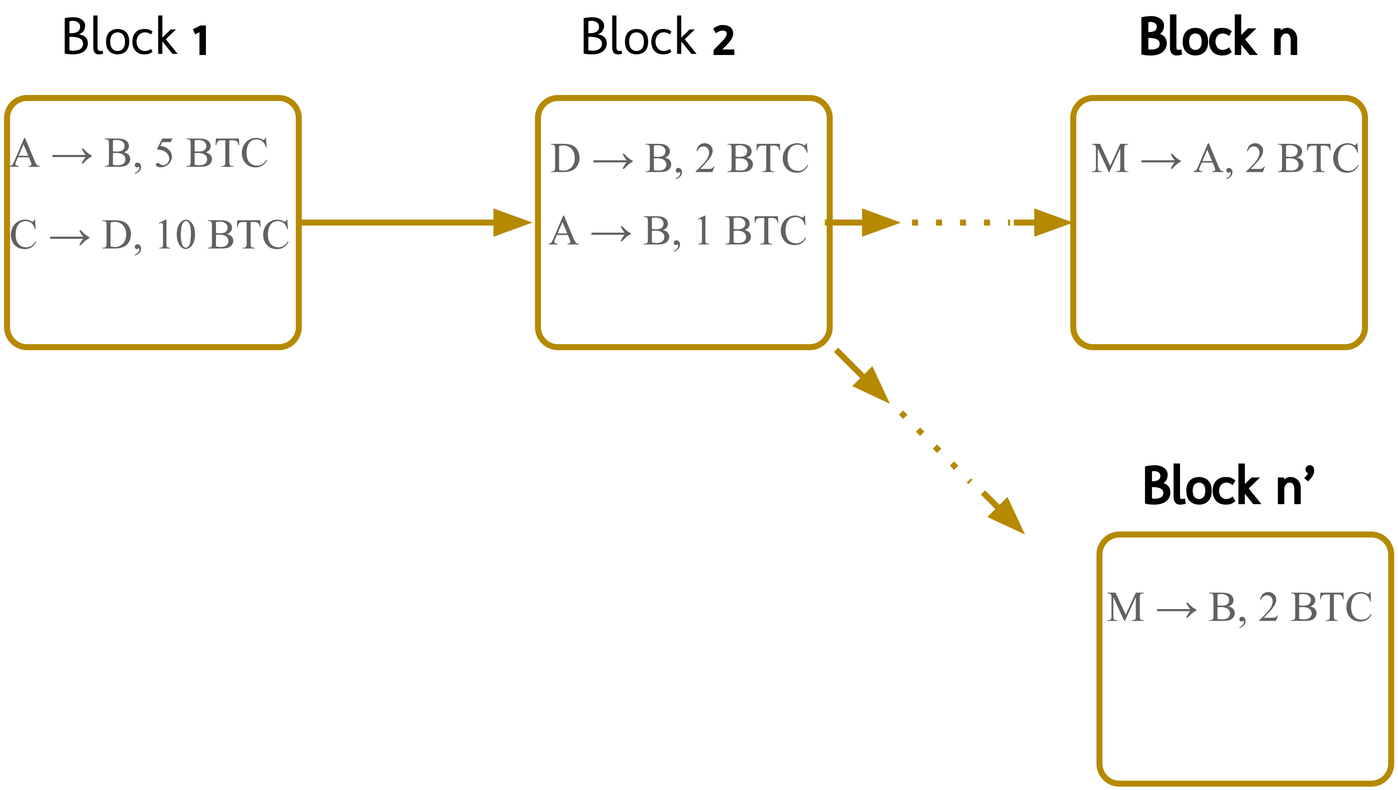
Bitcoin: Blockchain



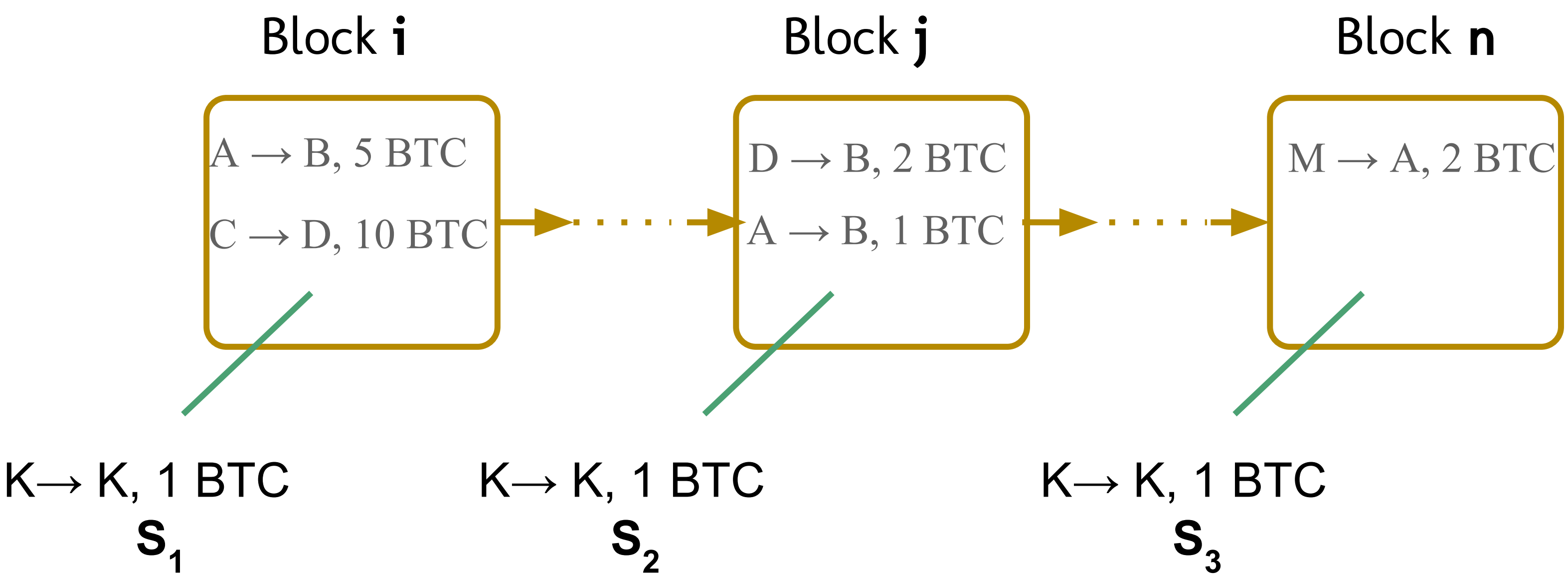
However, Can Forks Happen?



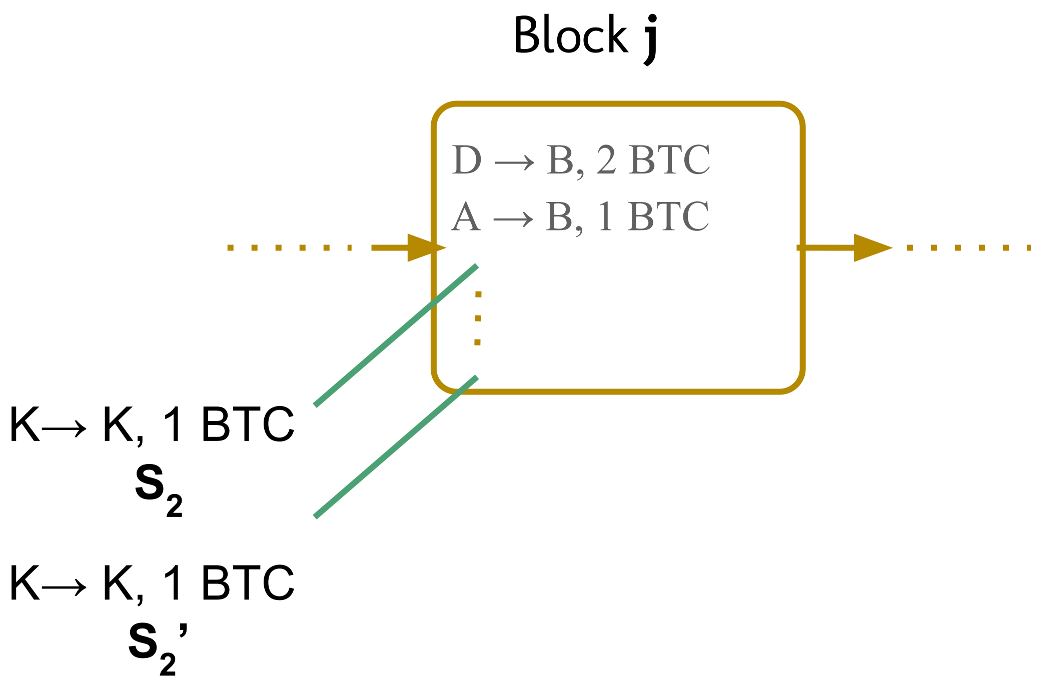
However, Can Forks Happen? Answer: **No!**



Keybase "Witnessing" Summaries



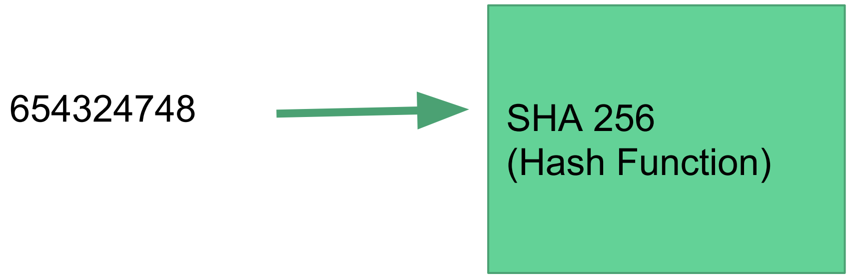
Equivocation Within a Block?



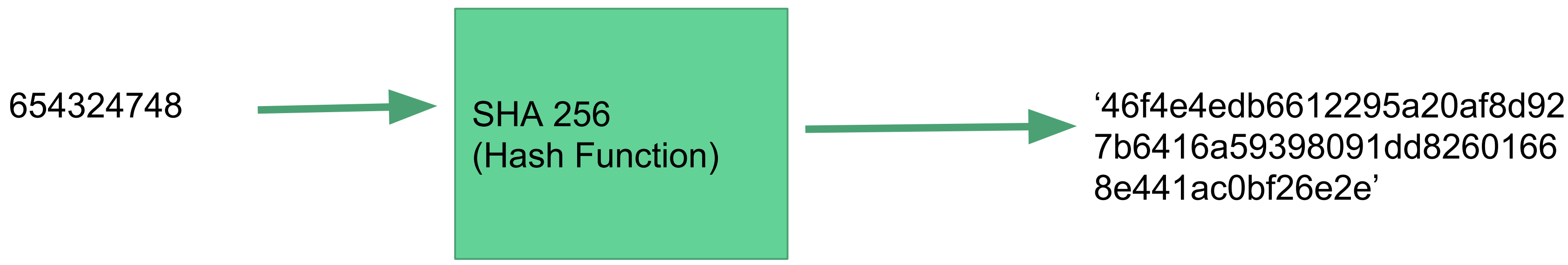
Hashes: A Reminder

SHA 256
(Hash Function)

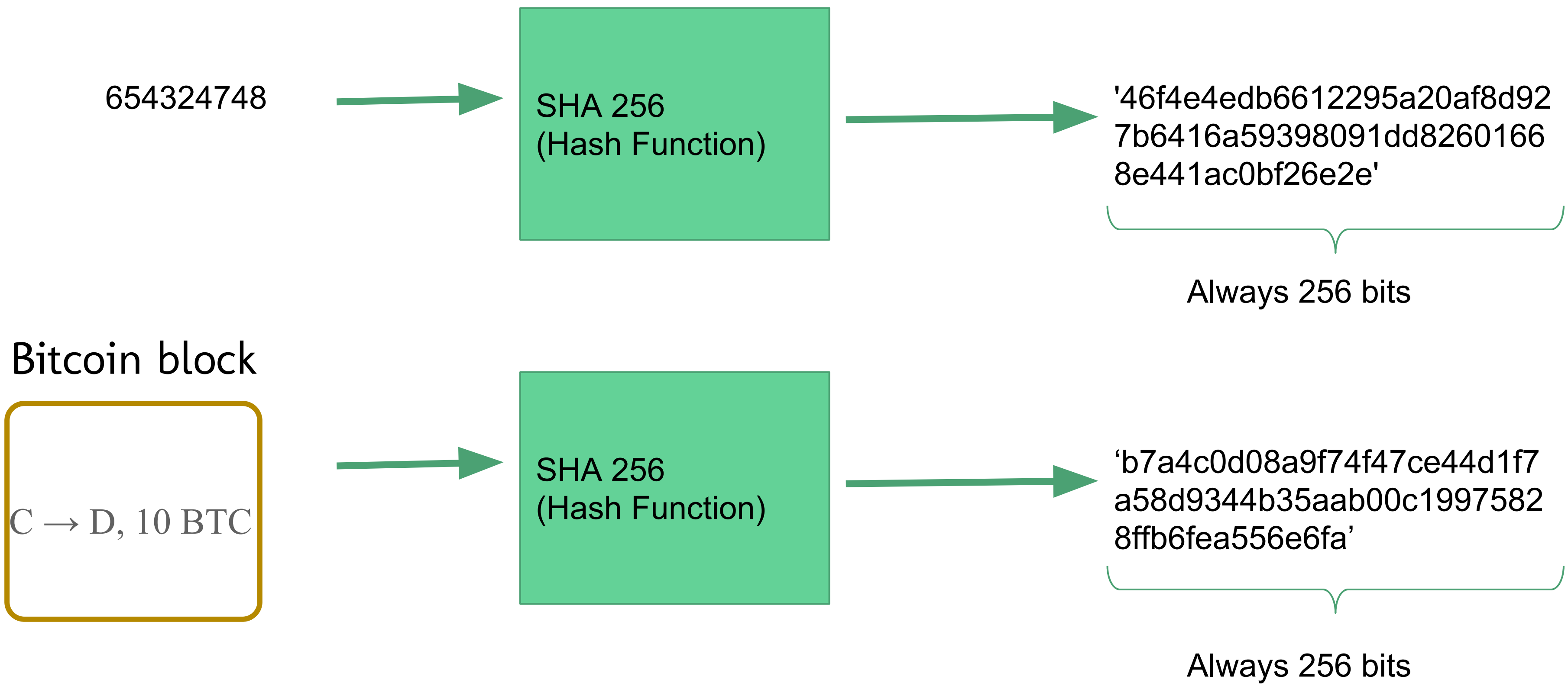
Hashes: A Reminder



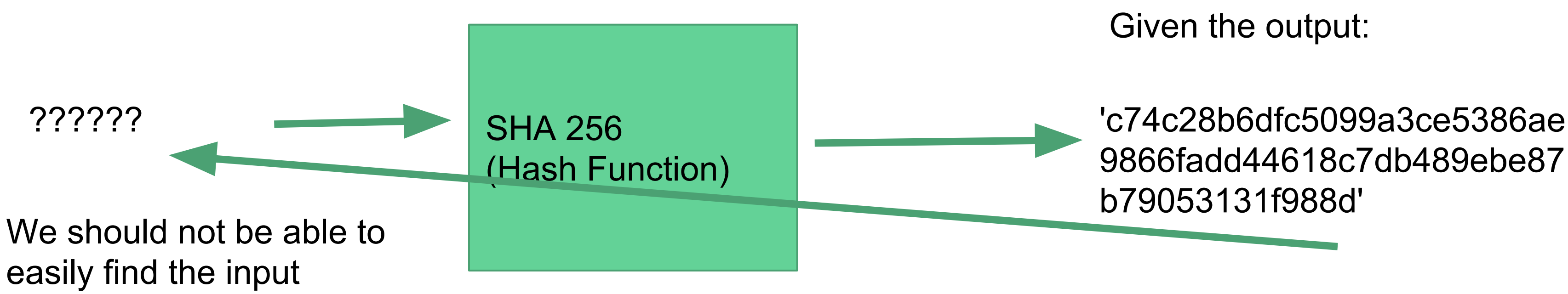
Hashes: A Reminder



More Properties of Ideal Hash Functions



More Properties of Ideal Hash Functions



One Way (OW):

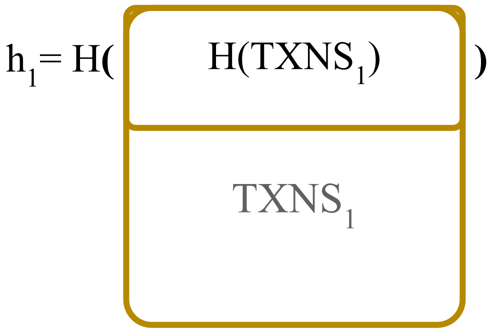
Given y, “infeasible” to find any x such that h(x) = y.

Takeaways

- Can feed in any size data as input → fixed length output
- Randomness: Small change in input data → entirely different output
- One-way: Given a hash, can’t feasibly find any data that hashes to it
- Perfect for a tamper-evident, append-only log!

Blockchain: A Closer Look!

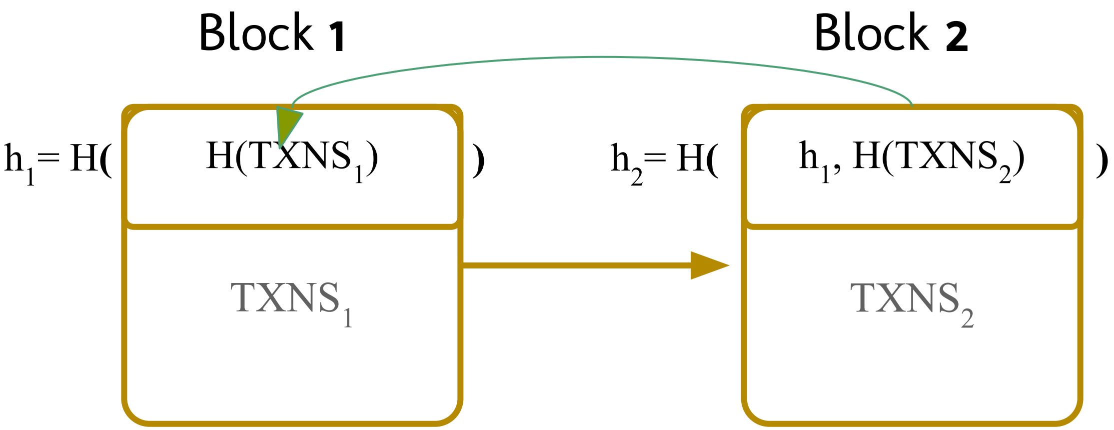
Block **1**



Block Header:

- Contains hash pointer to TXN data
- Contains hash pointer to previous block

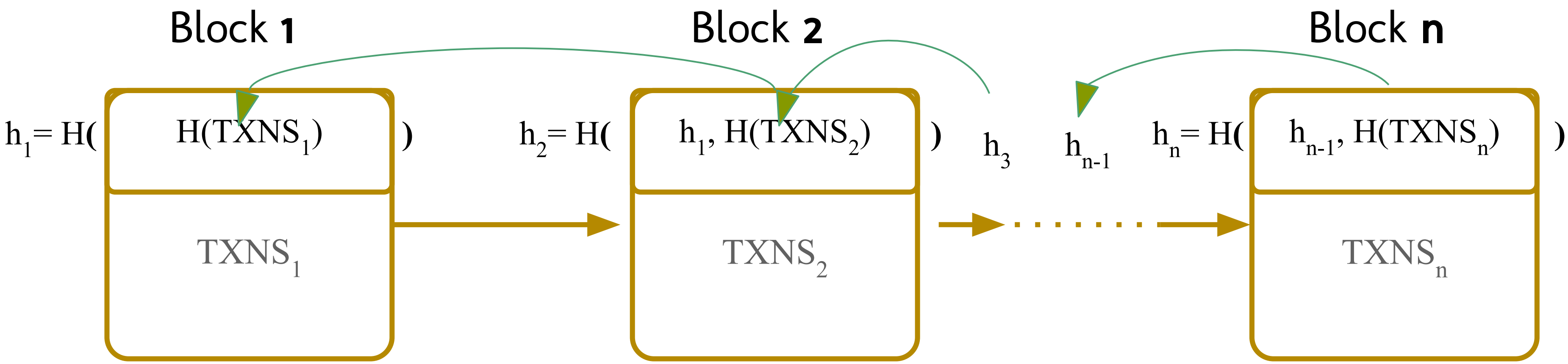
Blockchain: A Closer Look!



Block Header:

- Contains hash pointer to TXN data
- Contains hash pointer to previous block

Blockchain: A Closer Look!

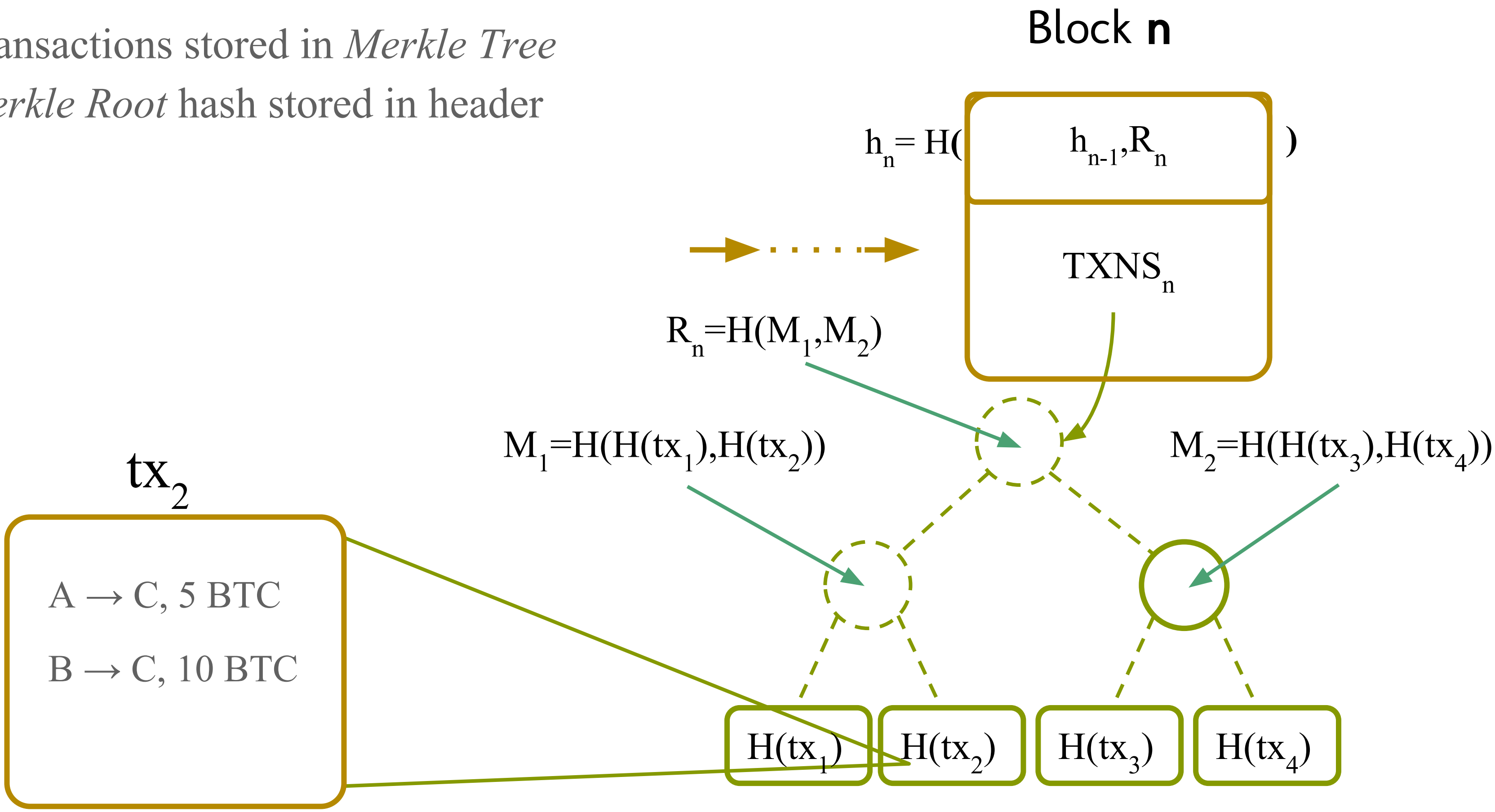


Block Header:

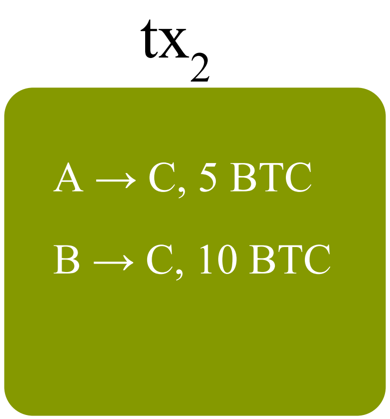
- Contains hash pointer to TXN data
- Contains hash pointer to previous block

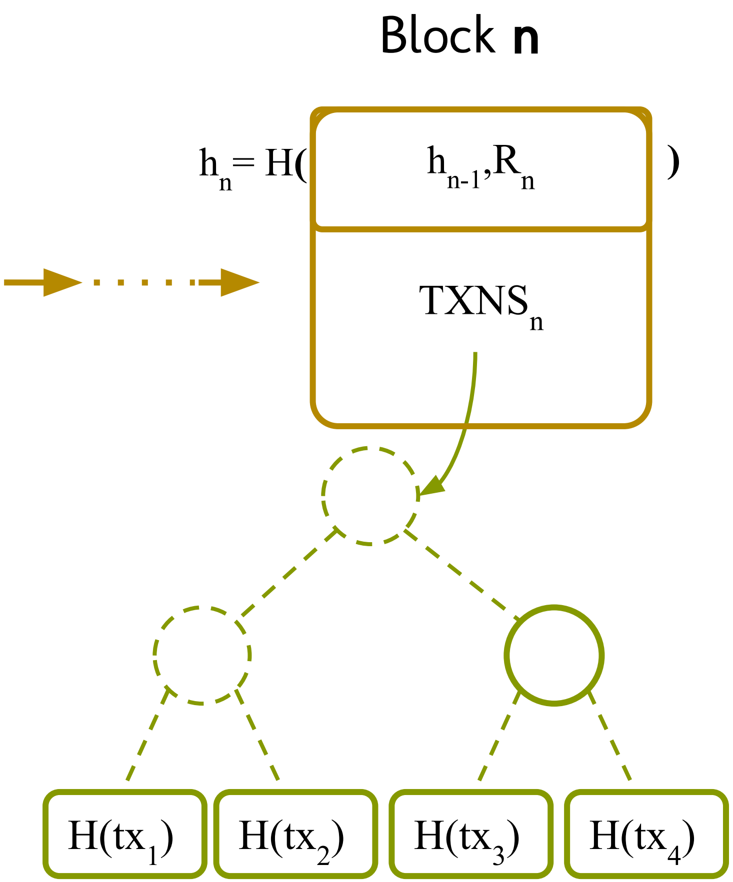
Transactions: A Closer Look!

- Transactions stored in *Merkle Tree*
- *Merkle Root* hash stored in header



Transactions: Membership Proof





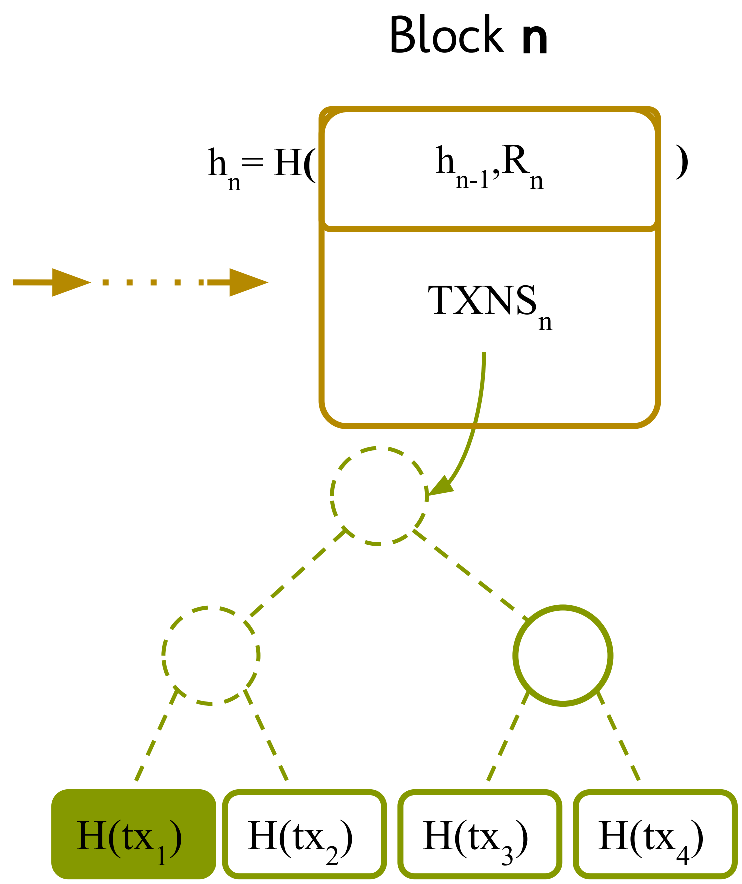
Membership proof for tx_2?

Transactions: Membership Proof

tx_2

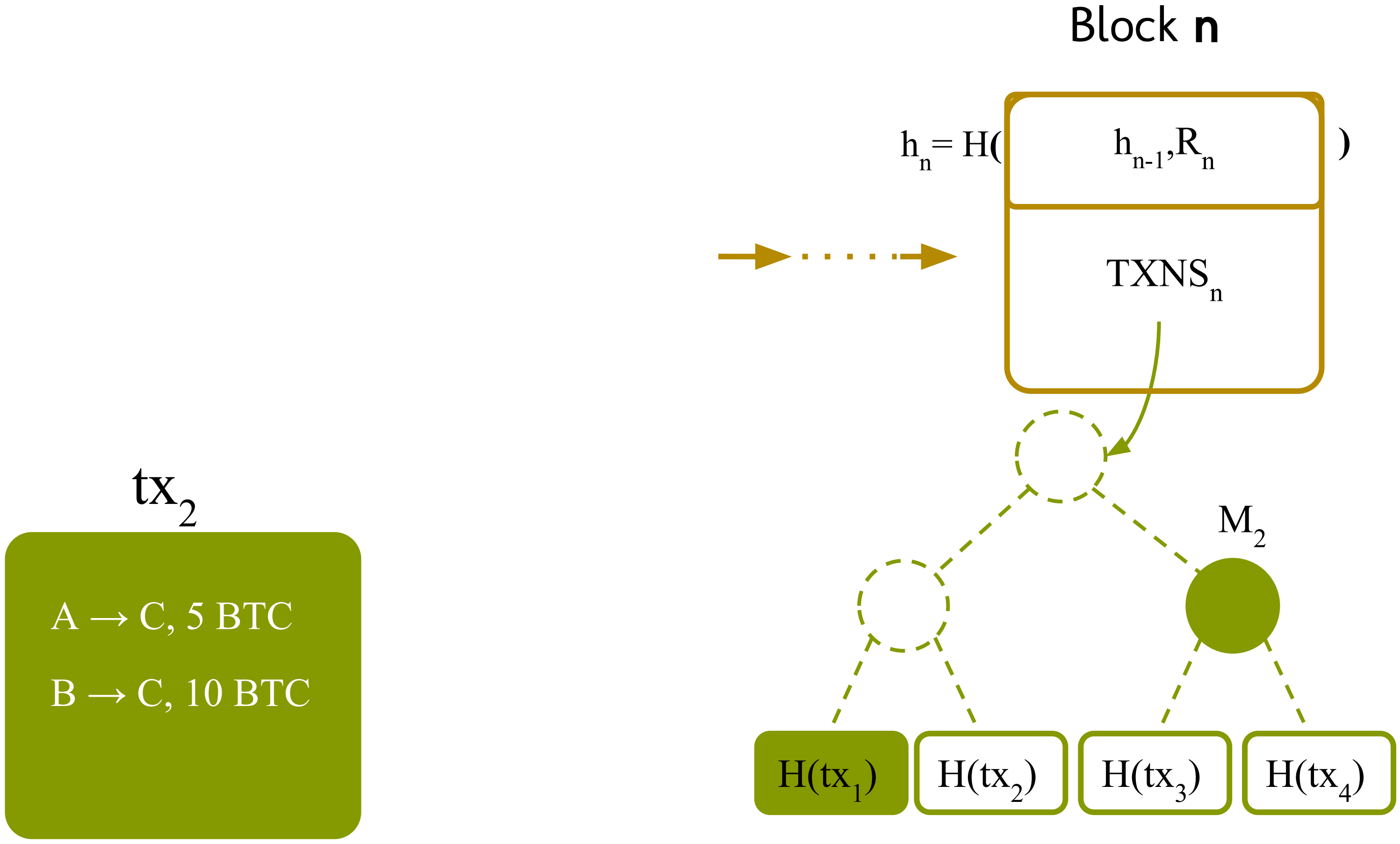
A → C, 5 BTC

B → C, 10 BTC



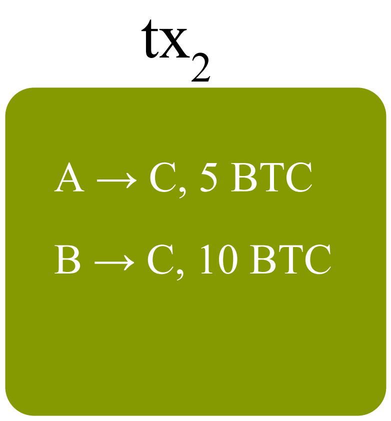
Membership proof for tx_2

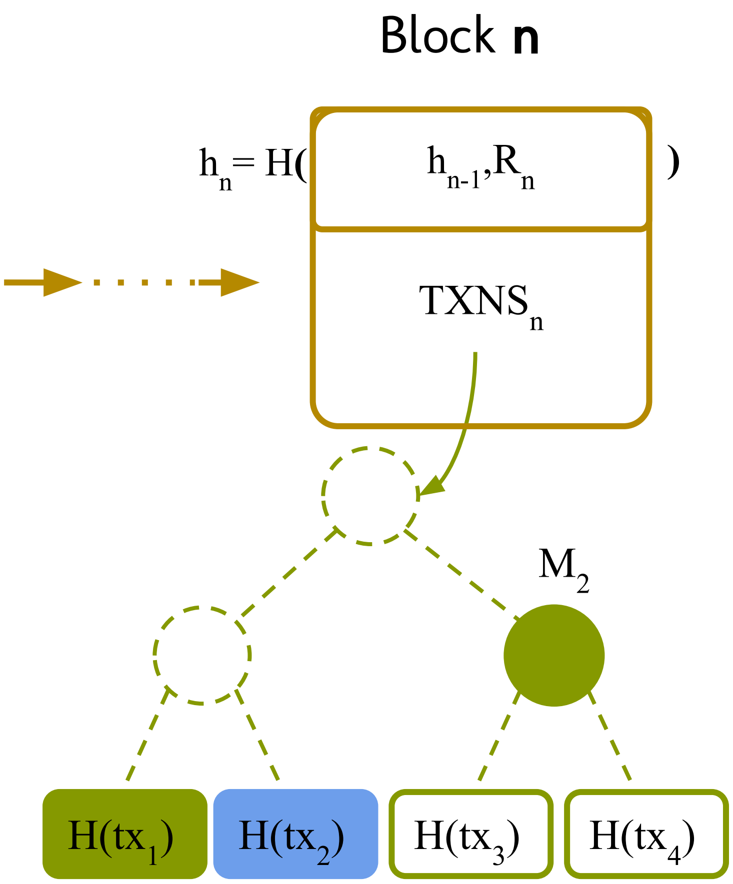
Transactions: Membership Proof



Membership proof for tx$_2$

Transactions: Membership Proof





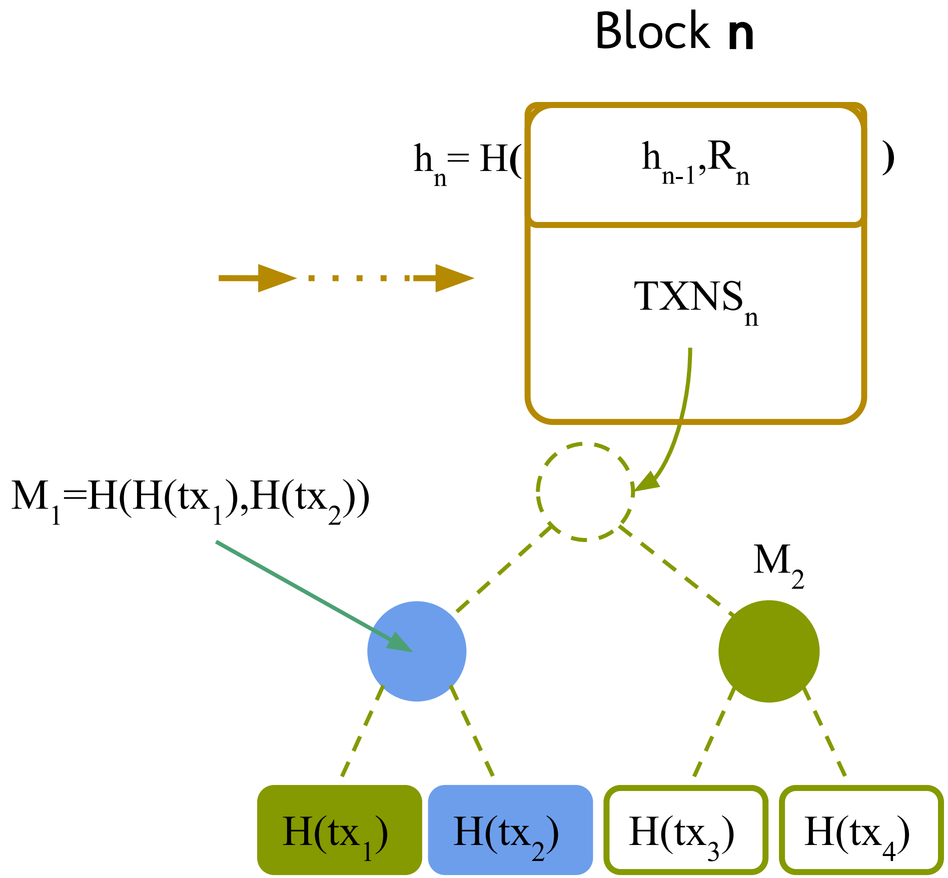
Membership proof for tx$_2$

Transactions: Membership Proof

tx_2

A → C, 5 BTC

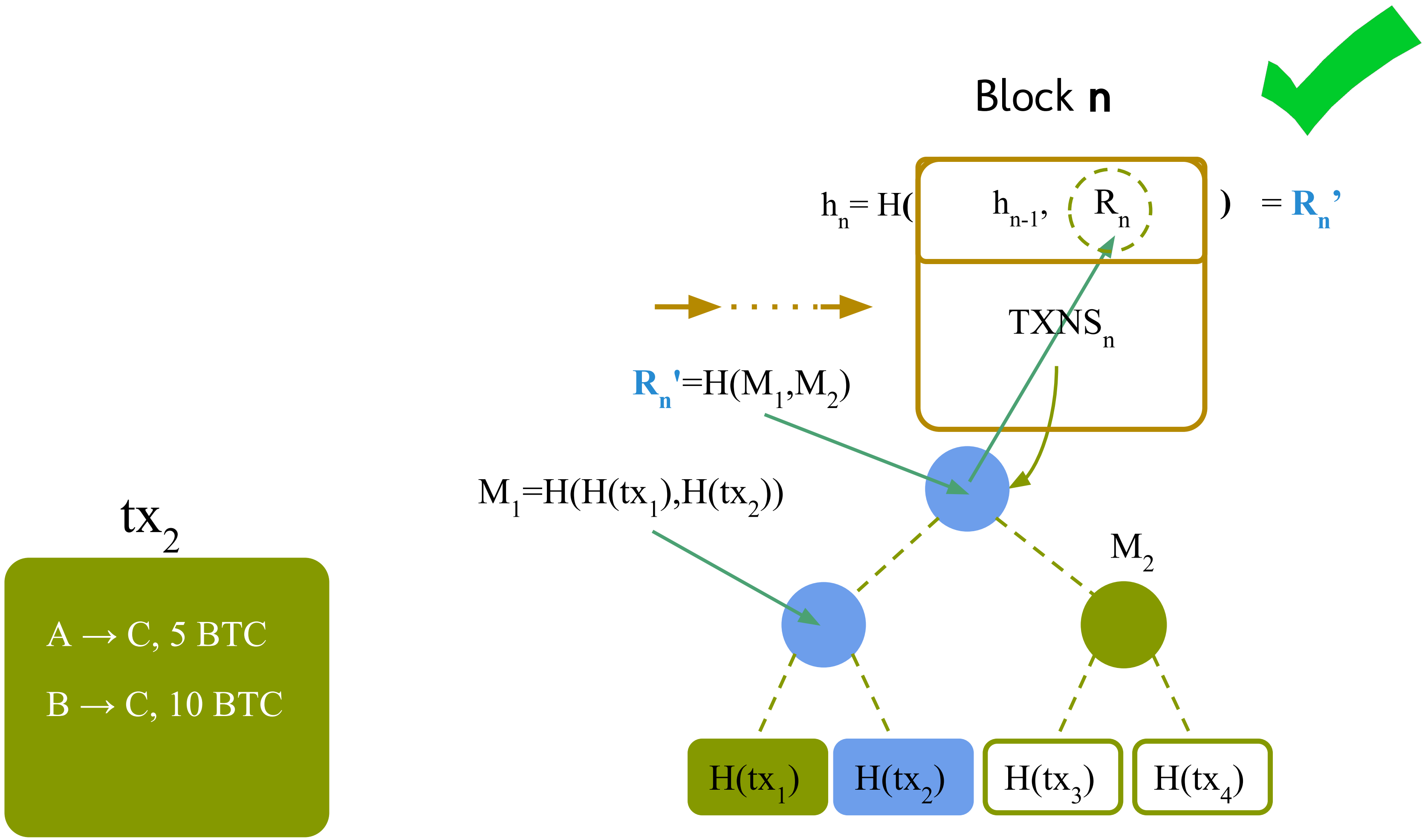
B → C, 10 BTC



Membership proof for tx_2

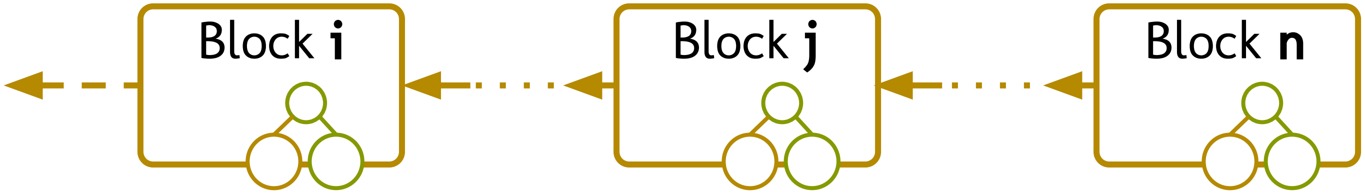
Membership Proof Success

Yay!



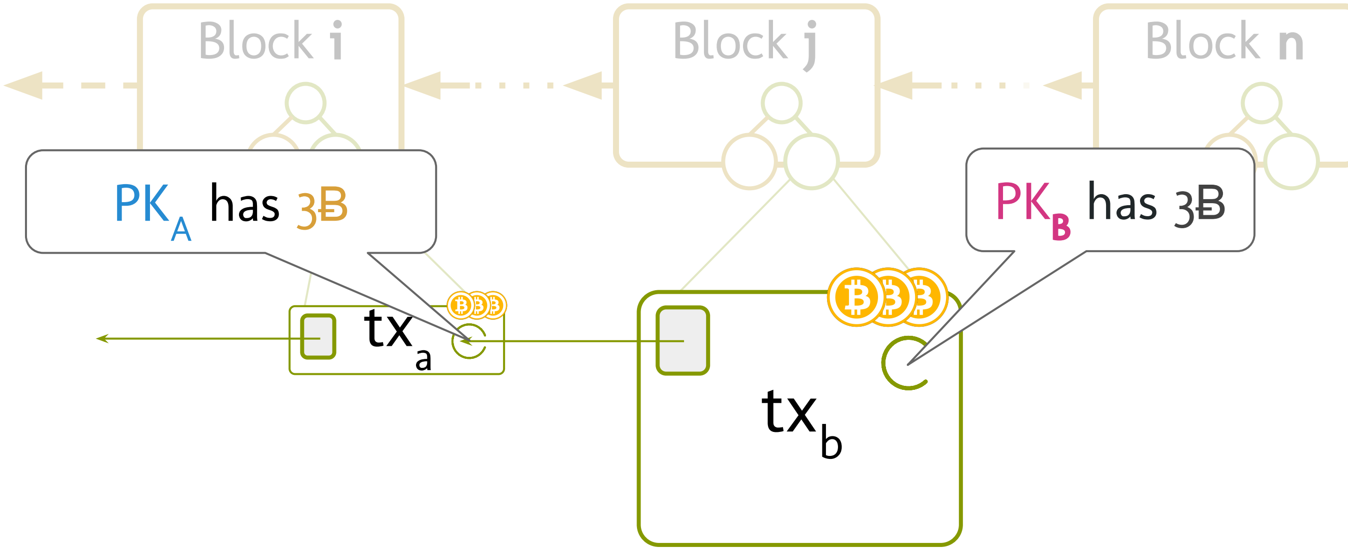
Membership proof for tx_2

Transaction Format



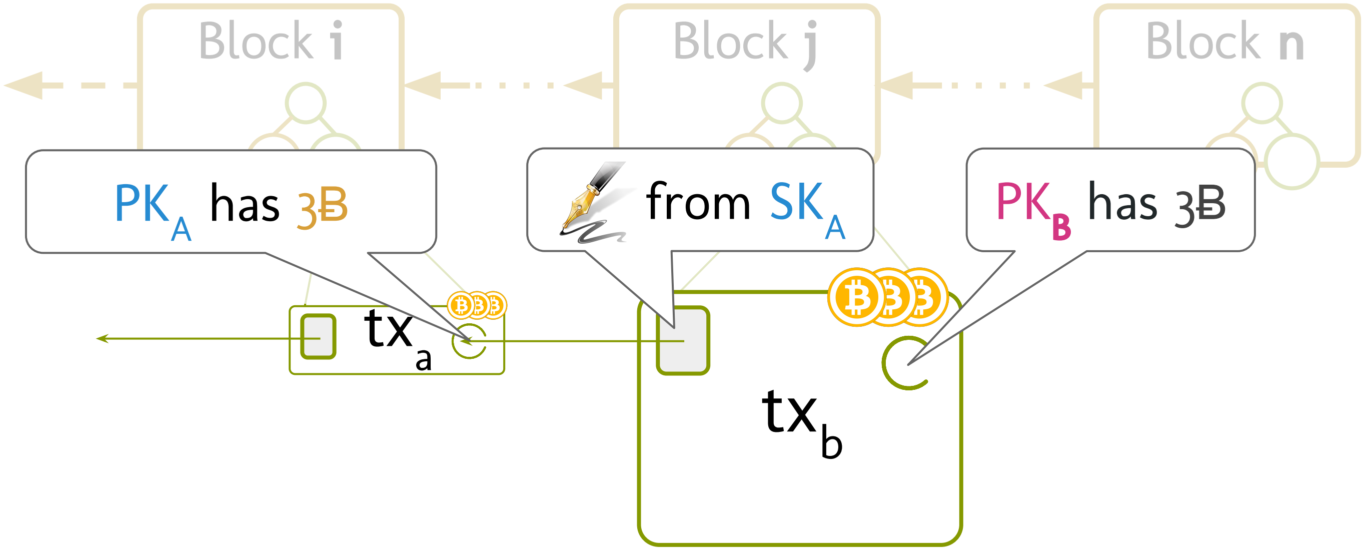
- Merkle tree of TXNs in each block

Transaction Format



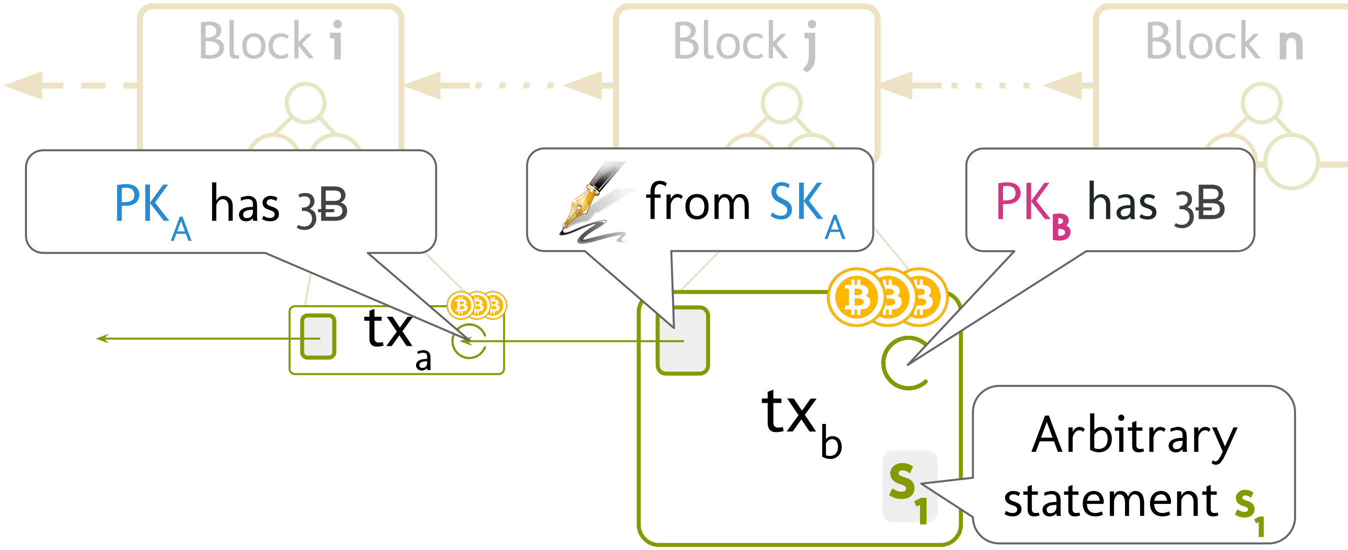
- Transactions transfer coins

Transaction Format



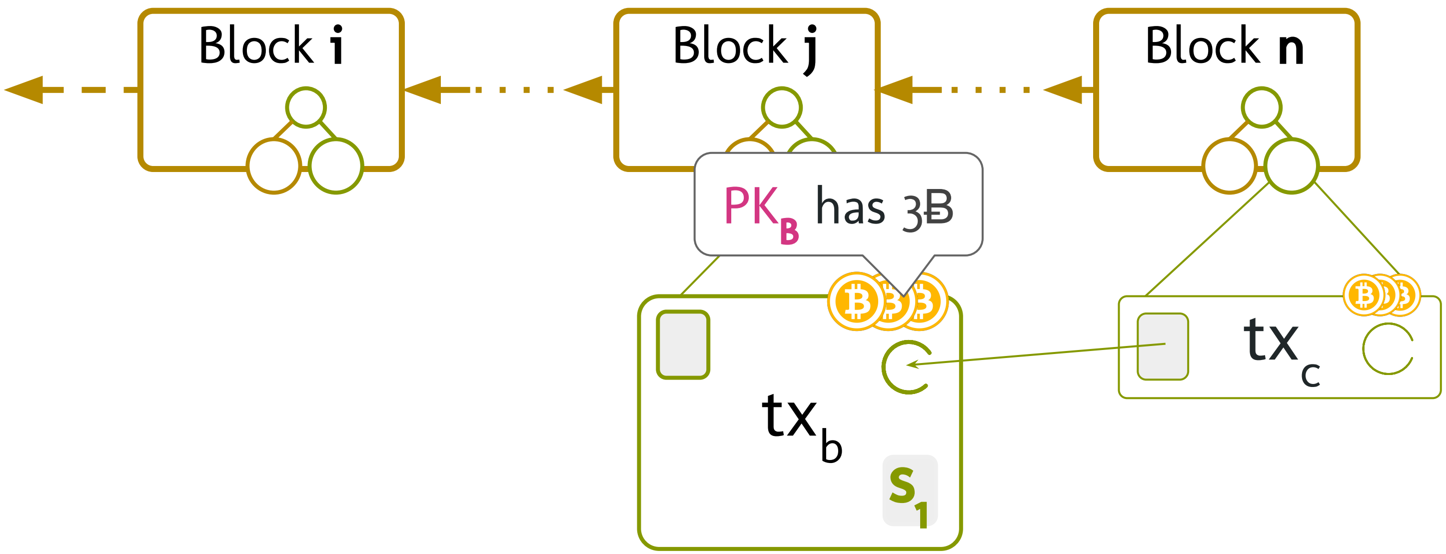
- Transactions transfer coins
- Output = # of coins and owner's PK
- Input "spends" previous output

Transaction Format



Data can be embedded in TXNs.

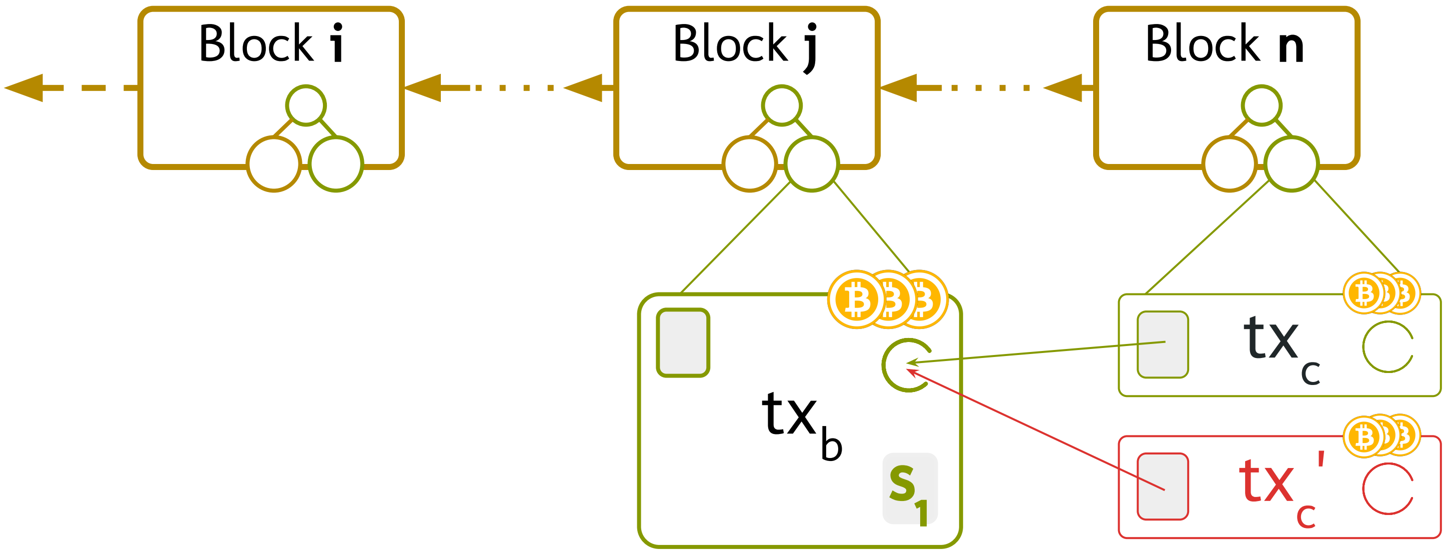
Transaction Format



Bob gives Carl 3Ƀ,

What happens if Bob tries to double spend?

Transaction Format



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

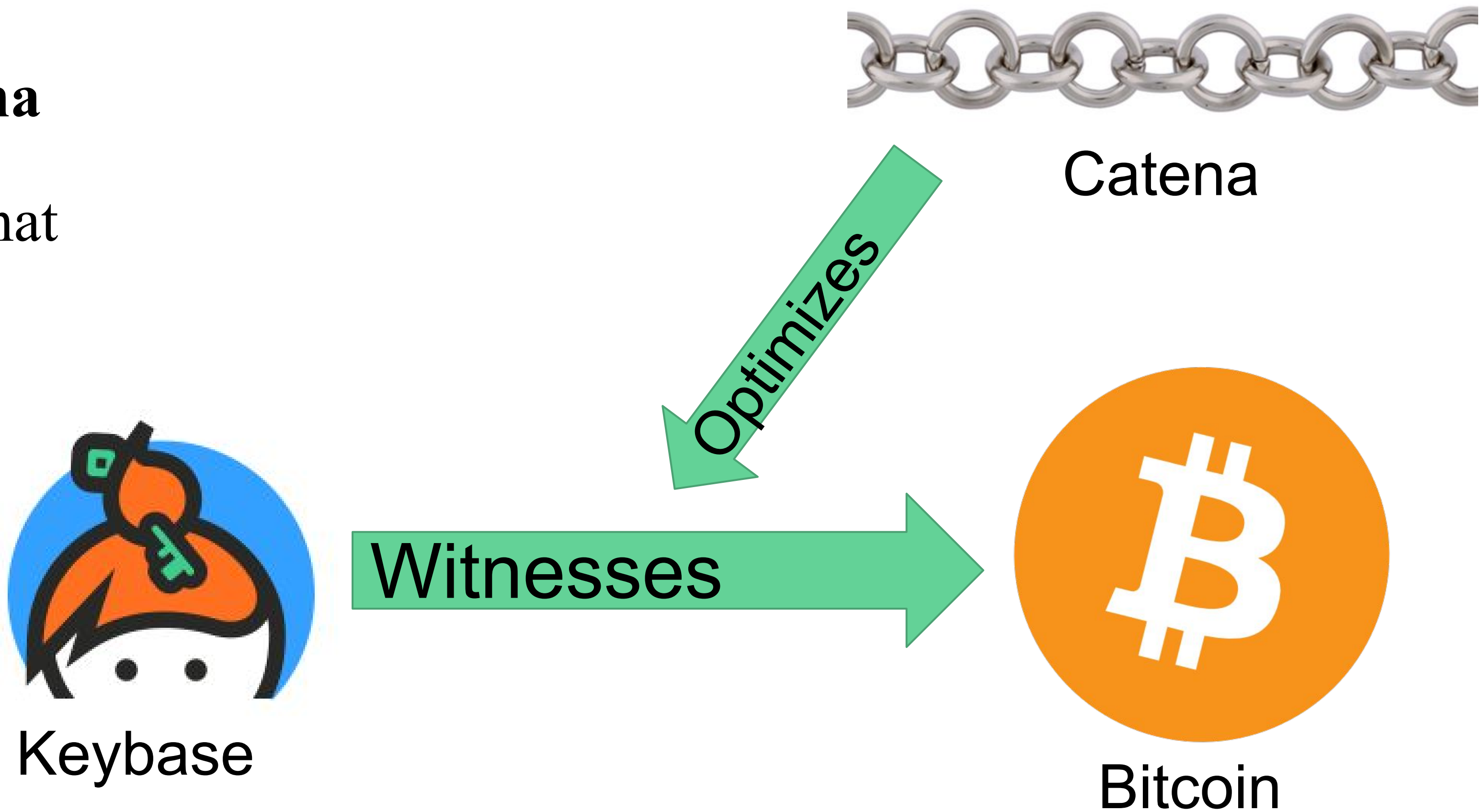
Outline

Keybase

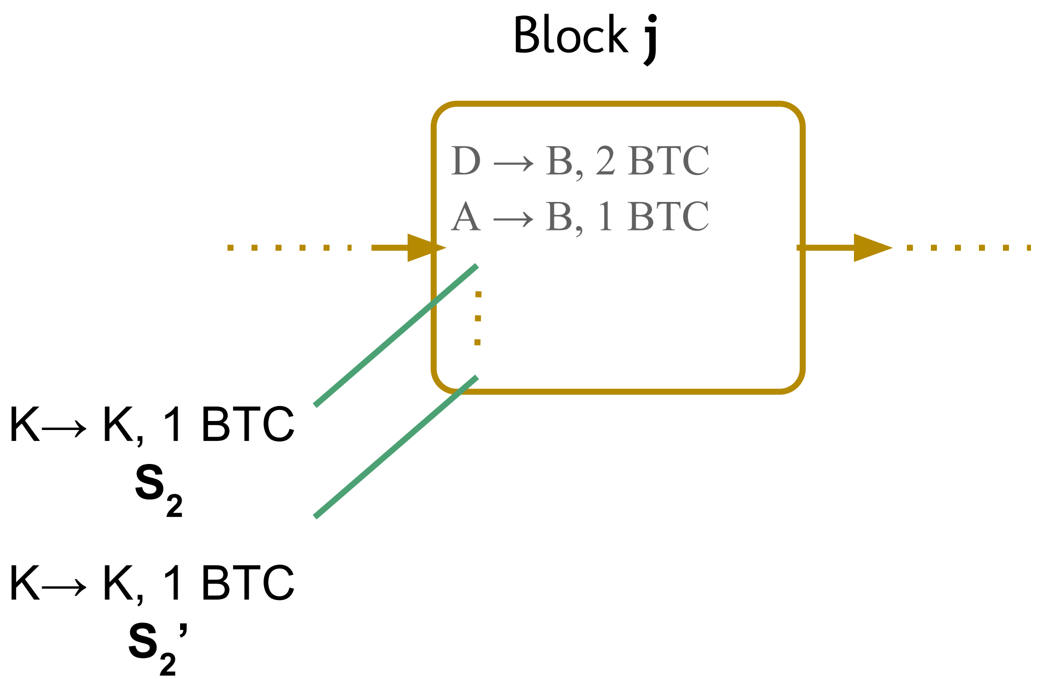
Bitcoin

Catena

Keychat



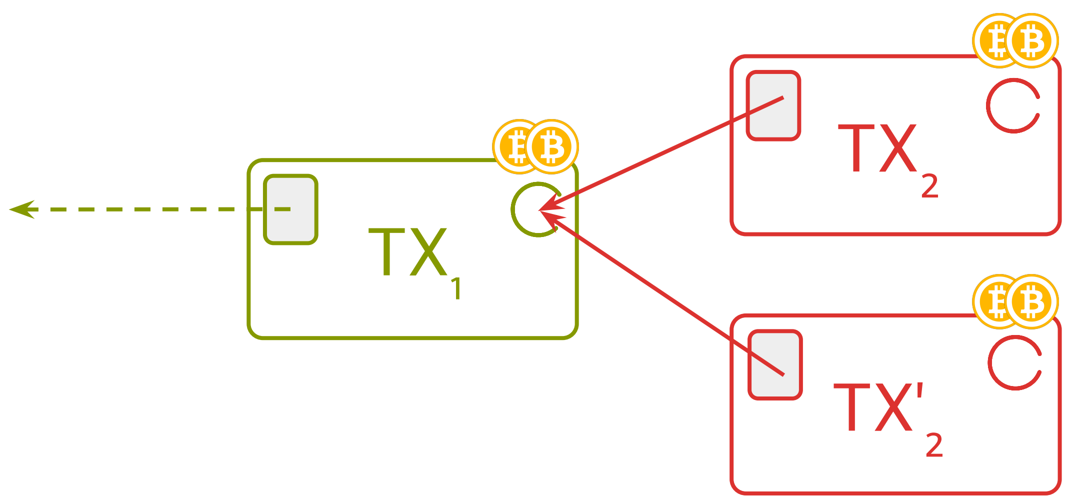
Problem with Keybase - Equivocation Within a Block?



Auditors must download the entire Bitcoin blockchain to check if any blocks contain invalid Keybase transactions

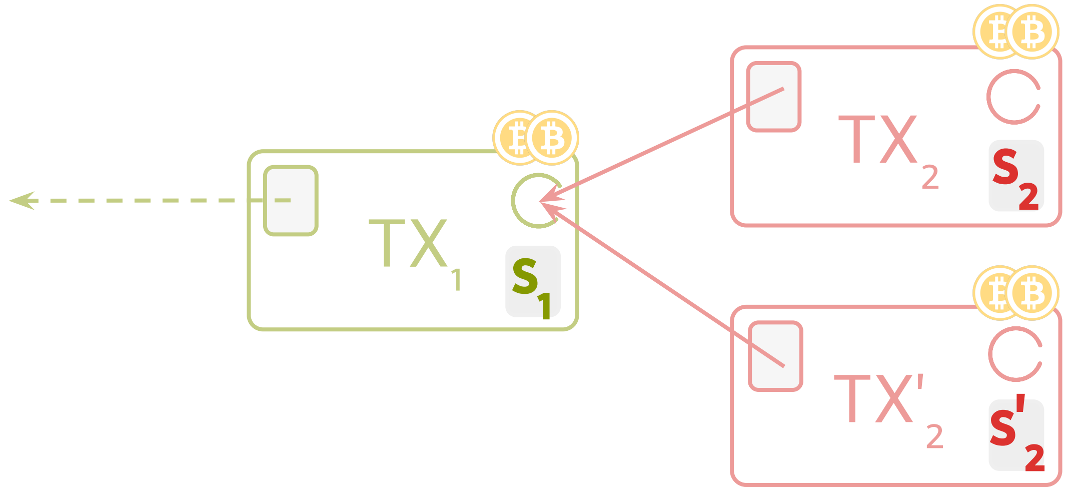
Key idea behind Catena

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

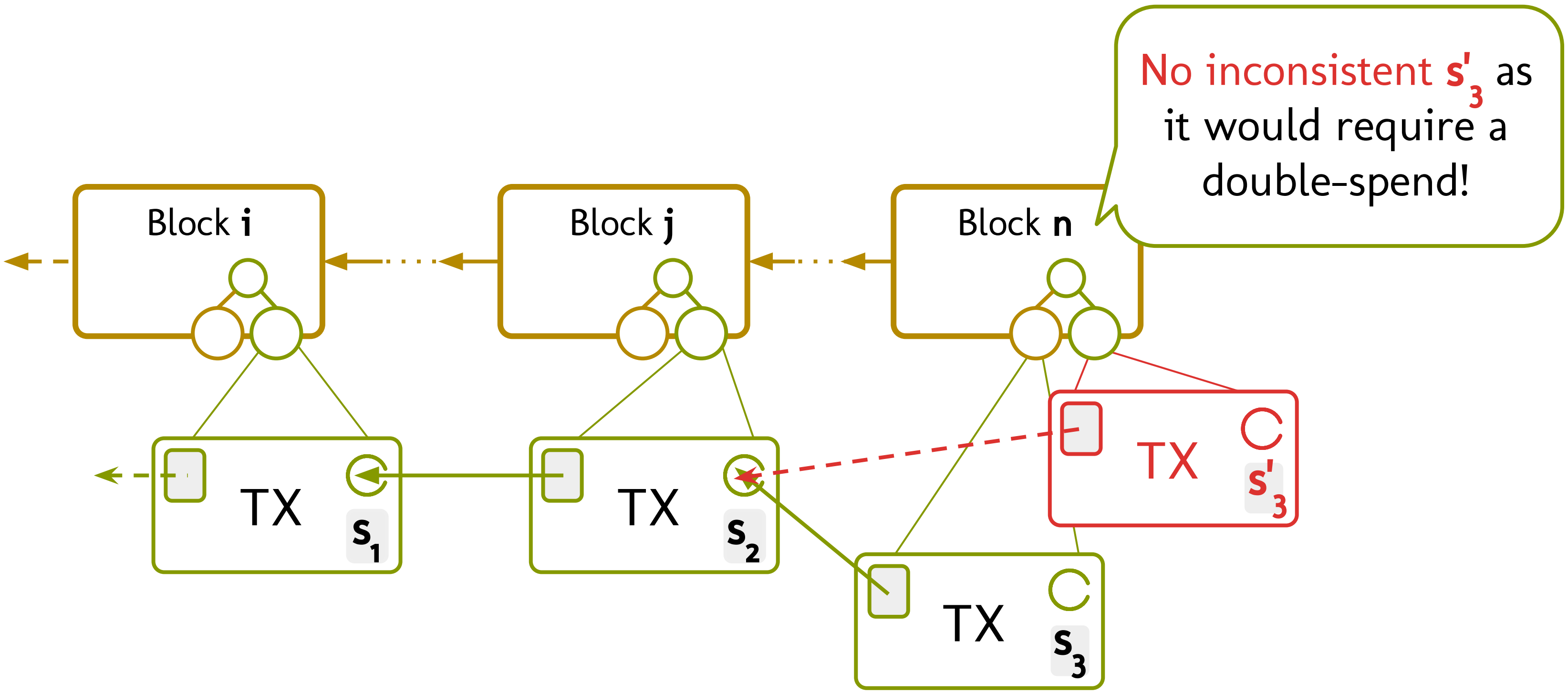


Key idea behind Catena

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

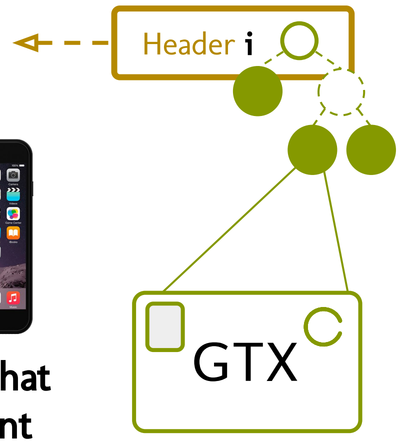


Catena

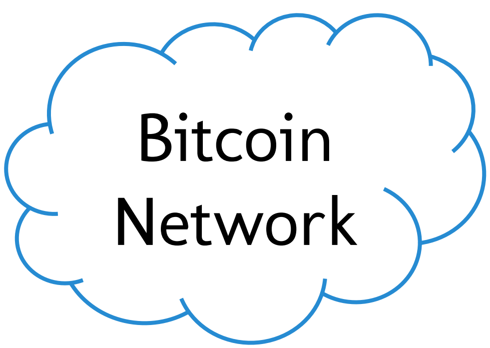


Linear chain of transactions containing statements

Efficient auditing

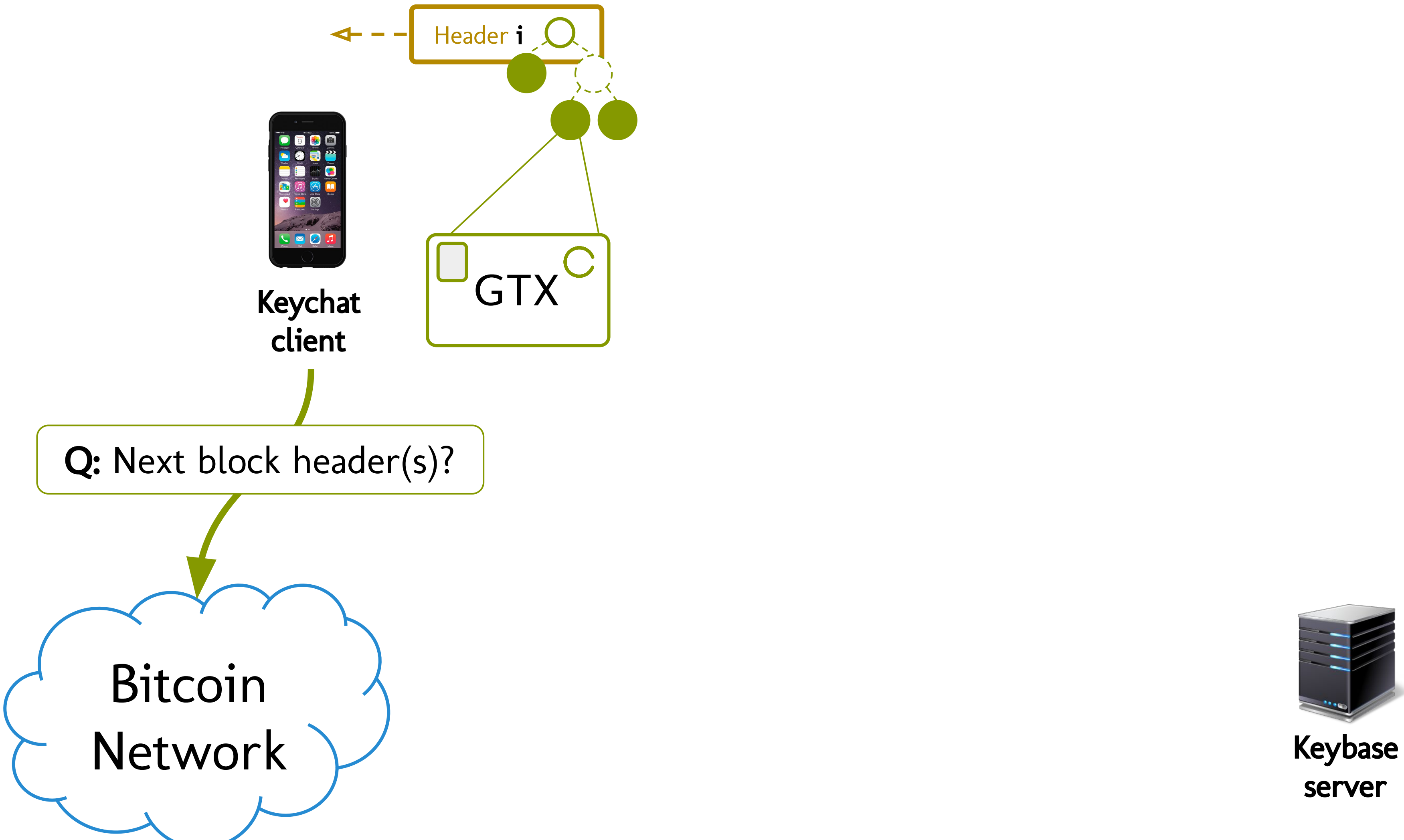


Keychat
client

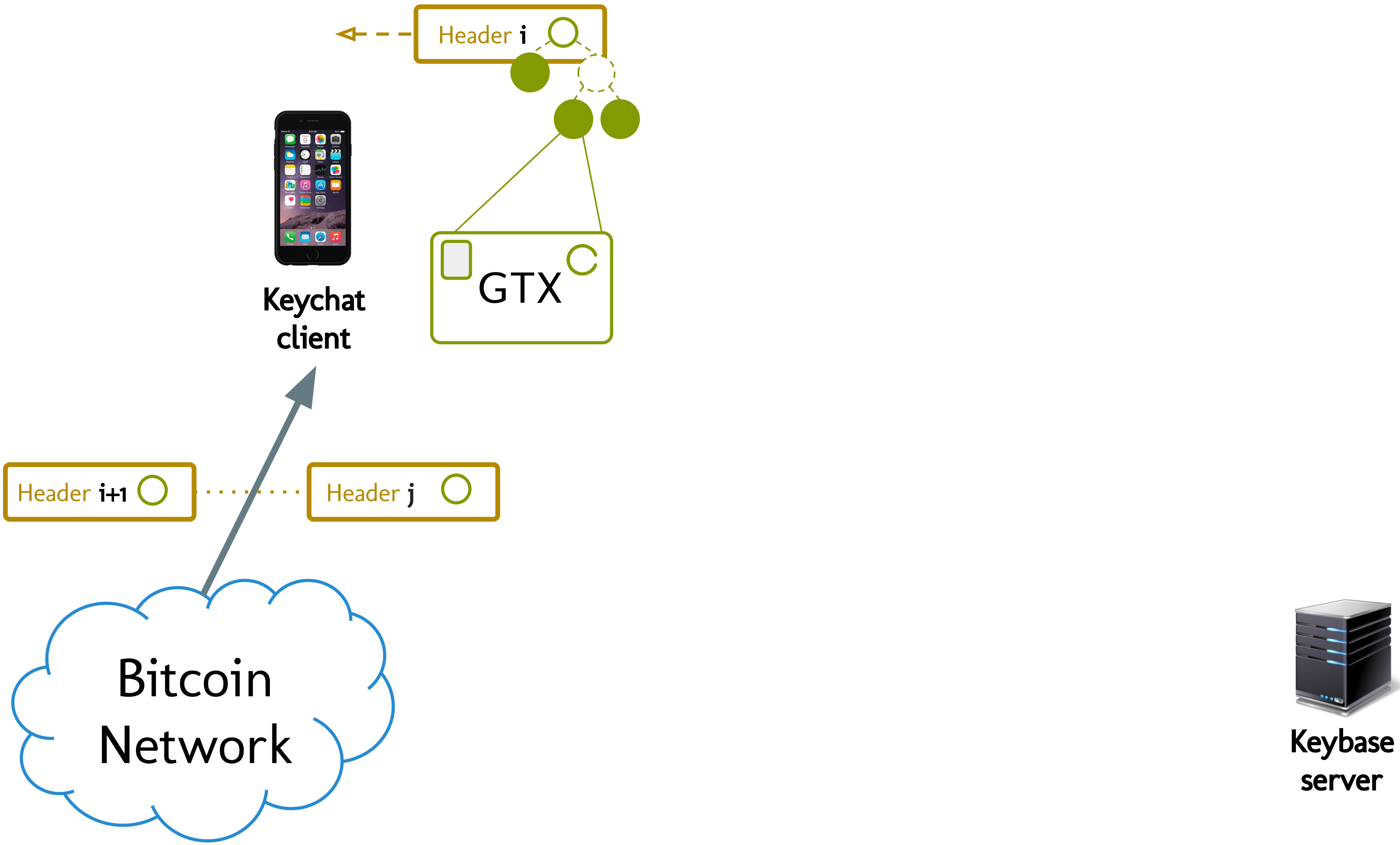


Keybase
server

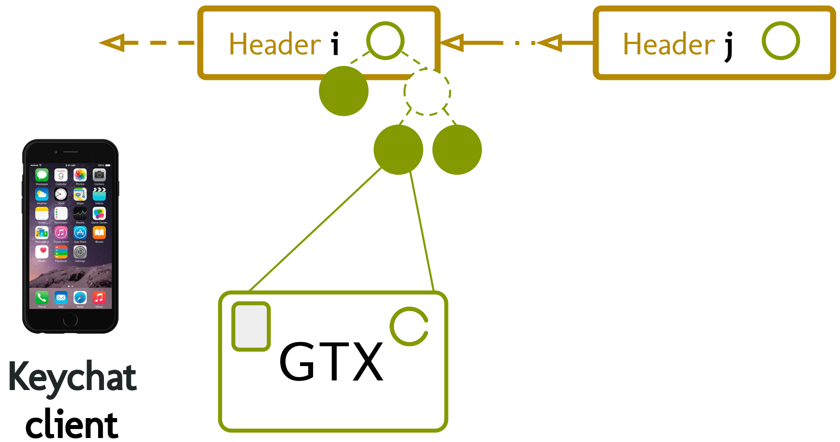
Efficient auditing



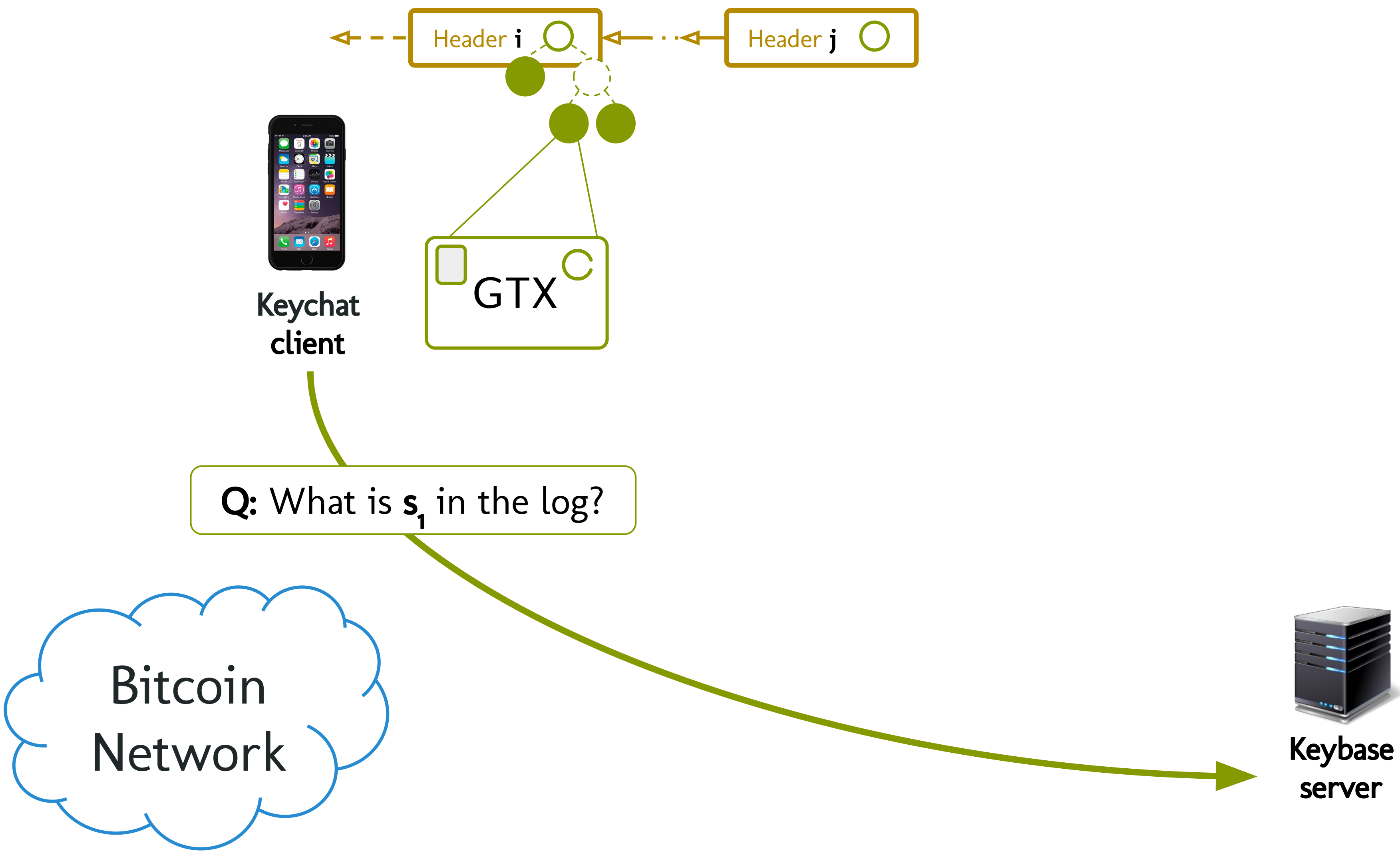
Efficient auditing



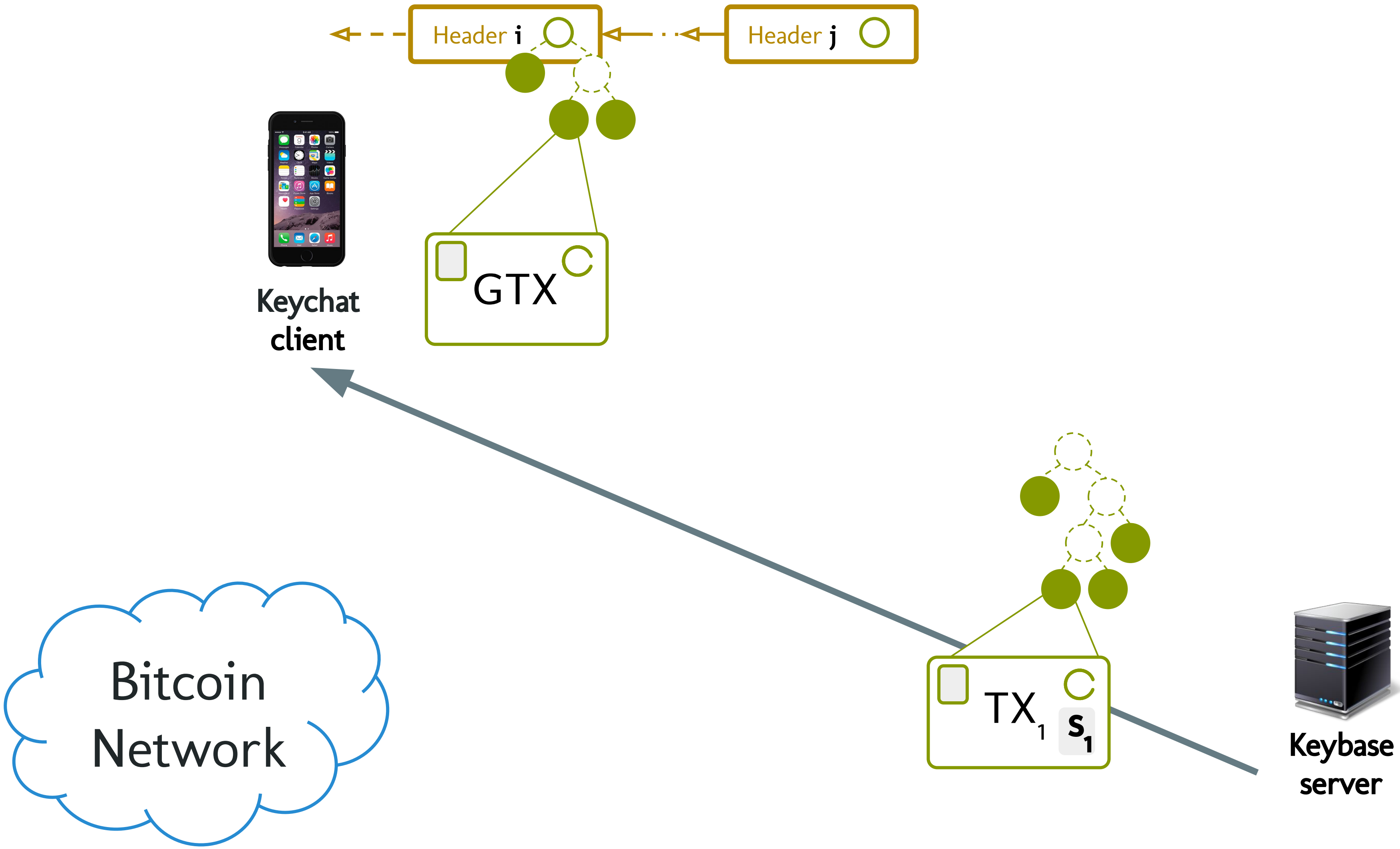
Efficient auditing



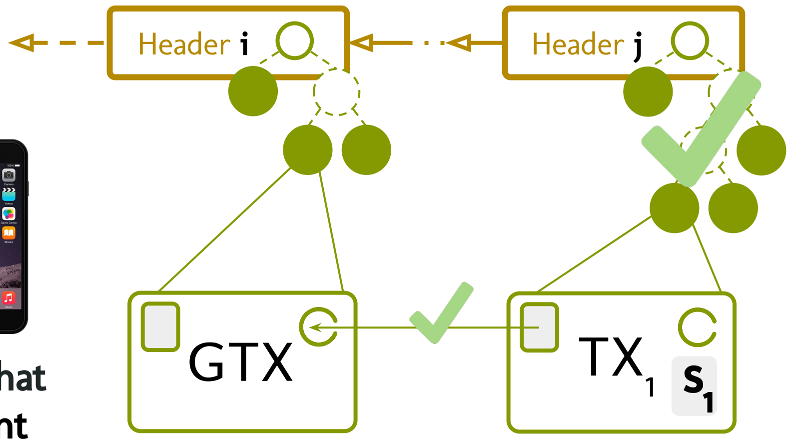
Efficient auditing



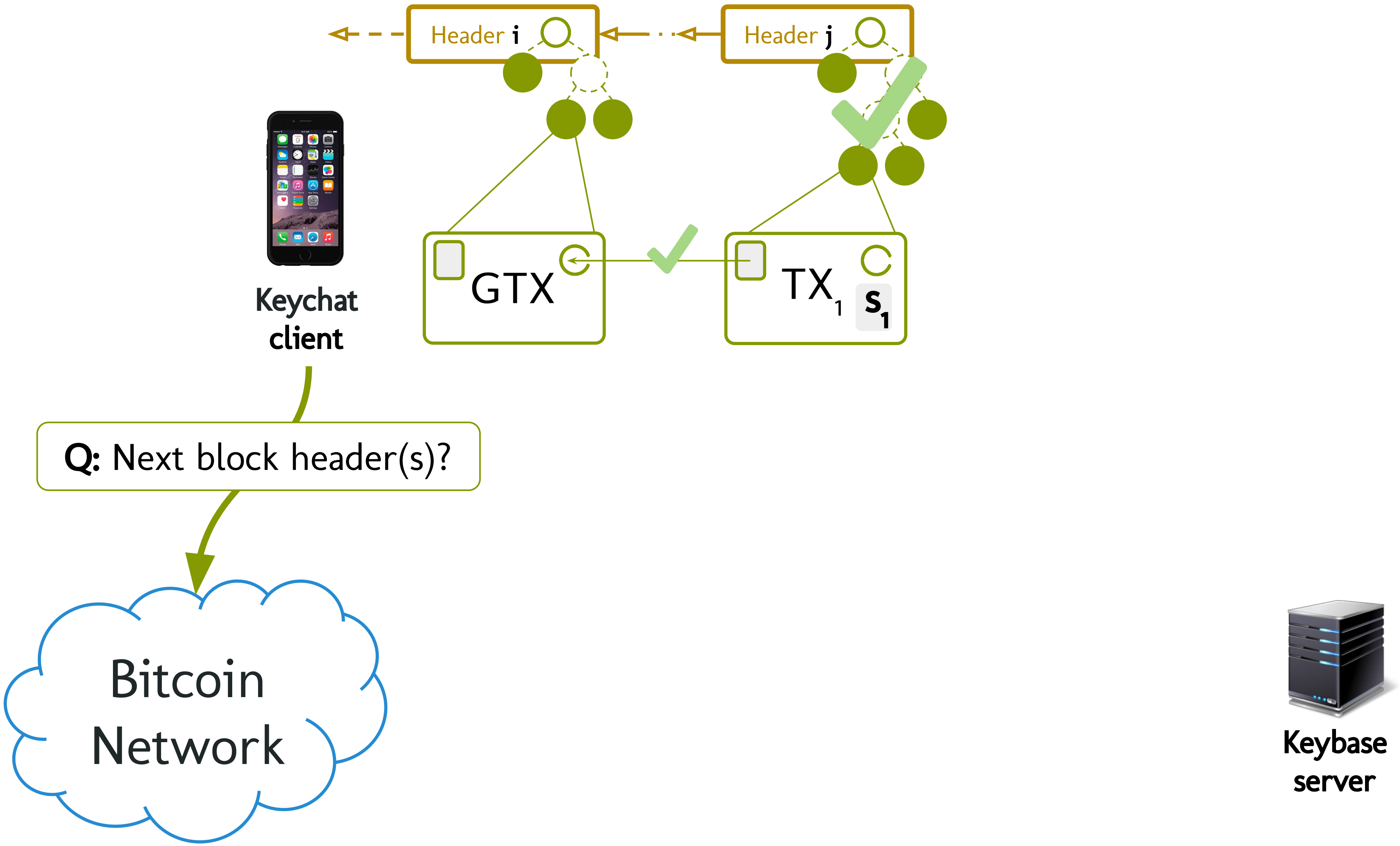
Efficient auditing



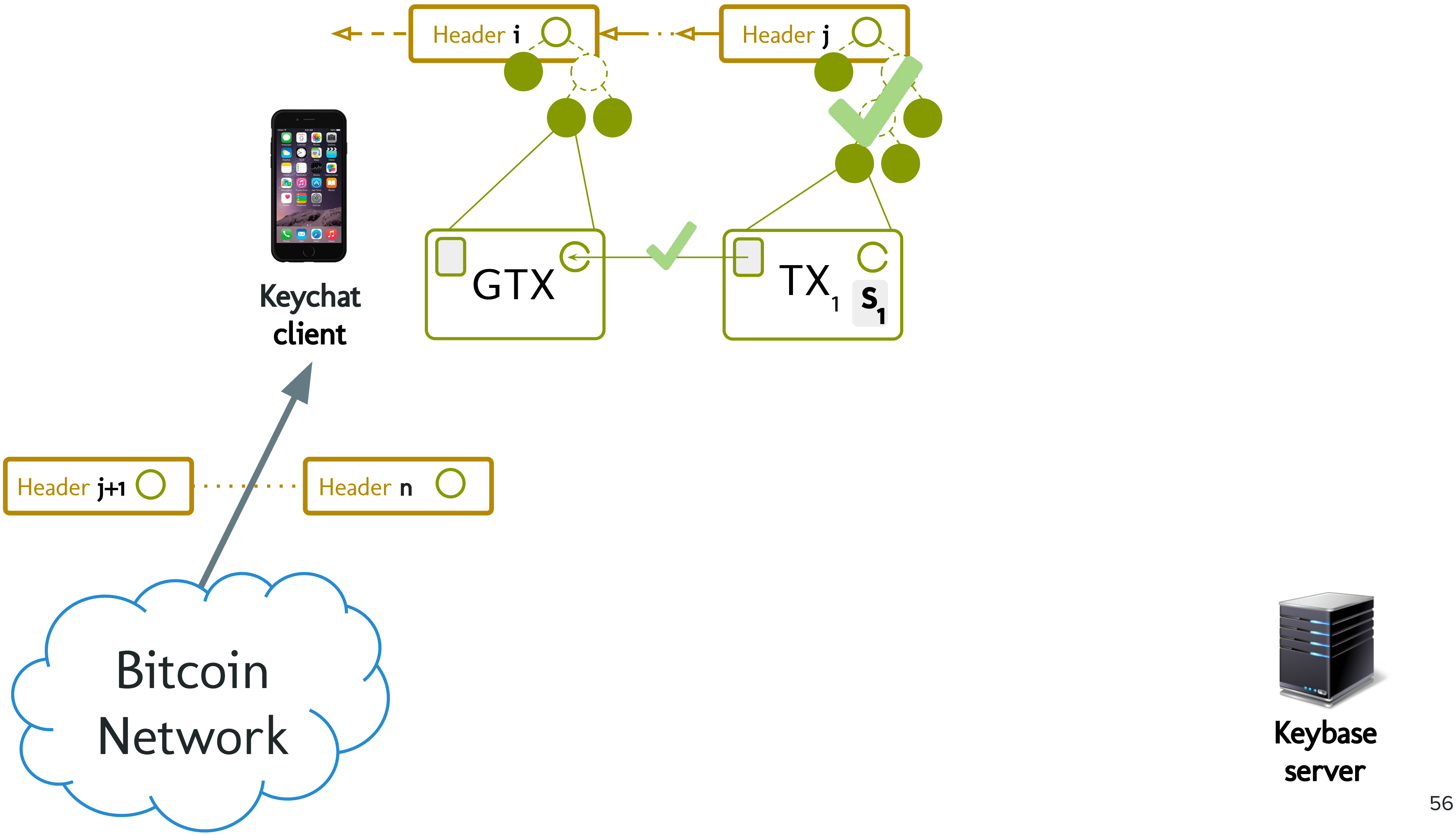
Efficient auditing



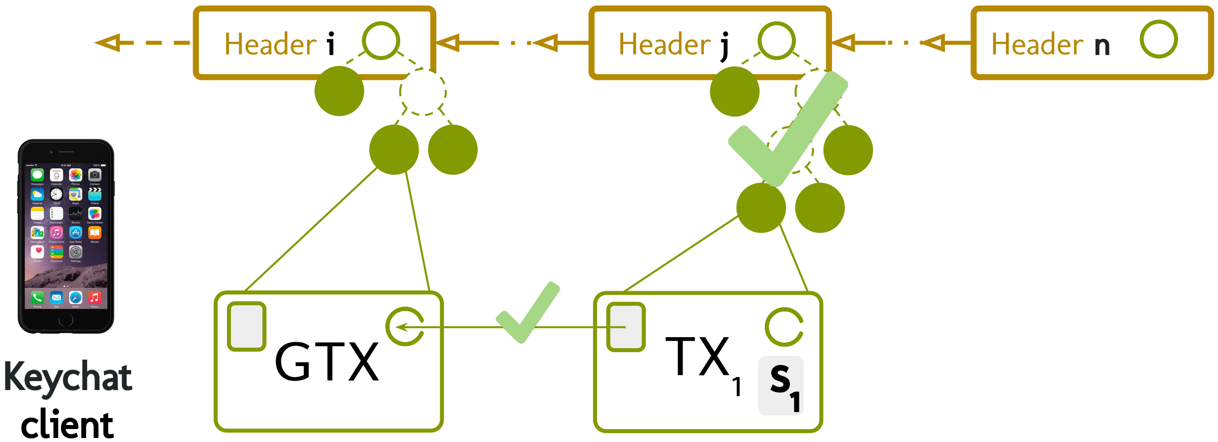
Efficient auditing



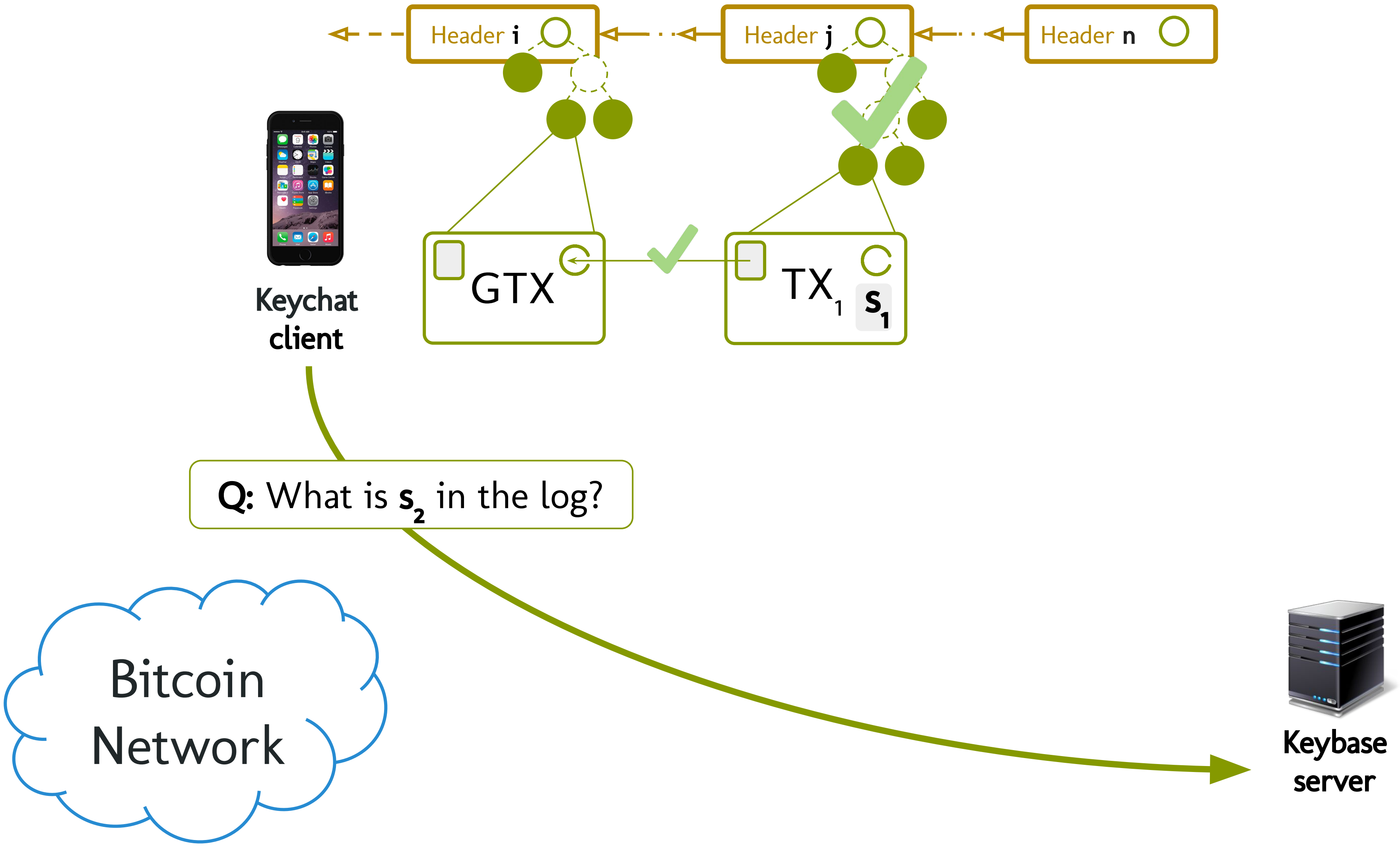
Efficient auditing



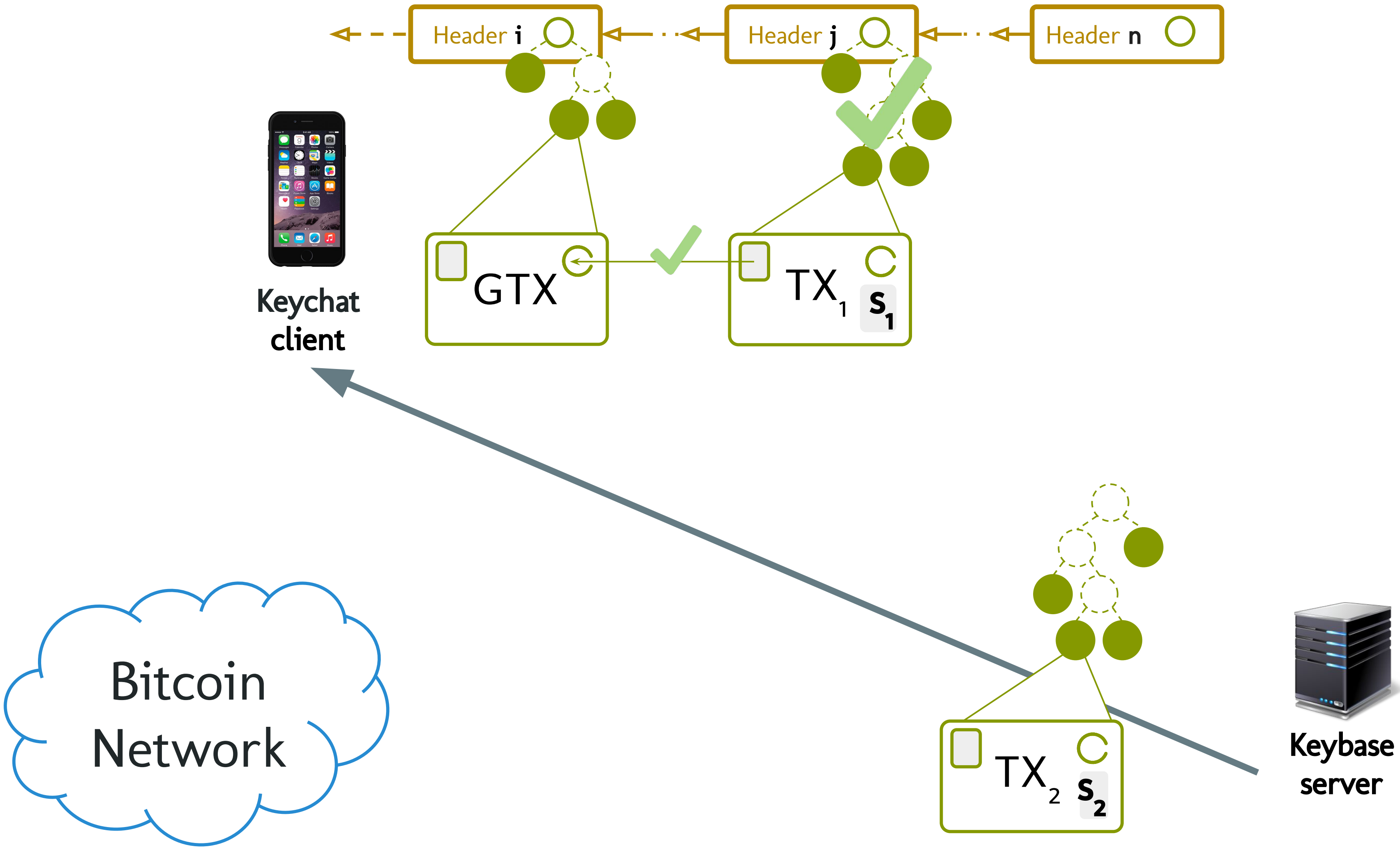
Efficient auditing



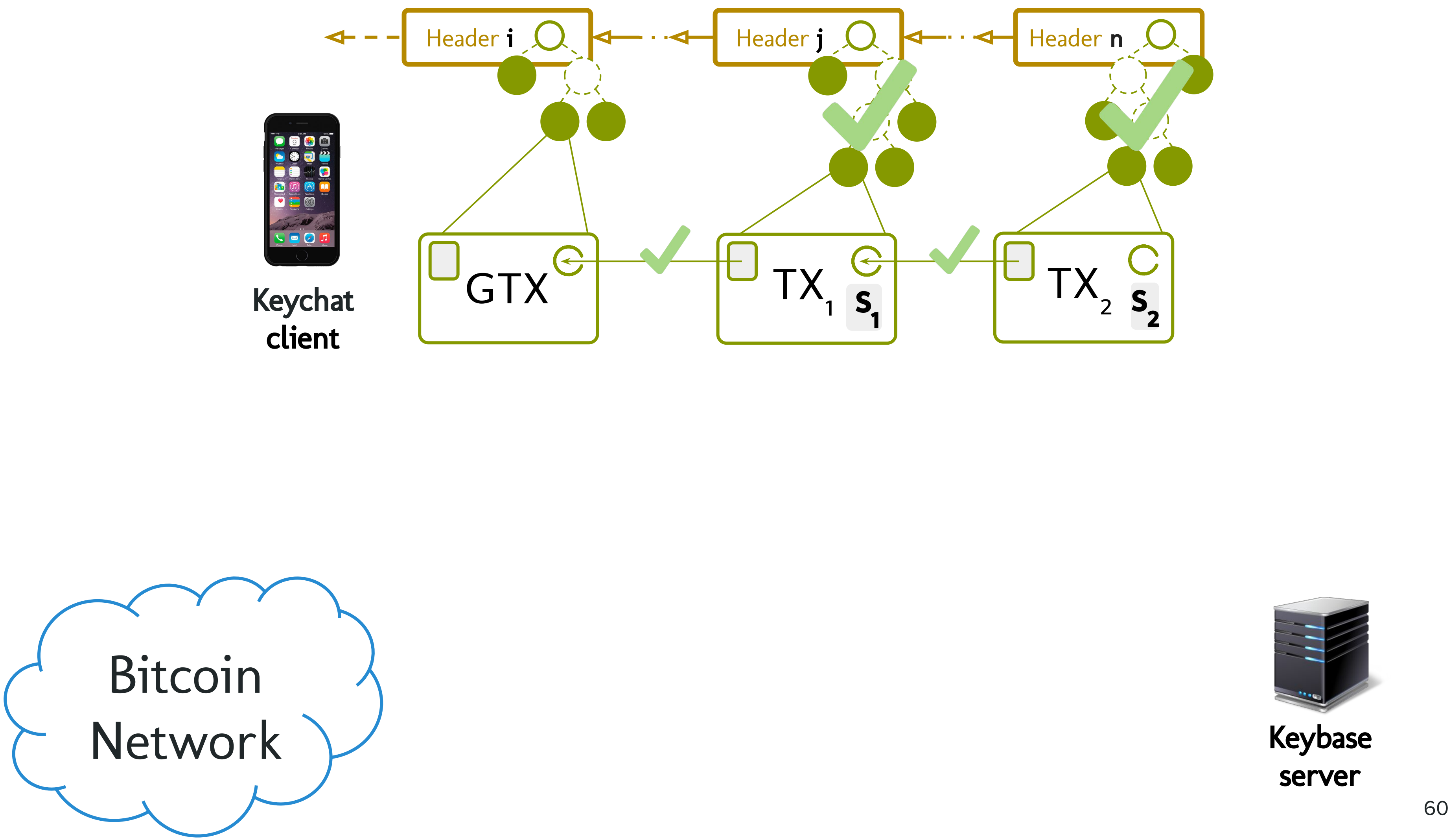
Efficient auditing



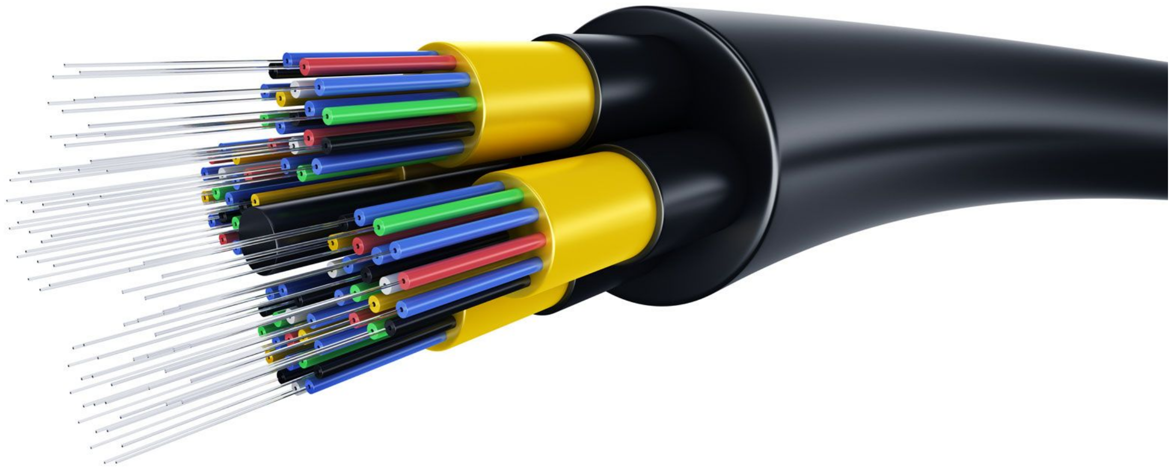
Efficient auditing



Efficient auditing



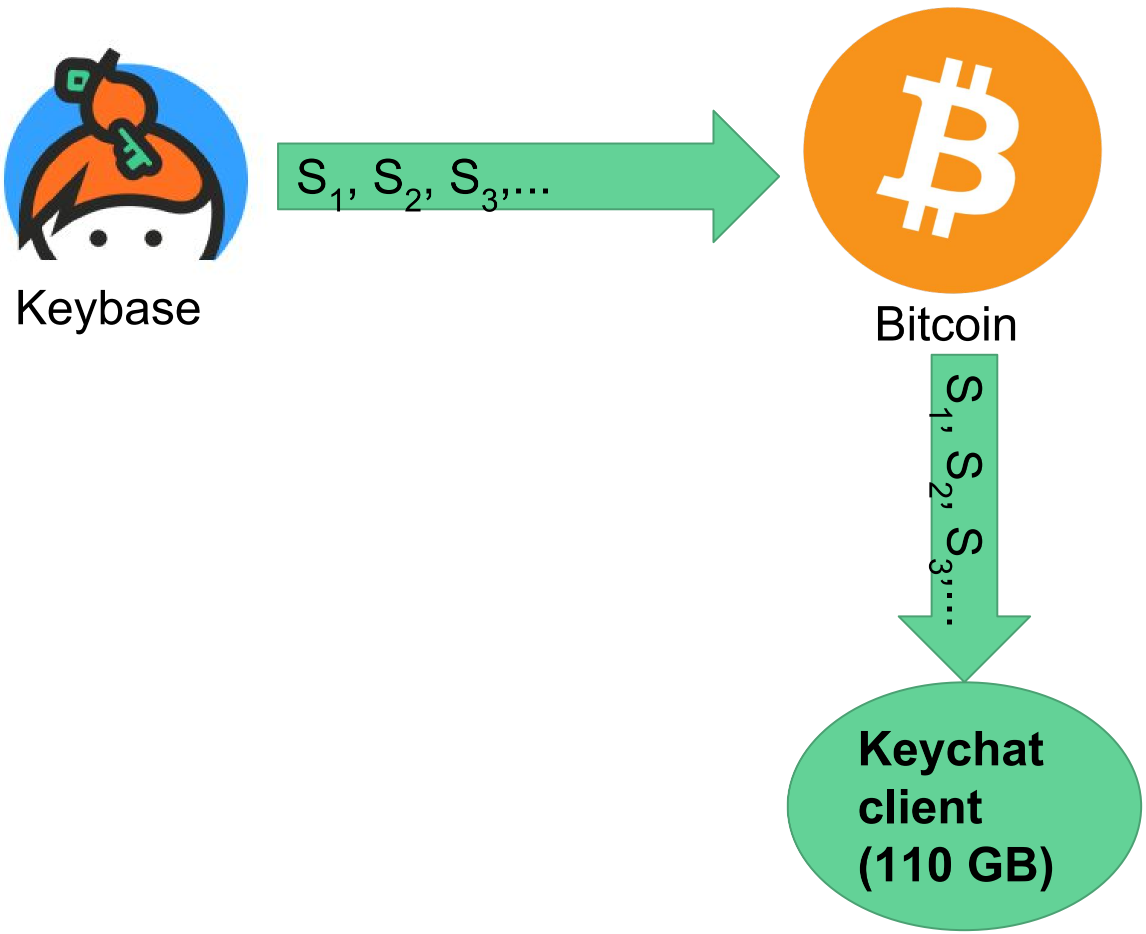
Auditing bandwidth



e.g., **460K** block headers + **10K** statements = **~41** MB
(80 bytes each) (around 600 bytes each)

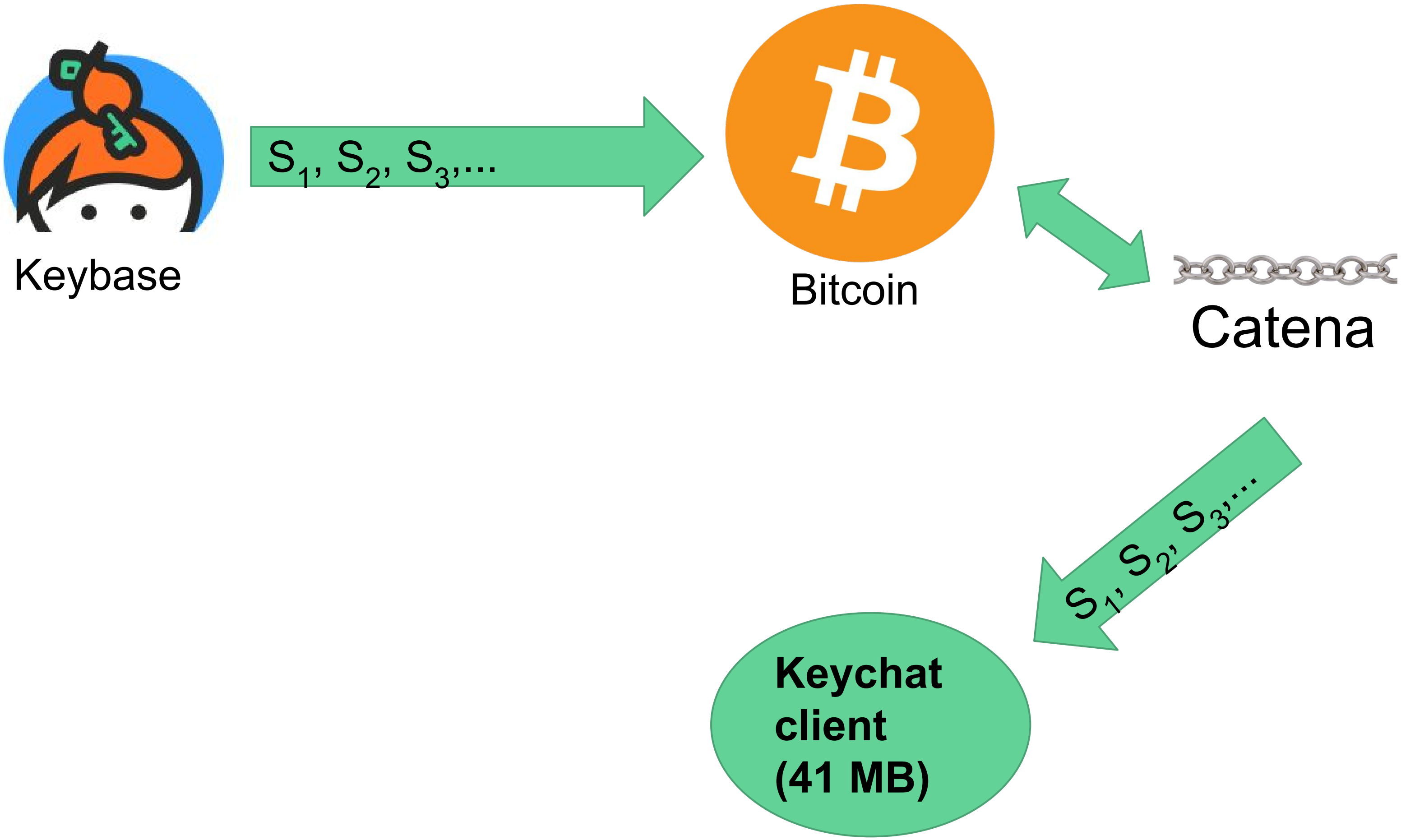
Recap

- Keybase can equivocate, so they witness the directory in Bitcoin, but inefficiently



Recap

- Keybase can equivocate, so they witness the directory in Bitcoin, but inefficiently
- Use Catena to make auditing the Keybase PKD more efficient



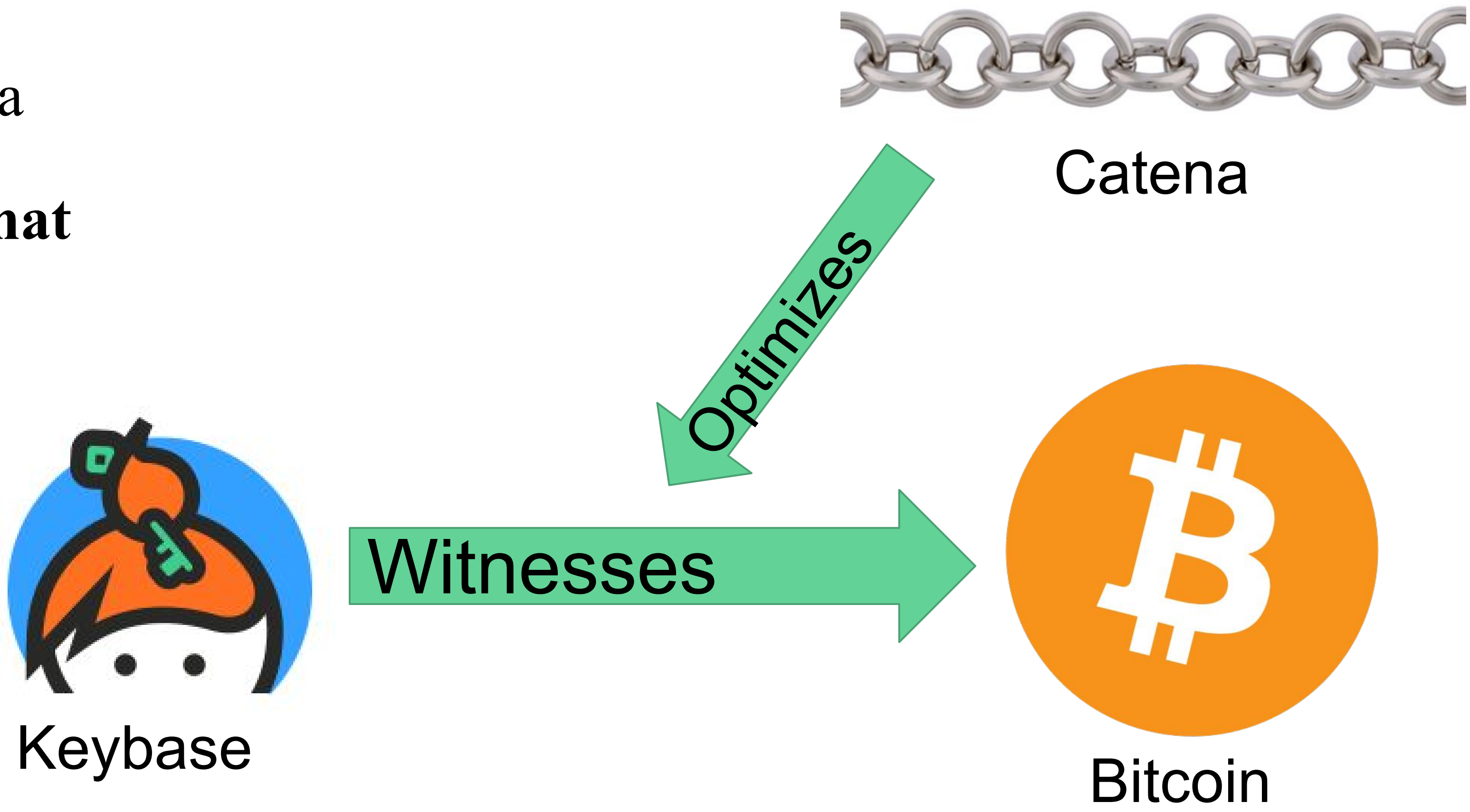
Outline

Keybase

Bitcoin

Catena

Keychat



Keychat

- Uses the Keybase PKD, so users can communicate securely without fear of public key equivocation
- Implemented using Meteor, a Javascript framework that allows KeyChat to work as both a website and an Android app

Next steps

- Implement Catena for Keybase using Java to efficiently witness the Keybase Public Key Directory in the Bitcoin blockchain
- Implement Keychat using Meteor

Acknowledgements

Thanks to our mentor Alin Tomescu for his support and guidance!

Thanks to PRIMES for this opportunity!

Thanks to our parents for their support!

Thanks to all of you for being such a great audience!

Ask us questions!