Catena: Efficient Non-equivocation via \texttt{bitcoin}

Alin Tomescu
alinush@mit.edu

Srinivas Devadas
devadas@mit.edu
What is non-equivocation?
What is non-equivocation?

- At time $i$, publishes digest $s_i$
What is non-equivocation?

- At time $i$, publishes a single digest $s_i$
What is **non-equivocation**?

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

$s_1 = \text{SHA256}(t_1)$

Public-key directory

\begin{align*}
\text{PK}_A & \quad \text{PK}_B \\
\end{align*}
What is non-equivocation?

- At time $t_2$, Alice, Bob and others "see" $s_1$, $s_2$, ...

$s_1 = \text{SHA256}(t_1)$

$s_2 = \text{SHA256}(t_2)$
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

Public-key directory

\[ s_1 = \text{SHA256}(t_1) \]

\[ s_2 = \text{SHA256}(t_2) \]
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 $t_2$, malicious **server** publishes $s_2$ and $s_2'$. 

*Public-key directory*
What is **equivocation**?

- **s₂**: Leave Alice's key intact, add fake $\text{PK}_B'$ 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 $\Rightarrow$ **MITM**
What is **equivocation**?

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

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
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.
Alice gives Bob 3Ƀ, Bitcoin miners collected 1Ƀ as a fee.
Bob gives Carol 2B, Bitcoin miners collected another B 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

Either $TX_2$ or $TX'_2$ but not both!
Moral of the story

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

Either $s_2$ or $s'_2$ but not both!
Outline

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

Block $i$

Block $j$

Block $n$

TX

TX

TX

$s_1$

$s_2$

$s_3$
Previous work

Need to **download full blocks** to find inconsistent $s'_3$.
Our work

No inconsistent $s'_3$ as it would require a double-spend!
Previous work
Our work
Outline

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

Catena log server

Server's funds
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

- $\text{TX}_2$ "spends" $\text{TX}_1$'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
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

Catena log server

GTX

Catena client

Header i

Bitcoin P2P (7000 nodes)

Catena log server
Efficient auditing

Q: Next block header(s)?

Bitcoin P2P (7000 nodes)
Efficient auditing

Catena client

GTX

Header $i$

Header $i+1$  Header $j$

80 bytes each

Bitcoin P2P (7000 nodes)

Catena log server
Efficient auditing

![Diagram showing Catena client, GTX, and CATENA servers connected through Bitcoin P2P network with 7000 nodes.](image)
Efficient auditing

Q: What is $s_i$ in the log?
Efficient auditing

Catena client

GTX

Header i

Header j

Bitcoin P2P (7000 nodes)

600 bytes

Catena log server
Efficient auditing

Bitcoin P2P (7000 nodes)
Efficient auditing

Q: Next block header(s)?

Catena client

Bitcoins P2P (7000 nodes)

Catena log server
Efficient auditing

Catena client

Header $i$

GTX

TX$_1^{s_1}$

Header $j$

Header $j+1$

Header $n$

Bitcoin P2P (7000 nodes)

Catena log server
Efficient auditing

Catena client

GTX

TX₁

Bitcoin P2P (7000 nodes)

Catena log server
Efficient auditing

Q: What is $s_2$ in the log?
Efficient auditing

![Diagram showing the process of efficient auditing in a blockchain system.](image)

**Catena client**

**Catena log server**

**Bitcoin P2P (7000 nodes)**
Efficient auditing

Bitcoin P2P (7000 nodes)
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?

bitcoin P2P

~7000 full nodes
Supports up to ~819,000 incoming connections
Catena scalability

P2P

~7000 full nodes
Supports up to ~819,000 incoming connections

Q: Next block header(s)?

Catena client 1

Catena client 2

...

Catena client 100,000?
Catena scalability

~7000 full nodes
Supports up to ~819,000 incoming connections

100,000 Catena clients ⇒ "Unintended" DDoS attack on Bitcoin.

Catena client 1
Catena client 2
...
Catena client 100,000?
Catena scalability

Catena client 1

Catena client 2

::

Catena client 100,000

Header Relay Network (HRN)
Volunteer nodes
Blockchain explorers
Facebook, Twitter, GitHub, etc.

Catena scalability

Catena client 1

Catena client 2

::

Catena client 100,000
Catena scalability

Catena client 1

Catena client 2

\[\vdots\]

Catena client 100,000

Header Relay Network (HRN)

Volunteer nodes

Blockchain explorers

Facebook, Twitter, GitHub, etc.
Catena scalability

Catena client 1

Catena client 2

::

Catena client 100,000

Header Relay Network (HRN)
Volunteer nodes
Blockchain explorers
Facebook, Twitter, GitHub, etc.
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

Need to download full blocks to find inconsistent $s'_3$

Catena

No inconsistent $s'_3$ as it would require a double-spend!
Extra slides
Bitcoin: The full picture

Payment TXs

A ➝ B, $95K

B ➝ A, $100K

Payment verification

A ➝ M, $95K

Miners

Peer-to-peer network

Customers

Merchants

A ➝ B, $95K

A ➝ M, $95K
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 \( \Rightarrow \) 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)

\[ tx_1[0] = (2B, PK) \]
Previous work

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

\[ tx_1[0] = (2B, PK) \]

\[
\text{sign}_{SK}(i, s) \]

\[
\text{sign}_{SK}(i, s') \]
Previous work

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

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

\[ \text{SIG}_{\text{SK}}(\text{tx}_1[0], \text{tx}_2) \]

\[ \text{tx}_2[0] = (2B, \text{PK}') \]

\[ \text{sk} \]

\[ \text{sign}_{\text{SK}}(i, s) \]

\[ \text{sign}_{\text{SK}}(i, s') \]

\[ \text{extractSK}() \]

Secret key \( \text{SK} \)
Previous work

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

Disincentivizes equivocation by locking Bitcoin funds under $SK$.
Does not prevent equivocation by malicious outsiders!