Catena: Efficient Non-equivocation via Bitcoin

Alin Tomescu
alinush@mit.edu

Srinivas Devadas
devadas@mit.edu
Overview

1. **What?**
2. **How?**
3. **Why?**
The equivocation problem

**Non-equivocation:** "Saying the same thing to everybody."

Various clip art images in this document © unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [https://ocw.mit.edu/fairuse](https://ocw.mit.edu/fairuse).
The equivocation problem

Equivocation: "Saying different things to different people."

Malicious service

Bob

S

S

S

S

Alice

S'
The equivocation problem

**Equivocation:** "Saying different things to different people."
The equivocation problem

Equivocation: "Saying different things to different people."

"Mom said I can go outside..."
The equivocation problem

Equivocation: "Saying different things to different people."
What is **equivocation**?

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

![Diagram showing public-key directory and key trees]

*Public-key directory*

**Alice**

**Bob**
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 $\implies$ **MITM**
What is **equivocation**?

**Bad:** "Stating different things to different people."
Catena prevents equivocation!

Damn.

Malicious service with Catena

$S_1 \quad S_2 \quad S_3$

Bob

Alice
Catena prevents equivocation!

I gotta show Alice and Bob the same $S_4$...

Malicious service with Catena
So what?

Secure software update
- Attacks on Bitcoin binaries

Secure messaging
- HTTPS
- "We assume a PKI."
So what?

Secure software update
- Attacks on Bitcoin binaries

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

"Blockchain" for X
10,000 feet view

- Bitcoin-based append-only log,
  - Generalizes to other cryptocurrencies
- ...as hard-to-fork as the Bitcoin blockchain
  - Want to fork? Do some work!
- ...but efficiently auditable
  - 600 bytes / statement (but can batch!)
  - 80 bytes / Bitcoin block
- Java implementation (3500 SLOC)
  - https://github.com/alinush/catena-java
Overview

1. What?
2. How?
   a. Bitcoin background
3. Why?
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
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

- 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 $3\text{B}$, Bitcoin miners collected $1\text{B}$ as a fee.
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 $\Rightarrow$ 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!
Overview

1. What?
2. How?
   a. Bitcoin background
   b. Previous work
3. Why?
Previous work
Previous work

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

...or trust majority of nodes to not hide statements.
Our work

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

1. What?
2. How?
   a. Bitcoin background
   b. Previous work
   c. Design
3. Why?
Starting a Catena log

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

- TX₁ "spends" GTX's output, publishes s₁
- Coins from server back to server (minus fees)
- Inconsistent s₁' would require a double-spend
Appending to a Catena log

- TX₂ "spends" TX₁'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
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
Applying 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
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
(4) No freshness guarantee
Overview

1. Catena: What?
2. Catena: How?
   a. Bitcoin background
   b. Previous work
   c. Design
   d. Efficient auditing
3. Potentially-interesting applications
Efficient auditing

Catena log server

Catena client

Header i

GTX

Bitcoin P2P
(11,500 nodes)
Efficient auditing

Q: Next block header(s)?

Bitcoin P2P
(11,500 nodes)
Efficient auditing

Catena log server

Catena client

Header $i$

GTX

Header $i+1$

Header $j$

80 bytes each

Bitcoin P2P

(11,500 nodes)
Efficient auditing

![Diagram showing Catena client and GTX with Bitcoin P2P network (11,500 nodes)]
Efficient auditing

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

Bitcoin P2P
(11,500 nodes)
Efficient auditing

CEO base

Client

Header i

Header j

GTX

TX_{1, s_1}

Bitcoin P2P

(11,500 nodes)
Efficient auditing

Q: Next block header(s)?

Bitcoin P2P (11,500 nodes)
Efficient auditing

Bitcoin P2P (11,500 nodes)
Efficient auditing

Catena log server

Catena client

Efficient auditing

Bitcoin P2P (11,500 nodes)

Catena log server

60
Efficient auditing

Q: What is $s_2$ in the log?

Bitcoin P2P
(11,500 nodes)
Efficient auditing

Bitcoin P2P (11,500 nodes)
Efficient auditing

Catena client

Bitcoin P2P (11,500 nodes)
Auditing bandwidth

e.g., 500K block headers + 10K statements = ~46 MB
(80 bytes each) (around 600 bytes each)
Overview

1. What?

2. How?
   a. Bitcoin background
   b. Previous work
   c. Design
   d. Efficient auditing
   
   e. **Scalability**

3. Why?
Catena scalability

Catena client 1

Catena client 2

Catena client 200,000?

Source: https://bitnodes.earn.com/
Catena scalability

Catena client 1

Q: Next block header(s)?

Catena client 2

Q: Next block header(s)?

Catena client 200,000?

bitcoin P2P

>11,500 full nodes
Supports up to ~1,345,500 incoming connections

Source: https://bitnodes.earn.com/
Catena scalability

200,000 Catena clients ⇒ "Accidental" DDoS attack on Bitcoin.

Catena client 1

Catena client 2

\[ \text{Catena client } 200,000? \]

\[ \text{Q: Next block header(s)?} \]

\[ \text{bitcoin P2P} \]

\[ \text{>11,500 full nodes} \]

\[ \text{Supports up to } \sim 1,345,500 \text{ incoming connections} \]

Source: https://bitnodes.earn.com/
Catena scalability

Catena client 1

Catena client 2

⋮

Catena client 200,000

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

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

Catena client 1

Catena client 2

Catena client 200,000

Q: Next block header(s)?
Catena scalability

Q: Next block header(s)?

Catena client 1

Catena client 2

\ldots

Catena client 200,000

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

bitcoin P2P
The cost of a statement

To append a statement, must issue TXN and pay fee.

**TXN size:** 235-byte

Fee as of Dec 13th, 2017: $16.24 (10 mins)
Fee as of Feb 28th, 2017: $0.78 (10 mins)

**PS:** Statements can be "batched" using Merkle trees.

Latest fees: [https://bitcoinfees.info/](https://bitcoinfees.info/)
Overview

1. What?
2. How?
3. Why?
   a. Secure software update
Secure software update

Example attack:
(1) Compromise bitcoin.org (or the network)
(2) Change the Bitcoin binary to your malicious binary
(3) Wait for people to install your malicious Bitcoin binary
(4) Steal their coins, steal their data, etc.

Example: bitcoin.org, "0.13.0 Binary Safety Warning," August 17th, 2016

Typical defense: Devs sign Bitcoin binaries with SK and protect SK.
Problem: (1) Not everyone checks sig. (2) Hard to detect stolen SK.
Solution: Publish signatures in a Catena log ⇒ Can at least detect.
Software transparency to the rescue

\[ h_1 = \text{SHA256(bitcoin-0.0.1.tar.gz)} \]

Catena log for Bitcoin binaries
Software transparency to the rescue

Catena log for Bitcoin binaries

$h_1 = \text{SHA256(bitcoin-0.0.1.tar.gz)}$

$h_2' = \text{SHA256(evilcoin-0.0.2.tar.gz)}$
Software transparency to the rescue

\[ h_1 = \text{SHA256(bitcoin-0.0.1.tar.gz)} \]
\[ h_2' = \text{SHA256(evilcoin-0.0.2.tar.gz)} \]
\[ h_2 = \text{SHA256(bitcoin-0.0.2.tar.gz)} \]
Software transparency to the rescue

Catena log for Bitcoin binaries

\[ h_1 = \text{SHA256}(\text{bitcoin-0.0.1.tar.gz}) \]
\[ h'_2 = \text{SHA256}(\text{evilcoin-0.0.2.tar.gz}) \]
\[ h_2 = \text{SHA256}(\text{bitcoin-0.0.2.tar.gz}) \]

Must double-spend to equivocate!
Overview

1. What?
2. How?
3. Why?
   a. Secure software update
   b. Secure messaging
Public-key distribution

A = \{Alice, PK_A\}, SK_A

B = \{Bob, PK_B\}, SK_B
Public-key distribution

A = {Alice, PK_A}, SK_A

B = {Bob, PK_B}, SK_B
Public-key distribution

A = \{Alice, PK_A\}, SK_A
B = \{Bob, PK_B\}, SK_B
A' = \{Alice, PK_M\}, SK_M
B' = \{Bob, PK_M\}, SK_M
B = \{Bob, PK_B\}, SK_B
Public-key distribution

A = {Alice, PK_A}, SK_A
B = {Bob, PK_B}, SK_B

A' = {Alice, PK_M}, SK_M
B' = {Bob, PK_M}, SK_M

I've been impersonated!

A = {Alice, PK_A}, SK_A
B = {Bob, PK_B}, SK_B

I've been impersonated!

A' = {Alice, PK_M}, SK_M
B' = {Bob, PK_M}, SK_M

I've been impersonated!
Public-key distribution

A = \{Alice, PK_A\}, SK_A
B = \{Bob, PK_B\}, SK_B

A' = \{Alice, PK_M\}, SK_M
B' = \{Bob, PK_M\}, SK_M

I've been impersonated!

A = \{Alice, PK_A\}, SK_A
B = \{Bob, PK_B\}, SK_B

I've been impersonated!
Public-key distribution

A = \{Alice, PK_A\}, SK_A
B = \{Bob, PK_B\}, SK_B

A' = \{Alice, PK_M\}, SK_M
B' = \{Bob, PK_M\}, SK_M

B = \{Bob, PK_B\}, SK_B

I'm not impersonated.

I'm not impersonated.
KeyChat

Idea: Store \( t_1, t_2, \ldots, t_n \) in a Catena log.

Publishing \( t'_3 \) and \( t_3 \) would require a double-spend!
Overview

1. What?
2. How?
3. Why?
   a. Secure software update
   b. Secure messaging
   c. "Blockchain" for X
I need a "blockchain" for ...


"Blockchain" = Byzantine State Machine Replication (SMR) =
= Agree on log of ops + Execute ops = Agree on final state.

Permissioned "blockchain" via:
- Your favorite Byzantine SMR algorithm
- Ethereum Smart Contract (pay fees per Ethereum op)
- Catena + 2f+1 replicas (pay fees per op batch)
I need a "blockchain" for ...


"Blockchain" = Byzantine State Machine Replication (SMR) =
= Agree on log of ops + Execute ops = Agree on final state.

Permissioned "blockchain" via:
- Your favorite Byzantine SMR algorithm
- Ethereum Smart Contract (pay fees per Ethereum op)
- Catena + 2f+1 replicas (pay fees per op batch)
- Don't need execution? Use Catena directly.

Permissionless "blockchain:" Roll your own. But proceed with caution?
Conclusions

What we did:
- Enabled applications to efficiently leverage Bitcoin's publicly-verifiable consensus
  - Download transactions selectively rather than full blockchain
  - ~46 MB instead of gigabytes of bandwidth

Why it matters:
- Secure software update schemes
- Public-key directories for HTTPS and secure messaging
- "Blockchain" for X

For more, read our paper!
Ask me questions!  

https://people.csail.mit.edu/alinush/

Previous work

No inconsistent $s'_3$ as it would require a double-spend!

Catena

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