TrInc: Small Trusted Hardware for Large Distributed Systems

Dave Levin
University of Maryland

John R. Douceur

Jacob R. Lorch
Microsoft Research

Thomas Moscibroda
Trust in distributed systems

- Selfish Participants
- Malicious Participants
Trust in distributed systems

- Selfish Participants
- Malicious Participants

Powerful tool: Equivocation

A participant “equivocates” by sending conflicting messages to others
Equivocation is common and powerful

Byz. Generals
Equivocation is common and powerful

Byz. Generals

“Advance”

“Retreat”
Equivocation is common and powerful

**Byz. Generals**

- “Advance”
- “Retreat”

**Voting**

Voting
Equivocation is common and powerful

Byz. Generals

"Advance"

"Retreat"

Voting

"Counted your vote"

Tally w/o 's vote
Equivocation is common and powerful

Byz. Generals

- “Advance”
- “Retreat”

Voting

- “Counted your vote”
- Tally w/o your vote

BitTorrent

-
Equivocation is common and powerful

Byz. Generals

“Advance”

“Retreat”

Voting

“Counted your vote”

Tally w/o ‘s vote

BitTorrent

“I have piece 5”

“I don’t have piece 5”
Equivocation is common and powerful

Byz. Generals

“Advance”

“Retreat”

Voting

“Counted your vote”

Tally w/o ‘s vote

BitTorrent

“I have piece 5”

“I don’t have piece 5”

Leader election

Online games

Digital cash

Auctions

Trusted logs

soBGP

Version control

DHTs
Equivocation is common and powerful

Byz. Generals

- $f$ malicious users
- If completely untrusted, $3f+1$ users needed for consensus [Lamport et al, 1982]
Equivocation is common and powerful

Byz. Generals

- If users cannot equivocate, only $2f+1$ users are needed [Chun et al, 2007]
- If completely untrusted, $3f+1$ users needed for consensus [Lamport et al, 1982]
- $f$ malicious users

Byzantine Generals:
- "Advance"
- "Retreat"
Enter Trusted Hardware

Equivocation can be rendered **impossible** with **trusted hardware**

- New design space
- All participants have a trusted component
Enter Trusted Hardware

Equivocation can be rendered impossible with trusted hardware

- New design space
- All participants have a trusted component
Enter Trusted Hardware

Equivocation can be rendered **impossible** with **trusted hardware**

- New design space
- All participants have a trusted component
Enter Trusted Hardware

Equivocation can be rendered **impossible** with **trusted hardware**

- New design space
  - All participants have a trusted component

- To be practical, the hardware must be small
  - Ubiquity via low cost
  - Tamper-resilient
  - Easier to verify a small TCB
Contributions

1. TrInc – A new, practical primitive for eliminating equivocation

2. Applications of TrInc

3. Implementation in currently available hardware
Contributions

1. **TrInc** – A new, practical primitive for eliminating equivocation

2. Applications of TrInc

3. Implementation in currently available hardware
Motivating question

What is the **minimal abstraction** needed to make equivocation impossible?
What is the **minimal abstraction** needed to make equivocation impossible?

A counter and a key are enough
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attestations bind data to counters
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attest(36, data) → 34

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attest(36 \text{ data}) \rightarrow 34

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc: Trusted Incrementerer

1. Monotonically increasing counter
2. Key for signing attestations

Attest(36 data) → 36

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attest(36, data) → 36

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc: Trusted Incrementer

1. Monotonically increasing counter
2. Key for signing attestations

Attest(36, data) → 36 → <34, 36, data>_K

Attestations bind data to counters

“Bind this data to counter value 36”
TrInc Attestations

\[ \langle 34, 36, \text{data} \rangle_K \]

\[ \langle 36, 36, \text{nonce} \rangle_K \]
TrInc Attestations

Advance attestation
• Can only move to a state once
• “data” is forever bound to 36
• There was nothing bound to 35

Status attestation
• “What is your current counter?”
  • Nonces assure freshness
• There is nothing beyond 36 (yet)

< 34, 36, data >_K
< 36, 36, nonce >_K
Multiple counters

- Need multiple trusted counters
  - Systems running concurrently
  - Some systems benefit from more counters
Multiple counters

- Need multiple trusted counters
  - Systems running concurrently
  - Some systems benefit from more counters

**Trinket**

- Hardware that contains $\geq 1$ counter is a **Trinket**
  - Allocates and frees counters
  - Establishes session keys
TrInc is practical

- Trusted Platform Module (TPM) is ubiquitous
  - Tamper-resistance
  - Counters (currently 4)
  - Crypto
  - Small amount of storage
- It just lacks the right interface

TPM Penetration
Source: IDC 2006

Desktop PCs
Mobile PCs
x86 Servers
Contributions

1. **TrInc** – A new, practical primitive for eliminating equivocation

2. Applications of TrInc

3. Implementation in currently available hardware
Contributions

1. **TrInc** – A new, practical primitive for eliminating equivocation

2. **Applications** of **TrInc**

3. **Implementation** in currently available hardware
What can TrInc do?

- Trusted append-only logs
- Prevent under-reporting in BitTorrent
- Reduces communication in PeerReview
- BFT with fewer nodes and messages
- Ensure fresh data in DHTs
- Prevent Sybil attacks
What can TrInc do?

- Trusted append-only logs
- Prevent under-reporting in BitTorrent
- Reduces communication in PeerReview
- BFT with fewer nodes and messages
- Ensure fresh data in DHTs
- Prevent Sybil attacks
What can TrInc do?

• Trusted append-only logs

• Prevent under-reporting in BitTorrent

• Reduces communication in PeerReview

• BFT with fewer nodes and messages

• Ensure fresh data in DHTs

• Prevent Sybil attacks
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

< 3,8, purple >
< 8,9, green >
< 9,10, red >
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

< 3,8, >
< 8,9, >
< 9,10, >
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

< 3,8, □ >
< 8,9, □ >
< 9,10, □ >
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

< 3,8, >
< 8,9, >
< 9,10, >
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

<10,11>

<3,8>
<8,9>
<9,10>
Implementing a trusted log in TrInc

Append(\text{data}): Bind new data to the end of the log

Lookup(\text{sequence num}): No equivocating on what is or is not stored

Untrusted storage

\begin{itemize}
\item \langle 3,8, \rangle
\item \langle 8,9, \rangle
\item \langle 9,10, \rangle
\item \langle 10,11, \rangle
\end{itemize}
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage:  
\[
\begin{align*}
&< 3, 8, \textcolor{Purple}{10} > \\
&< 8, 9, \textcolor{Green}{10} > \\
&< 9, 10, \textcolor{Red}{11} > \\
&< 10, 11, \textcolor{Yellow}{11} >
\end{align*}
\]
Implementing a trusted log in TrInc

**Append**(data):
Bind new data to the end of the log

**Lookup**(sequence num):
No equivocating on what is or is not stored

Untrusted storage

- `<3,8,>`
- `<8,9,>`
- `<9,10,>`
- `<10,11,>`
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

< 3,8, >
< 8,9, >
< 9,10, >
< 10,11 >
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

<3,8,1><8,9,2><9,10,3><10,11,4>
Implementing a trusted log in TrInc

Append(data):
Bind new data to the end of the log

Lookup(sequence num):
No equivocating on what is or is not stored

Untrusted storage

<3,8, purple>
<8,9, green>
<9,10, red>
<10,11, yellow>

<9,10, red> Fast lookups
Few hardware accesses
TrInc-A2M

- Attested Append-only Memory (A2M)
  - Stores logs in trusted storage
  - Accesses trusted storage for all methods

- A2M shown to solve
  - Byzantine fault tolerance using fewer nodes
  - SUNDR file system
  - Quorum/Update protocol

- By construction, TrInc solves these systems, too
What can TrInc do?

- Trusted append-only logs
- Prevent under-reporting in BitTorrent
- Reduces communication in PeerReview
- BFT with fewer nodes and messages
- Ensure fresh data in DHTs
- Prevent Sybil attacks
What can TrInc do?

- Trusted append-only logs
- Prevent under-reporting in BitTorrent
- Reduces communication in PeerReview
- BFT with fewer nodes and messages
- Ensure fresh data in DHTs
- Prevent Sybil attacks
BitTorrent primer
BitTorrent primer

Fast, users share the work
BitTorrent primer

Fast, users share the work
BitTorrent primer

Fast, users share the work

Does not have piece 2
BitTorrent primer

Fast, users share the work
BitTorrent primer

Fast, users share the work

File pieces
BitTorrent primer

Fast, users share the work

File pieces
BitTorrent primer

Fast, users share the work

Interested

File pieces
BitTorrent primer

Fast, users share the work

Interested

Interested
Fast, users share the work

File pieces

Interested

Interested

110

101

110

101
Piece under-reporting is equivocation

Yields prolonged interest from others and faster download times
Piece under-reporting is equivocation
Piece under-reporting is equivocation
Piece under-reporting is equivocation
Piece under-reporting is equivocation
Piece under-reporting is equivocation
Piece under-reporting is equivocation

I never received

I received
Applying TrInc

• What does the counter represent?
  • The number of pieces received

• To what do peers attest?
  • Their bitfield
  • The most recent piece received

• When do peers attest?
  • When they receive
  • When they sync their counters
I have and most recently received
I have and most recently received

I have and most recently received

I have and most recently received
TrInc-BitTorrent

1
I have ▼ and most recently received ▼

2
I have ▼ ▼ and most recently received ▼

3
I have ▼ ▼ ▼ and most recently received ▼

✓ Counter matches the bitfield size
TrInc-BitTorrent

1
I have yellow and most recently received yellow

2
I have yellow-purple and most recently received purple

3
I have yellow-purple-blue and most recently received blue

✓ Counter matches the bitfield size
TrInc - BitTorrent

1. I have ▼ and most recently received ▲

2. I have ▲ and most recently received ▲

3. I have ▲ ▼ and most recently received ▲

✓ Counter matches the bitfield size
✓ Attests to most recent piece
TrInc-BitTorrent

1. I have ▼ and most recently received ▶

2. I have ▼▼ and most recently received ▶

3. I have ▼▼▲ and most recently received ▶

✓ Counter matches the bitfield size
✓ Attests to most recent piece
Why attest to the latest piece?
Why attest to the latest piece?
Why attest to the latest piece?

I have
Why attest to the latest piece?

I have
Why attest to the latest piece?

I have □
Why attest to the latest piece?

1. I have [yellow]

2. I have [purple]

2. I have [blue]
Why attest to the latest piece?

1. I have [yellow]
   Looks good to me

2. I have [purple, blue]
   Looks good to me

2. I have [purple, blue]
   Looks good to me

TrInc – NSDI 2009  Dave Levin
Why attest to the latest piece?

1. I have ☢️

2. I have ☢️

3. I have ☢️

Looks good to me
Why attest to the latest piece?

Lesson: Without the full log, must ensure proper behavior at each step.
Macrobenchmarks

• TrInc-BitTorrent
  • Solves piece under-reporting

• TrInc-A2M
  • Reduces hardware requirements
  • Higher throughput

• TrInc-PeerReview
  • Reduces the communication necessary to achieve fault detection
Contributions

1. **TrInc** – A new, practical primitive for eliminating equivocation

2. Applications of TrInc

3. Implementation in currently available hardware
Contributions

1. TrInc – A new, practical primitive for eliminating equivocation

2. Applications of TrInc

3. Implementation in currently available hardware
Implementation

- Gemalto .NET Smartcard
  - Crypto unit (RSA & 3-DES)
  - 32-bit micro-controller
  - 80 KB persistent memory

- A few dozen lines of C#

- Case studies
  - TrInc-A2M
  - TrInc-PeerReview
  - TrInc-BitTorrent
TrInc microbenchmarks

Operation time (msec)

noop | Asym Attest (advance) | Asym Attest (status) | Symm attest (advance) | Symm Attest (status) | Verify

0 | 225 | 200 | 125 | 100 | 75
TrInc microbenchmarks

32 msec to write a counter

Operation time (msec)

- noop
- Asym Attest (advance)
- Asym Attest (status)
- Symm attest (advance)
- Symm Attest (status)
- Verify

TrInc – NSDI 2009
Dave Levin
TrInc microbenchmarks

32 msec to write a counter

Operation time (msec)

- noop
- Asym Attest (advance)
- Asym Attest (status)
- Symm attest (advance)
- Symm Attest (status)
- Verify

Only 2x
Why so slow?

• Fundamentally new application of trusted hardware
  • Typically used for bootstrapping
  • TrInc makes it intrinsic to the protocol

• It can be faster
  • There just has not been the call for it prior to TrInc
Summary

- Equivocation is a versatile and powerful
- A small amount of trust can secure a large system

**TrInc** is
- Minimal – A counter and a key
- Versatile – Applies to a wide range of systems
- Practical – Uses the same components available today
TrInc speeds up A2M

TrInc-A2M  A2M

Operation time (msec)

Append  Lookup (successful)  Lookup (too early)  Lookup (forgotten)  End  Truncate  Advance

150  300  300  450  300  150  270
TrInc speeds up A2M

TrInc does not go to h/w for successful lookups

Operation time (msec)

Append
Lookup (successful)
Lookup (too early)
Lookup (forgotten)
End
Truncate
Advance

TrInc-A2M  A2M
TrInc speeds up A2M

TrInc does not go to h/w for successful lookups

TrInc requires attestations

Operation time (msec)

Append  Lookup (successful)  Lookup (too early)  Lookup (forgotten)  End  Truncate  Advance

TrInc does not go to h/w for successful lookups.
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation
Block Revelation

Strategically *under*-report
TrInc-BitTorrent Results

Cumulative number of blocks obtained vs. Time into the download (sec)
TrInc-BitTorrent Results

Cumulative number of blocks obtained vs. Time into the download (sec)

Representative peer
TrInc-BitTorrent Results

![Graph showing cumulative number of blocks obtained over time into the download.]

- **Cumulative number of blocks obtained**
- **Time into the download (sec)**

**Lines in the graph:**
- **Representative peer**
- **Under-reporter: from all**
TrInc-BitTorrent Results

Cumulative number of blocks obtained vs. Time into the download (sec)

- Representative peer
- Under-reporter: from all

Under-reporter pulls ahead
TrInc-BitTorrent Results

But ultimately downloads slower

Under-reporter pulls ahead
TrInc-BitTorrent Results

But ultimately downloads slower

Under-reporter pulls ahead
TrInc-BitTorrent Results

But ultimately downloads slower

Truth-tellers
A median of 6% from the seeder

Under-reporter
73% of file from the seeder

Under-reporter pulls ahead