TrInc: Small Trusted Hardware for Large Distributed Systems

Paper By: Levin, Douceur, Lorch, and Moscibroda

Presented By: Nabil Abou Reslan
Outline

- TrInc and Trust in Distributed Systems
- Equivocation and TrInc
- TrInc Components
- How does TrInc achieve its goal?
- Case Study using TrInc
- Miscellaneous
Distributed systems rely heavily on messages to communicate.

These messages need to be trusted for security purposes.

Byzantine faults.

Handling Byzantine faults is costly.
What is TrInc?

- Stands for **Trusted Incrementer**
- TrInc is simple hardware that provides trust to a distributed system
- It does this through uniquely attesting messages to reduce tampering
- Can significantly increase system security at each distributed component
- Applies to Peer-to-Peer systems as well
Equivocation and TrInc

Definition: “making conflicting statements to others”

Selfish or malicious intents, not just Byzantine faults

TrInc provides trust by removing ability to equivocate

TrInc Goals:
  - Remove ability to equivocate
  - Reduce message overhead
  - Reduce number of non-faulty participants needed
Trlncc Components

- Unique Identity
- Incrementer
- Cryptographic functions and keys
- Incrementer Counter (Meta-counter)
- Attestation
- Queue
How does it work?

- For every new message:
  - Assign a new value to the Incrementer for that message
  - Value must be greater than previous number
  - Create an attestation to send along with message

- The attestation is encrypted

- Typically, the value of Incrementer should represent the progression of attested data

- Value of Incrementer does not decrease, so cannot assume value used in previous messages
How does it work?

- New Incrementers can be issued, with unique id
  - Can attest to multiple things
  - Allows processes to use their own Incrementer

- Queue is used to recover from power failures
  - Contains the last few message attestations
How does TrInc achieve its goal?

- It prevents equivocation by ensuring message attestations are always distinct
  - The counter never assumes previous values
  - Every counter has unique id
  - Every TrInc has unique id

- Reduce message overhead
  - Attested messages need not be verified like in the Byzantine Generals Problem
  - From $O(n^2)$ to $O(1)$
Case Study: BitTorrent

- BitTorrent is a Peer-to-Peer network that distributes the work of downloading large files
  - Peers trade pieces of the file with each other
  - Maintain list of pieces downloaded in a Bitfield
  - Peers want pieces they don’t have

- Equivocation: Under-reporting of pieces to increase interest

- Under-reporting is done to increase download speeds
Case Study: BitTorrent

- TrInc can attest to Bitfield and to the most recent piece received
- Bitfield can only increase, natural fit for Incrementer
- Peer can’t under-report
TrInc is independent from the system’s inner state

- TrInc can be inserted using a USB
- Or included in computer hardware

In order for TrInc to work, the receiver must also have a TrInc in use

The manufacturer knows the private key for every TrInc Identity
TrInc is hardware that increases trust in distributed systems

TrInc reduces message overhead

TrInc use unique attestations and encryption to prevent equivocation

TrInc can be used in P2P systems like BitTorrent to prevent under-reporting
Questions?

- How does TrInc reduce number of non-faulty participants needed in Byzantine fault tolerance to $2n+1$ from $3n+1$?
- Can we really trust the manufacturer with the private key?