Recent Approaches I:

"Secure" hardware as failure detectors
Why?

- Question I (or many others) have been pondering
  “Can attestation and/or confidential computing help with distributed systems?”
  Pretty easy to find today.
- Relates to failure detectors (what we studied 2 weeks ago)
- Some of you were concerned about the age of what we studied.

First, a bit about hardware

- attestation

- What programs are running on this laptop?

- What other programs are running on this laptop?
Remote Attestation

Need proof showing other node is running program P.

How?

Why?
How?

ENCLAVES / CONFIDENTIAL COMPUTING
Protecting data from attackers who have physical access.

Why?

Tamper-proof logs: Protecting data from modification.

Requirements:
- Do not override previous entries.
- Malicious or buggy server cannot respond to a request with **incorrect** on old values.
For insertions: check whether or not insertion succeeded

For lookups: correct value is returned

No replay

How (today)?

→ Attest that a trusted log is used.

→ Require log to sign all responses
  → With what key?

→ What information does a response include?
- Can we implement this today?
  
  Type of adversary

- Performance costs?

- Safety?
Why focus on tamper-proof logs?

Q: The power of tamper-proof logs to solve Byzantine consensus?

Previously

0 Partial synchrony + reliable channels + HMAC/Signatures

$3f+1$

Can we do better?
Q1. Do we need to change interfaces?

Q2. How to change the protocol?

Attempt 1

Count \( f+1 \) replies
Safety: log agreement $\iff$ linearizable order

Cannot ensure liveness. Why?
Problem: Attestation after commit does not prevent malicious processes from equivocating.

Solution: Log and attest all messages on the way.

1. Process a single (or a fixed set) of client requests at a time.
2. Add each message to a log before sending it. Include attestation indicating where in the log a message shows up.
3. Receivers process messages in log order.

Trusted log ensures different sequences.
Append $R_B$

Consequence: Faulty nodes cannot cause logs to diverge.

Q: Is this enough to ensure liveness?

PBFT view change: If valid view change requests from other replicas. Why?

View change here:

$\begin{array}{c}
\text{If } \frac{f}{f+1} + f < 1 \\
& \text{f}
\end{array}$

Fault Tolerance Thus Far
Need $(f+1)$ responses at the client

$(f+1)$ view change requests

... 

Up to $f$ nodes can fail

Need $2f+1$

Are we done?

Observations

1. Faulty processes can act so that a correct processes log is ahead of others

![Diagram]

Appends Ra

Will not
(2) Clients are responsible for preventing Denial of Service?

How?

(3) Is this a problem?
Where are we on this/other protocols.

- Tor/Private relays?

- Others?