Aalto University

Can blockchains be made better using hardware-assisted security?

Lachlan J. Gunn, N. Asokan

What is a Blockchain?

A (public) ledger whose integrity is guaranteed

Each block is a set of transactions, cryptographically linked to the previous block

- Acceptance of one block implies agreement on entire history

Problem: How to reach consensus on what transactions get included in a block?

Choose who decides what transactions are included in a block

Devise a way for everyone to agree on the sequence of blocks

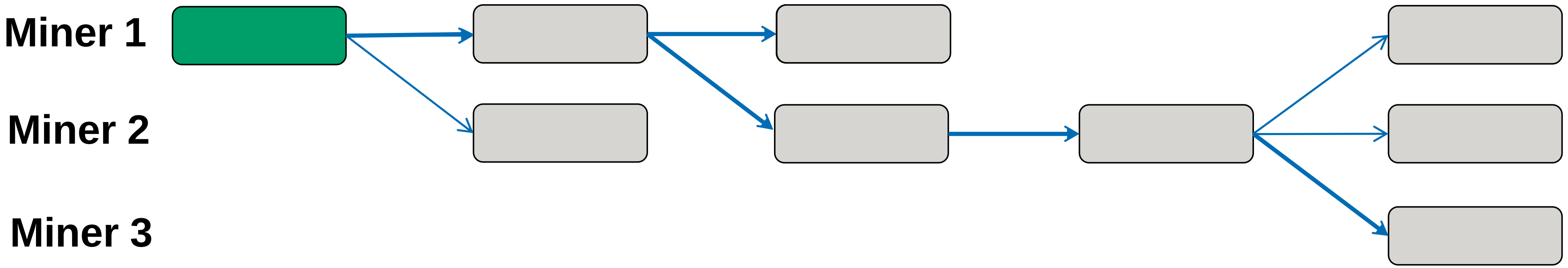
Proof of Work + “longest chain” rule

Bitcoin, Ethereum, etc. all use Proof of Work to agree on the next block:

Miners decide which transactions include in their proposal for the next block

Proof of Work: use computation power to solve a puzzle; winner proposes next block

- Chance of success proportional to amount of computation (work) performed
- Fair: any miner expending the same amount of work has the same chance of winning



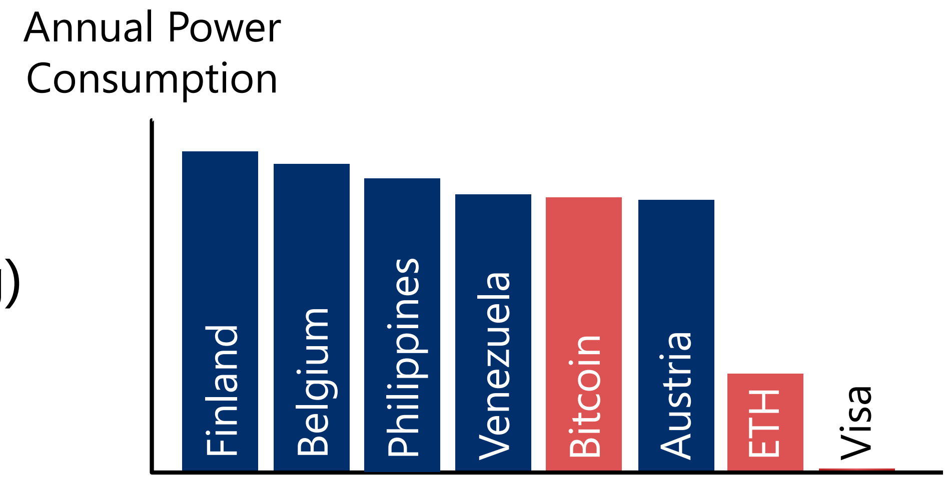
- Everyone follows the longest valid chain (chain with largest CPU power wins eventually)

What's wrong with Bitcoin, anyway?

The luxury of not trusting anyone does not come for free:

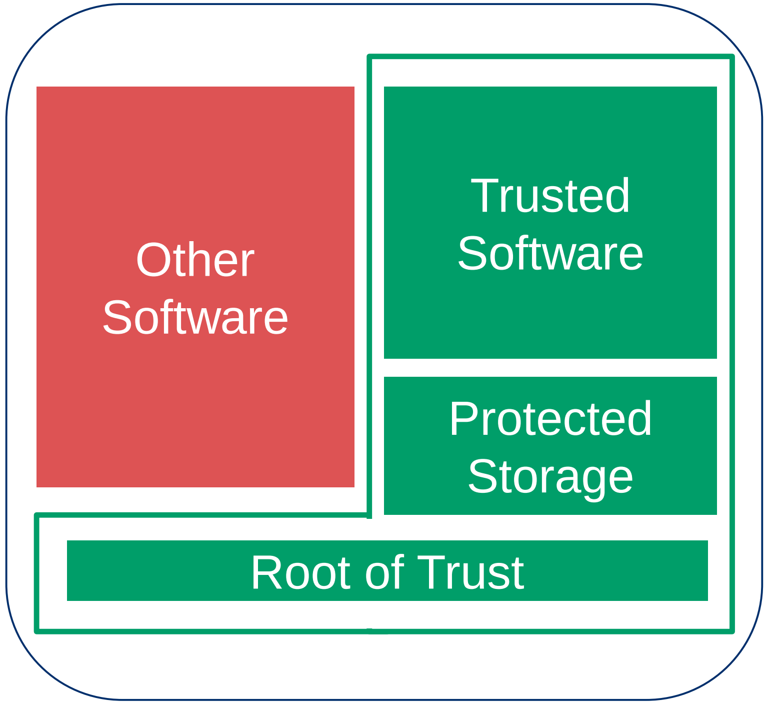
All transactions need to be online
Slow: long confirmation time, low throughput

Wasteful (energy expended on puzzle solving)
Probabilistic finality
Extremely scalable



Data: Digiconomist, CIA World Factbook

Can hardware-assisted “trusted computing” help?



Hardware support for

- Isolated execution
- Protected storage: Sealing
- Ability to report status to a remote verifier: Attestation

Trusted Execution Environment (TEE)

Cryptocards

https://www.ibm.com/security/cryptocards/

Trusted Platform Modules

https://www.infineon.com/tpm

ARM TrustZone

https://www.arm.com/products/security-on-arm/trustzone

Intel Software Guard Extensions

https://software.intel.com/en-us/sgx

Outline

How to use hardware-assistance to improve blockchains?

- **Changing the “business process”**
- **Replacing consensus (“longest chain” rule)**
- **…**

What challenges arise?

Changing the process

Fast off-chain transactions with TEEs

Offline
Fast

Bitcoin payments are made from/to cryptographic keys

TEE can enforce how a key is used and attest to such usage

1. Online (on-chain): transaction to transfer money to a TEE-protected key
   Proves initial balance using the blockchain
2. Offline: payment message + TEE-provided attestation: key used in only one outgoing payment

Fast, offline payment to any payee who

- is guaranteed instantaneously that double-spending is not possible!
- but must wait for on-chain confirmation before using the money with anyone!

Gopinath Nirmala, "Improving the Security and Efficiency of Blockchain-based Cryptocurrencies", MSc thesis @Aalto, 2017.
Dmitrienko et al., "Secure Wallet-Assisted Offline Bitcoin Payments with Double-Spender Revocation", ASIACCS '17.

Teechan: Net settlement with TEEs

Offline

Fast

TEEs can use attestation to create a secure channel between them.

1. Decide how much you trust the TEE. Set a credit limit.
2. Create a secure channel between the TEEs.
3. Transaction made via this channel: TEEs keep track of net transfer value.
4. Either TEE can close the channel and perform net settlement.

Fast, offline series of payments between two designated parties:

- guaranteed instantaneously that double-spending is not possible!
- can reuse the money for transactions with peer immediately
- but must wait for on-chain confirmation before using the money with anyone else

Lind et al. "Teechan: Payment Channels Using Trusted Execution Environments", Bitcoin '17.

Efficient

Proof of Elapsed Time

Proof of Work:

First miner to solve puzzle wins (gets to proposes next block)

Work ~ Exp (difficulty)

Proposals can be made at a rate proportional to computational power

Proof of Elapsed Time:

TEE issues attestation after waiting (idly) for a while; First miner to get the attestation wins

Idle wait time ~ Exp (difficulty)

Proposals can be made at a rate proportional to the number of idle CPUs

Intel, Hyperledger Sawtooth Documentation (2015).

Replacing Consensus

Byzantine Consensus

Slow

Probabilistic

Wasteful

Goals of classical Consensus schemes:

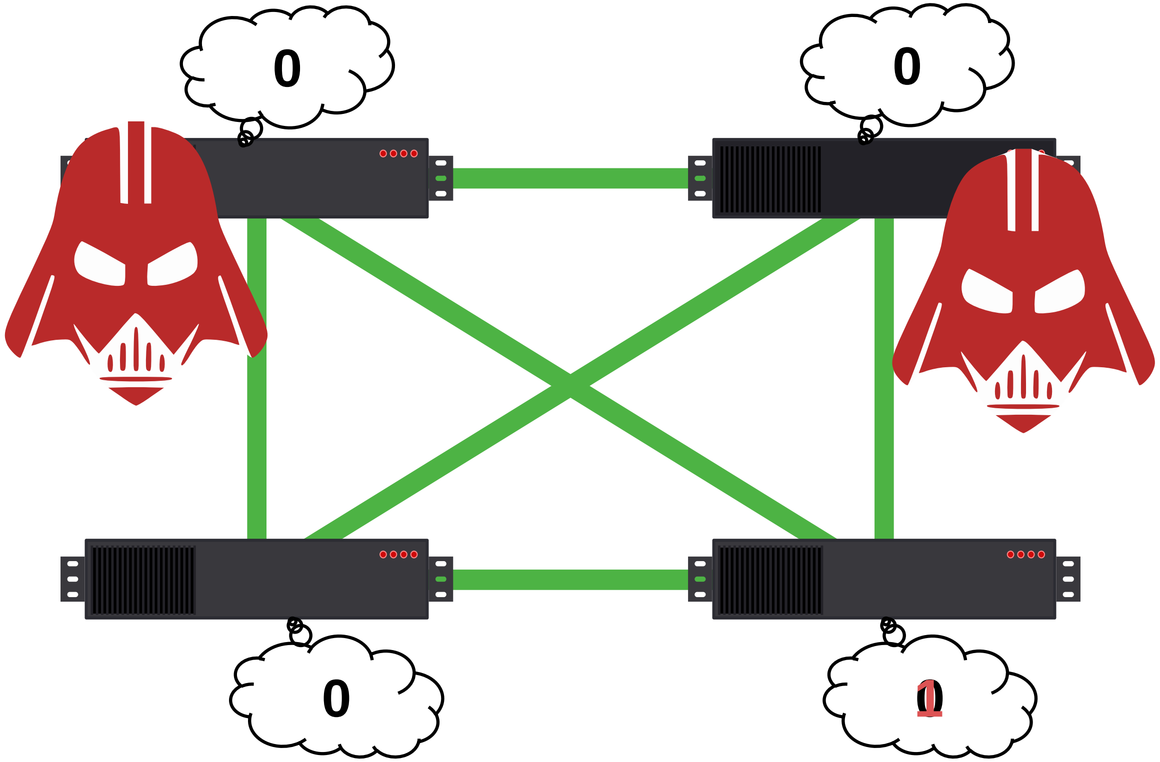
- Liveness: all (honest) nodes produce output
- Safety: all (honest) nodes output same value
- Finality: output values are definitive

Adversary model:

- Adversary can compromise some nodes
- Goals hold despite *f* compromised nodes

Limits:

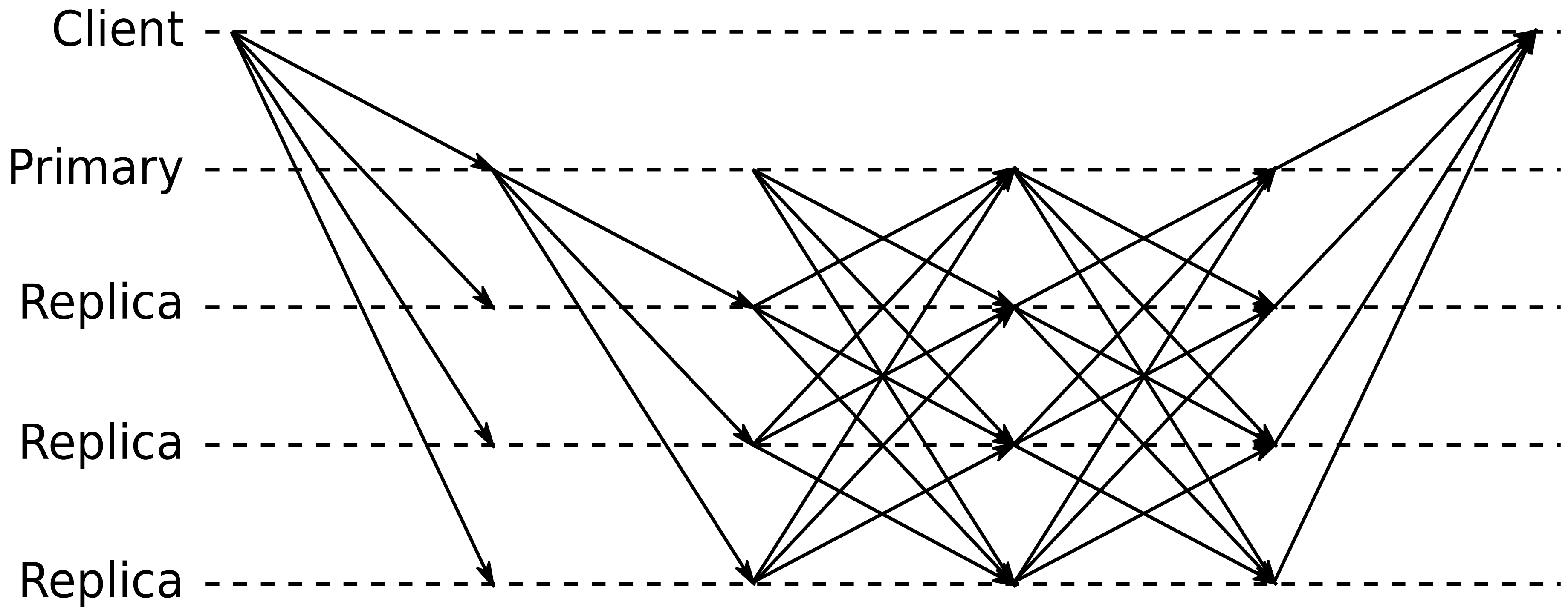
- No protocol can tolerate more than a third of nodes being compromised



PBFT

The first practical protocol for Byzantine fault tolerance

Less scalable than Proof of Work.

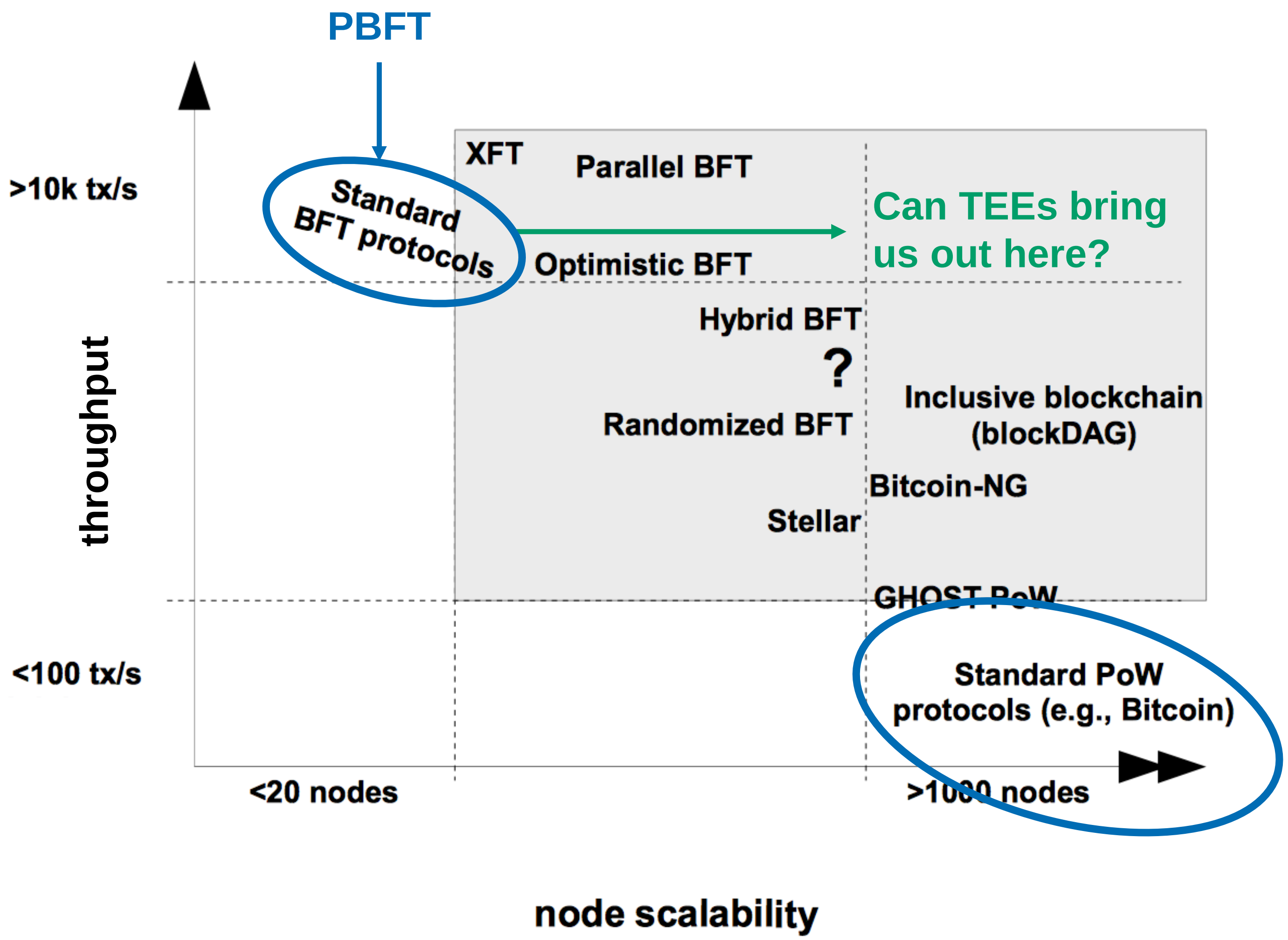


O(n^2) messages, $n = 3f + 1$

Castro & Liskov, "[Practical Byzantine Fault Tolerance]", OSDI'99.

The landscape of consensus mechanisms

Fast
Deterministic
Efficient
Scalable



Adapted from Marko Vukolić, "[The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication](#)." International Workshop on Open Problems in Network Security. Springer International Publishing, 2015.

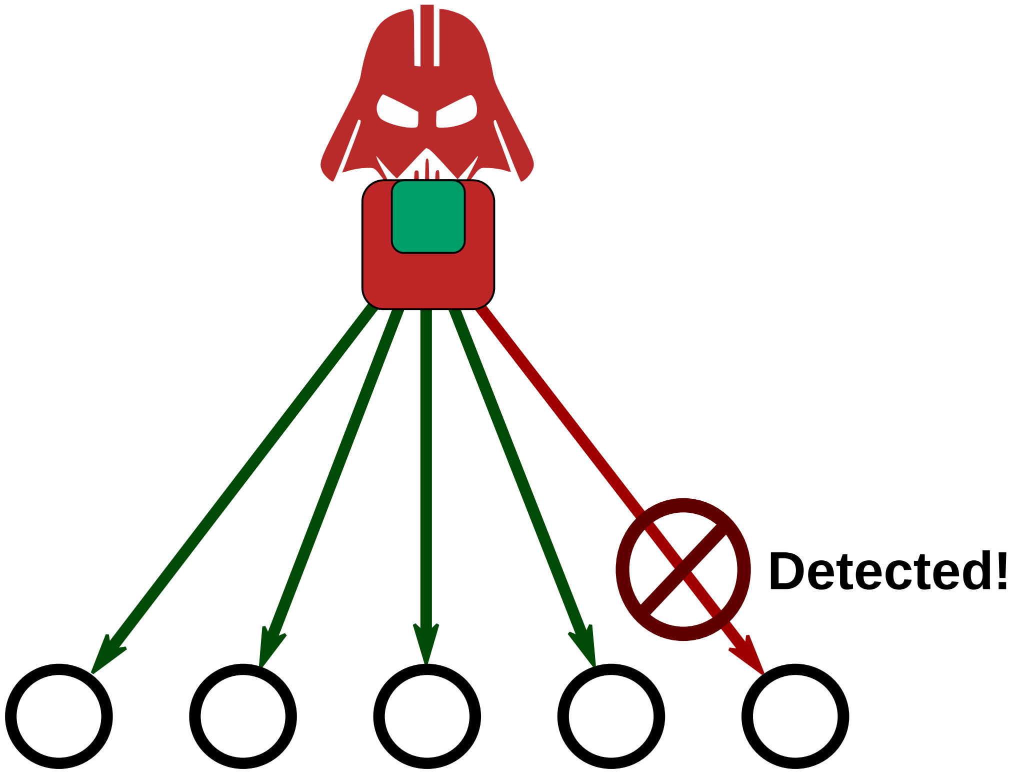
How can TEEs help design scalable consensus?

Problem: Compromised nodes can equivocate

Solution: Use attestation to prevent equivocation!

- Tolerate faults in ½ of the nodes

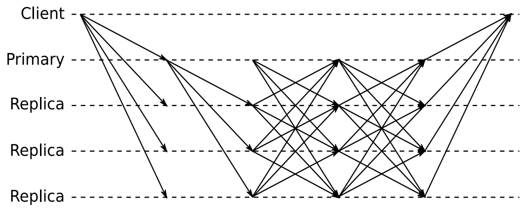
Applicability limited to permissioned settings



Chun et al., "Attested append-only memory: making adversaries stick to their word", SOSP '07

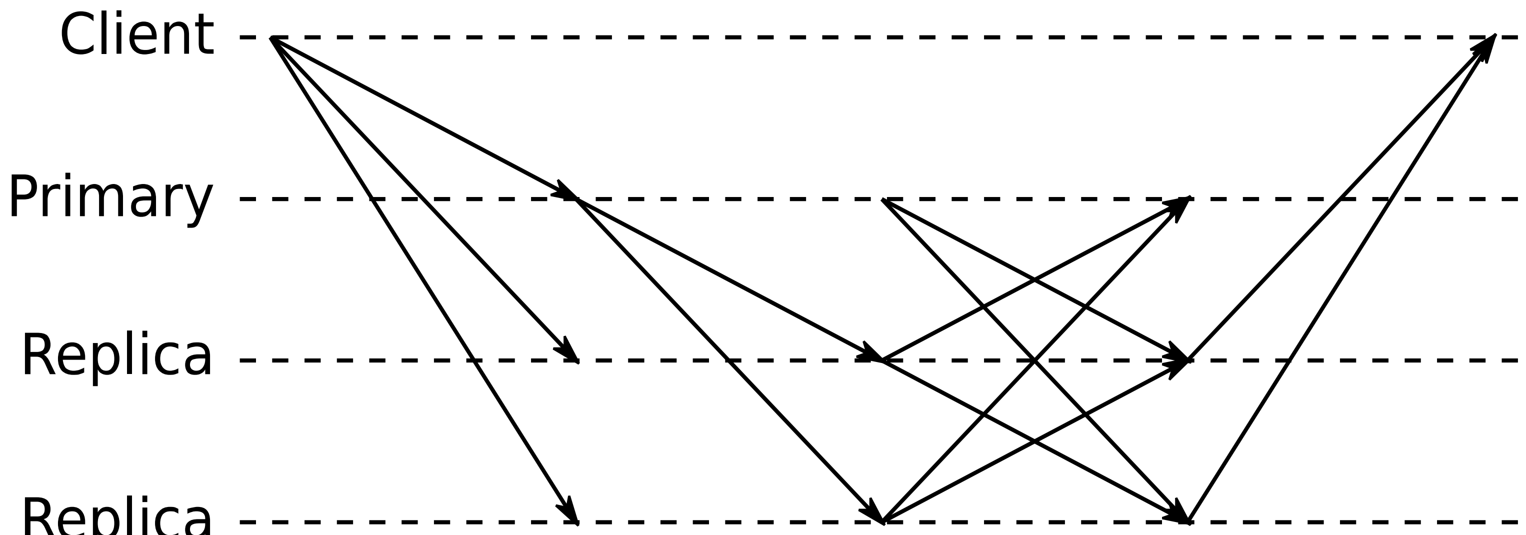
MinBFT

PBFT



Hardware-based monotonic counters

→ increase fault-tolerance

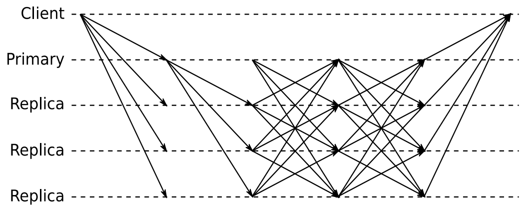


O(n^2) messages, $n = 2f + 1$

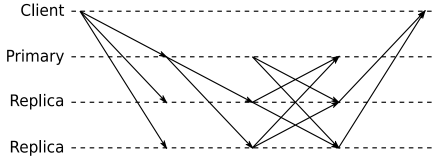
Veronese et al., "Efficient Byzantine fault-tolerance." IEEE Trans. Computers 62.1 (2013): 16–30.

FastBFT

PBFT

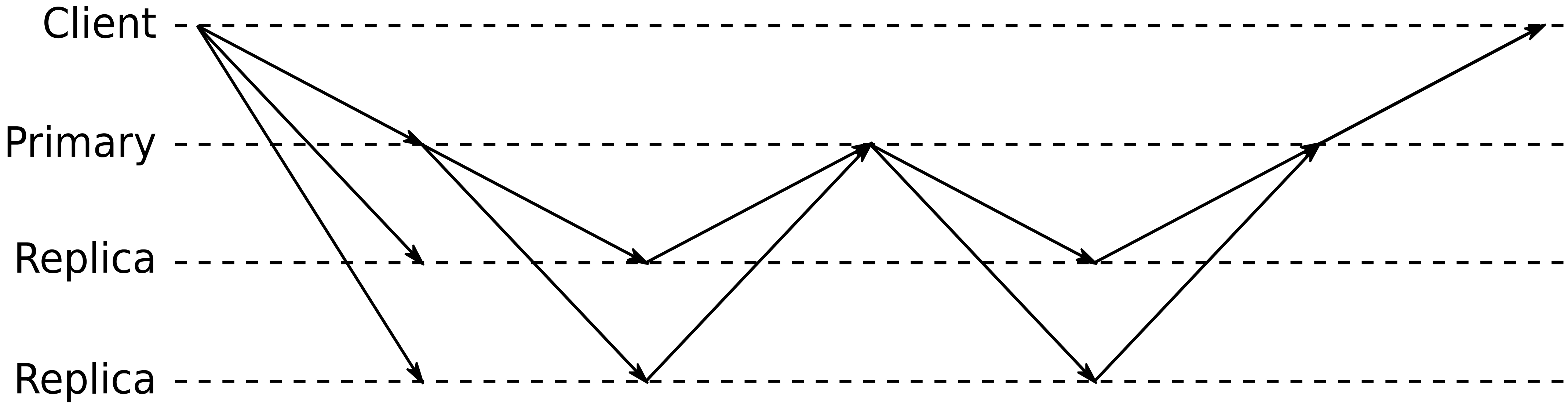


MinBFT



TEE-protected secret sharing, message aggregation

→ increase throughput

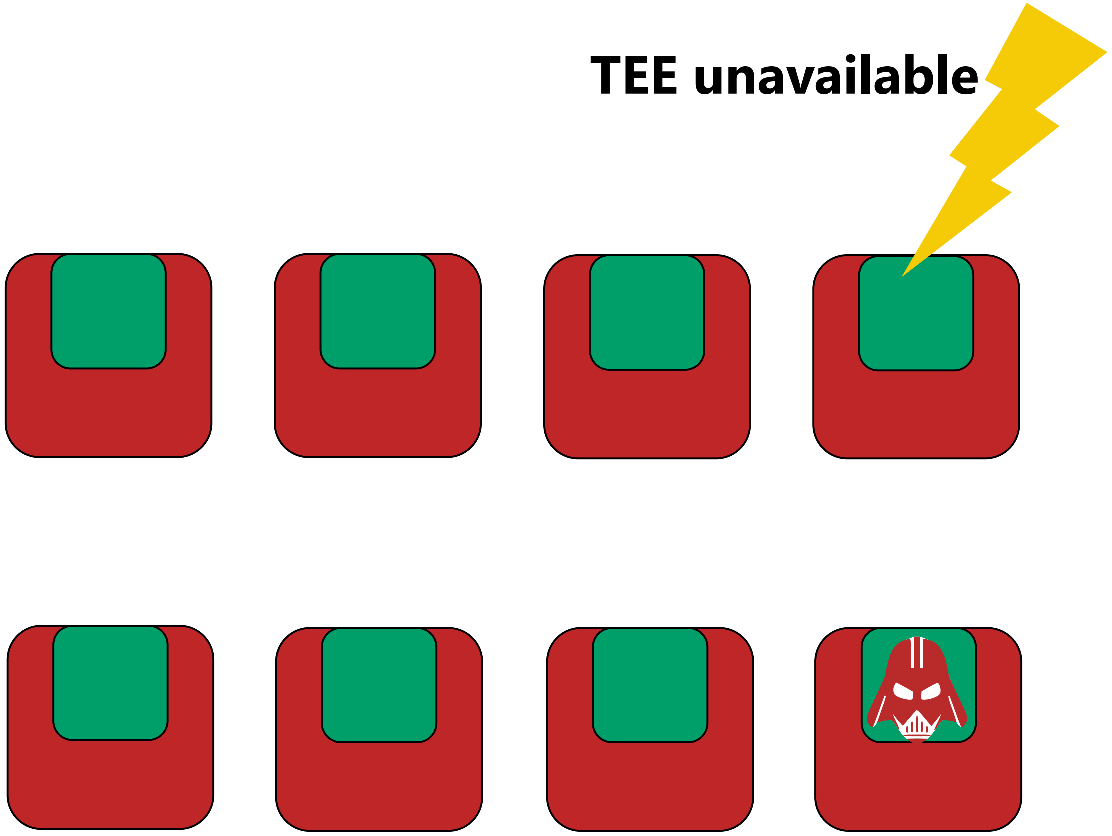


$O(n)$ messages, $n = 2f + 1$

Liu et al., "Scalable Byzantine Consensus via Hardware-assisted Secret Sharing", *IEEE Trans. Computers (2018).*

Challenges

Challenges in relying on hardware-assistance



TEE Availability:

- TEEs will not be universally available:
  - Gradual rollout
  - Obsolescence
  - Revocation

TEE Compromise:

- Compromising some TEEs should not completely break the system

Example: Dealing with TEE availability in consensus

Question: Can we improve consensus protocols by adding only a few TEEs?

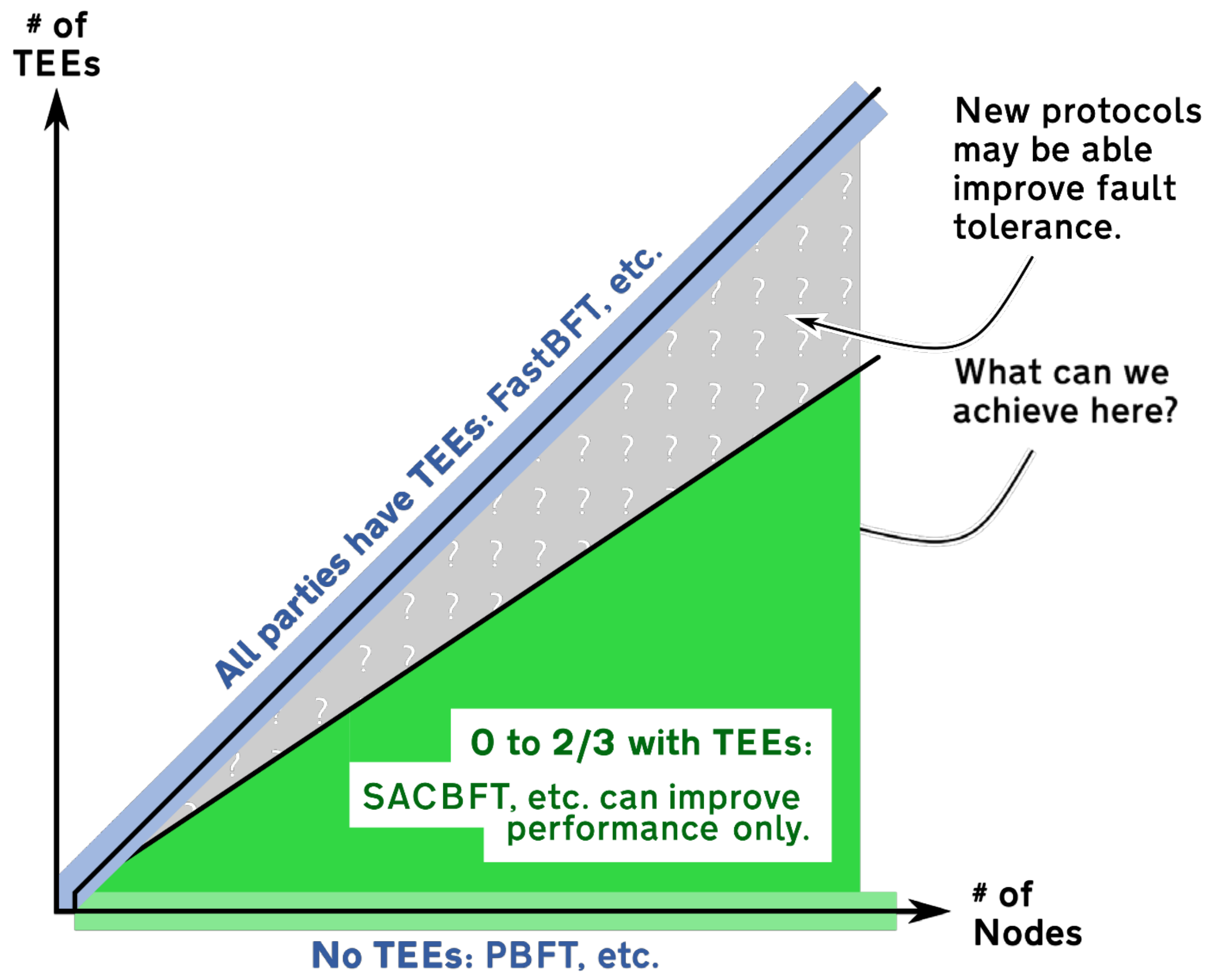
Answer*:

- can increase throughput if
  #TEEs > 1
- but fault tolerance cannot be increased if
  (#TEEs / #Nodes) ≤ 2/3

Open question: How can we optimally increase fault tolerance when

2/3 < (#TEEs / #Nodes) < 1

(* Forthcoming research report)



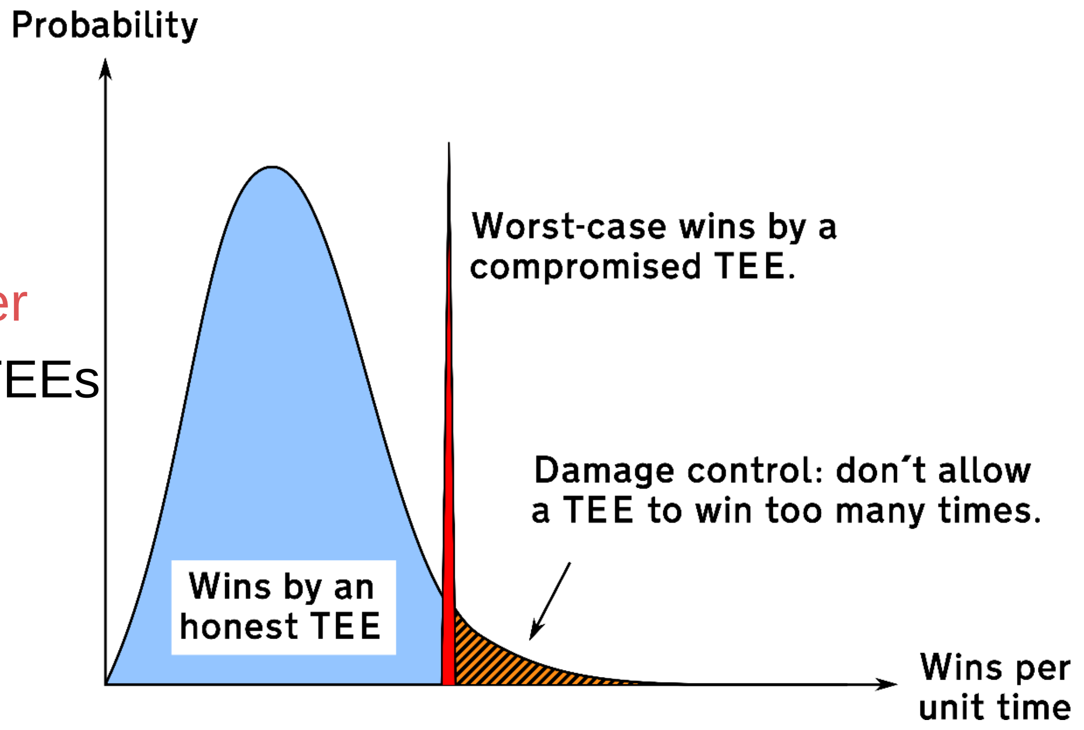
Example: Dealing with TEE compromise in PoET

Problem: A compromised TEE can win every block

Statistical solution: refuse blocks from machines that have won too many times

- Before: compromised TEEs give attacker unlimited power
- After: attacker power proportional to # of compromised TEEs

Open question: How can TEE-using applications detect/mitigate effects of TEE-compromise?



Intel, Hyperledger Sawtooth Documentation (2015).

Chen et al., "On Security Analysis of Proof-of-Elapsed-Time (PoET)", SSS 2017.

Summary

Hardware-assisted TEEs can improve blockchain-based systems

- Faster transactions, increased throughput, better efficiency,… *without sacrificing scalability*

Any solution relying on hardware-assisted security must

- Mitigate effects of hardware compromise
- Work without universal hardware support

https://ssg.aalto.fi/research/projects/bcon/
BCon project, Academy of Finland

http://www.icri-cars.org/
ICRI-CARS, Intel