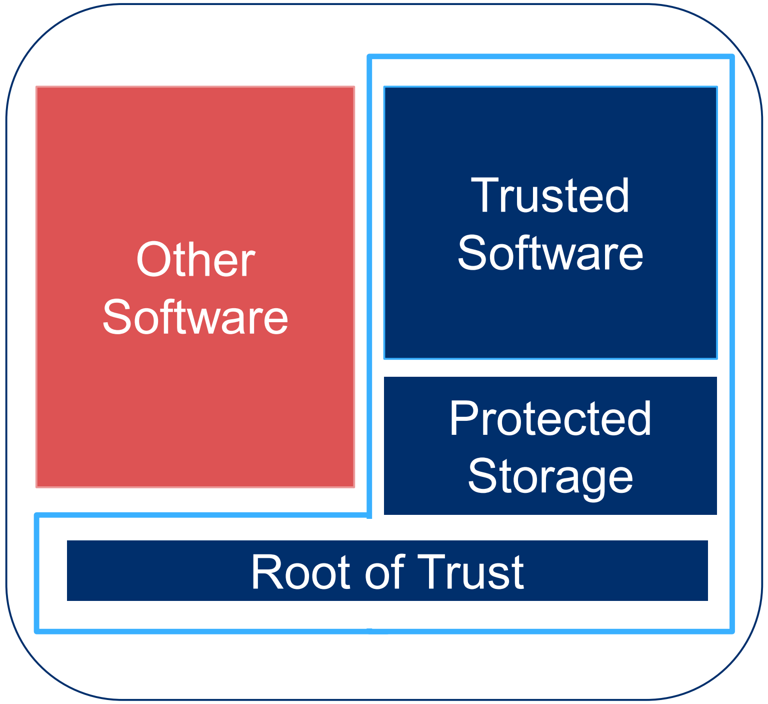
Aalto University

Common-sense applications of hardware-based TEEs

N. Asokan
http://asokan.org/asokan/
@nasokan

Acknowledgements: Thomas Nyman, Lachlan Gunn

Hardware-security mechanisms are pervasive



Hardware support for

- **Isolated execution: Isolated Execution Environment**
- **Protected storage: Sealing**
- **Ability to convince remote verifiers: Remote Attestation**

Trusted Execution Environments (TEEs)
Operating in parallel with "rich execution environments" (REEs)

Cryptocards

Trusted Platform Modules

ARM TrustZone

Intel Software Guard Extensions

https://www.ibm.com/security/cryptocards/ https://www.infineon.com/tpm https://www.arm.com/products/security-on-arm/trustzone https://software.intel.com/en-us/sgx

[A+14] "Mobile Trusted Computing", Proceedings of the IEEE, 102(8) (2014)
[EKA14] "Untapped potential of trusted execution environments", IEEE S&P Magazine, 12:04 (2014)

Concerns with TEEs: flaws

TPM Reset Attack

50,012 views

Evan Sparks
Published on Jun 18, 2007

A demonstration of a vulnerability in the TCG architecture v
running TPM without restarting the platform.

http://www.cs.dartmouth.edu/~pkilab/sparks/ (2007)

CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Authors:
Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo, *Columbia University*
Distinguished Paper Award Winner!

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang (2017)

Foreshadow (security vulnerability)

From Wikipedia, the free encyclopedia

This article is about the security vulnerability. For other uses, see Foreshadow (disambiguation).

Foreshadow is a vulnerability that affects modern microprocessors that was first discovered by two independent teams of researchers in January 2018, but was first disclosed to the public on 14 August 2018.[1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16] The vulnerability is a speculative execution attack on Intel processors that may result in the loss of sensitive information stored in personal computers, or third party clouds.[1] There are two versions: the first version (original/Foreshadow) (CVE-2018-3615) targets data from SGX enclaves; and the second version (next-generation/Foreshadow-NG) (CVE-2018-3620 and CVE-2018-3646) targets Virtual Machines (VMs), hypervisors (VMM), operating system (OS) kernel memory, and System Management Mode (SMM) memory.[1] Intel considers the entire class of speculative execution side channel vulnerabilities as "L1 Terminal Fault" (L1TF).[1] A listing of affected Intel hardware has been posted.[10][11]

Foreshadow is similar to the Spectre security vulnerabilities discovered earlier to affect Intel and AMD chips, and the Meltdown vulnerability that also affected Intel.[6] However, AMD products, according to AMD, are not affected by the Foreshadow security flaws.[6] According to one expert, "[Foreshadow] lets malicious software break into secure areas that even the Spectre and Meltdown flaws couldn't crack".[15] Nonetheless, one of the variants of Foreshadow goes beyond Intel chips with SGX technology, and affects "all [Intel] Core processors built over the last seven years".[2]

Foreshadow may be very difficult to exploit,[2][6] and there seems to be no evidence to date (15 August 2018) of any serious hacking involving the Foreshadow vulnerabilities.[2][6] Nevertheless, applying software patches may help alleviate some concern(s), although the balance between security and performance may be a worthy consideration.[5] Companies performing cloud computing may see a significant decrease in their overall computing power; individuals, however, may not likely see any performance impact, according to researchers.[9] The real fix, according to Intel, is by replacing today's processors.[5] Intel further states, "These changes begin with our next-generation Intel Xeon Scalable processors (code-

FORESHADOW
A logo created for the vulnerability, featuring a lock with a shadow

https://en.wikipedia.org/wiki/Foreshadow_(security_vulnerability) (2018)

Concerns with TEEs: suspicions of motives

Software

MS Palladium protects IT vendors, not you – paper

Anderson gives us the FAQs

By John Lettice 28 Jun 2002 at 10:27 SHARE ▼

https://www.theregister.co.uk/2002/06/28/ms_palladium_protects_it_vendors/ (2002)

Problem: Third-party uncertainty about your software environment is normally a feature, not a bug

https://www.eff.org/wp/trusted-computing-promise-and-risk (2003)

Trusting Intel – Next Generation of Backdooring?

We have seen that SGX offers a number of attractive functionality that could potentially make our digital systems more secure and 3rd party servers more trusted. But does it really?

The obvious question, especially in the light of recent revelations about NSA backdooring everything and the kitchen sink, is whether Intel will have backdoors allowing "privileged entities" to bypass SGX protections?

http://theinvisiblethings.blogspot.fi/2013/09/thoughts-on-intels-upcoming-software.html (2013)

Possible motivations for widespread deployment

Vendor lock-in

Restriction of digital rights

…

Regulatory requirements

Protection of end-user data

…

Example: regulatory compliance

The IMEI shall not be changed after the ME's final production process. It shall resist tampering, i.e. manipulation and change, by any means (e.g. physical, electrical and software).

NOTE: This requirement is valid for new GSM Phase 2 and Release 96, 97, 98 and 99 MEs type approved after 1st June 2002.

3GPP TS 42.009, 2001

Secure storage of RF configuration parameters

Early TEEs for mobile phones
(ca. 2001)

TrustZone®
Security Foundation by ARM®

[Saara Matala] "Historical insight into the development of Mobile TEEs", Aalto SSG research group blog (2019)

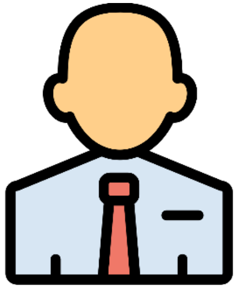
Mobile TEEs: Motivation

Business requirements:
- mobile payment
- subsidy lock
- custom silicon consolidation

Regulatory requirements:
- tamper-resistant IMEIs
- secure storage for RF

Supply-chain constraints:

Cost of discrete security chip too high on bill of materials!



☞ New approach: "*processor secure environments*"

Generic low-cost enabler emerged as skunkworks project within Nokia (rather than point solutions for particular use cases)

Mobile TEEs: Development

1982 — 1996 — 2002 — 2003 — 2004 — 2008 — 2015 — 2016 — 2019

2004 ARM TrustZone Tiago Alves and Don Felton
"TrustZone: Integrated Hardware and Software Security", Information Quarterly, 2004:3(4)

2003 Texas Instruments OMAP 161x and 73x processors
Harini Sundaresan, *"OMAP Platform Security Features"* whitepaper, 2003 (updated 2008)

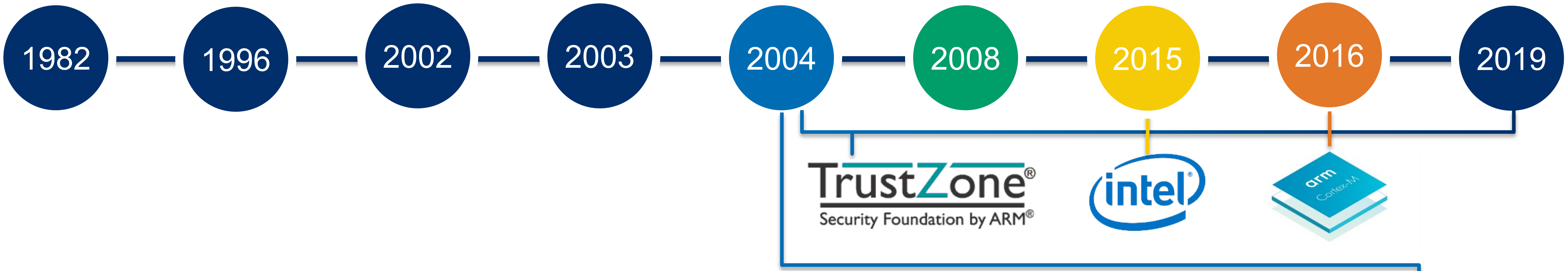
2002 Nokia, Kiiveri and Paatero US9111097B2
"Secure execution architecture"

1996 Intertrust, Ginter et al US 5892900A
"Systems And Methods For Secure Transaction Management And Electronic Rights Protection"

1982 Texas Instruments, Guttag US4521853A
"Secure microprocessor/microcomputer with secured memory"

1982 Texas Instruments, Guttag and Nussarallah US4521853A
"Security bit for designating the security status of information stored in a nonvolatile memory"

Mobile TEEs: Deployment



- **First deployment: Nokia 6630 (“Charlie”)**
  - first 3G phone with TI OMAP 1710 processor (June 2004)
- **ARM TrustZone currently widely deployed**
  - TrustZone-M for Cortex-M class microcontrollers (2016)
- **Ca. 2008, TEE unheard of academic circles**
  - first paper in FC 2008, ASIACCS 2009
- **Intel SGX**
  - SkyLake microarchitecture (2015)
  - wide availability of SDK “democratized” TEE research

Should we build systems that rely on TEEs?

Concerns with applicability of hardware-supported TEEs remain

But compelling common-sense applications exist

practical; protect end-users; address everyday needs

- Private membership test for malware scanning, private contact discovery,..
  [TLPEPA17] Circle Game, ACM ASIACCS https://arxiv.org/abs/1606.01655
- Protection of password-based web authentication
  [KKPMA18] SafeKeeper, WWW (WebConf) https://ssg.aalto.fi/research/projects/passwords/
- Secure accounting for function-as-a-service (FaaS) settings
  [AAKPS18], S-FaaS, in submission, https://export.arxiv.org/abs/1810.06080
- *Blockchains and cryptocurrencies*
  [LLKA19] FastBFT, IEEE TC https://doi.org/10.1109/TC.2018.2860009, [GLVA19] SACZyzzyva, SRDS, http://arxiv.org/abs/1905.10255
- …

Aalto University

Can blockchains be made better using hardware-assisted security?

Lachlan J. Gunn, N. Asokan

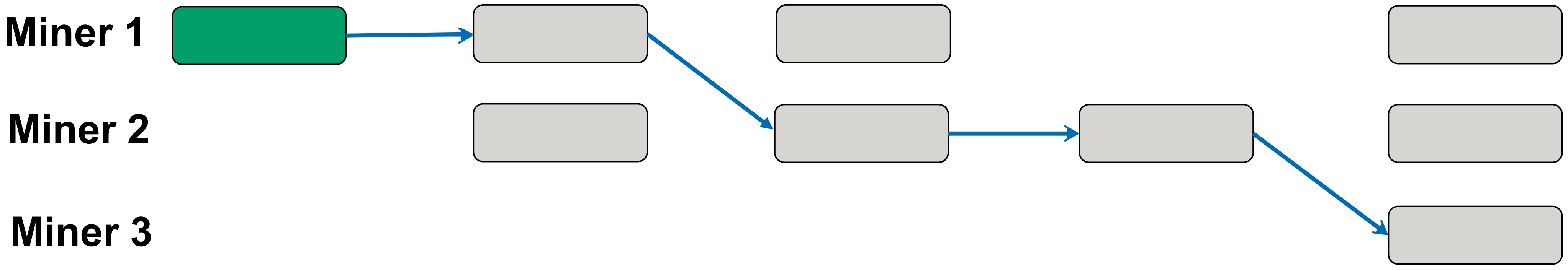
Proof of Work + “longest chain” rule

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



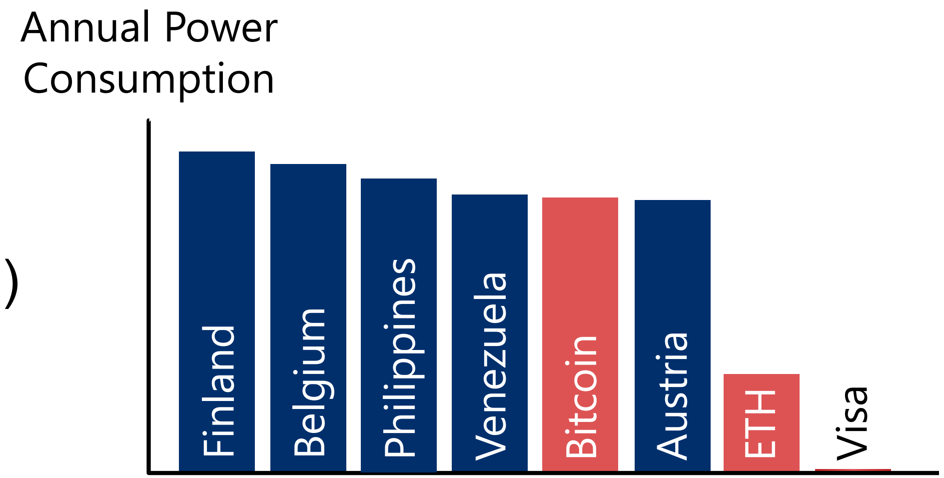
- Everyone follows the longest valid chain (chain with largest CPU power wins eventually)

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

Outline

Can hardware-assisted security improve blockchains?

Example approaches

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

What challenges arise?

Changing the process

Proof of Elapsed Time

Skip to TEE compromise

Efficient

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:

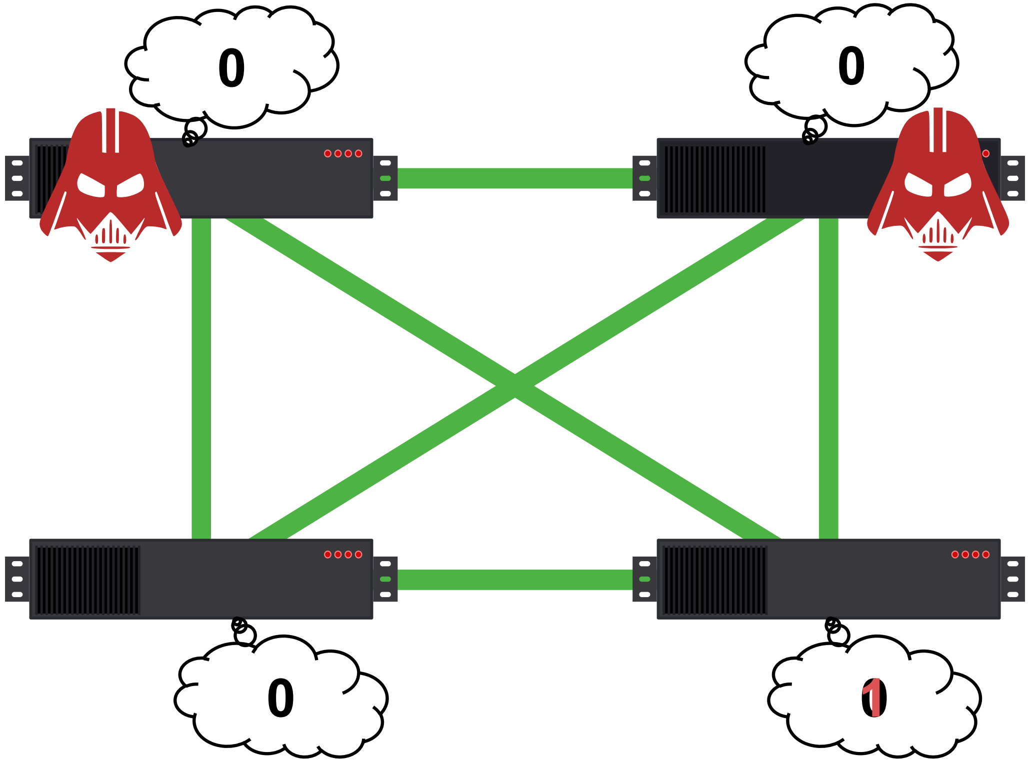
- Liveness: all (honest) nodes produce output
- Safety: all (honest) nodes output same value
- Finality: output values are definitive

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

Fast

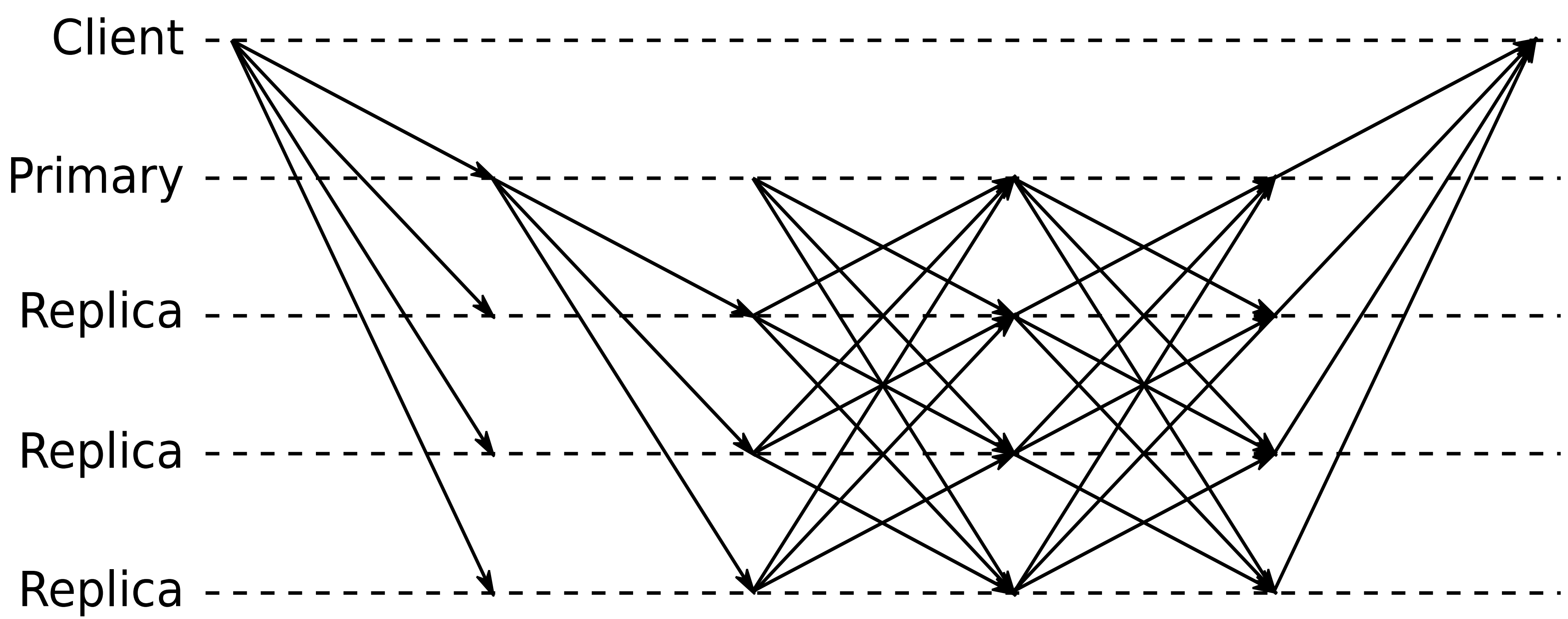
Deterministic

Efficient

Scalable

The first practical protocol for Byzantine fault tolerance

Less scalable than Proof of Work.

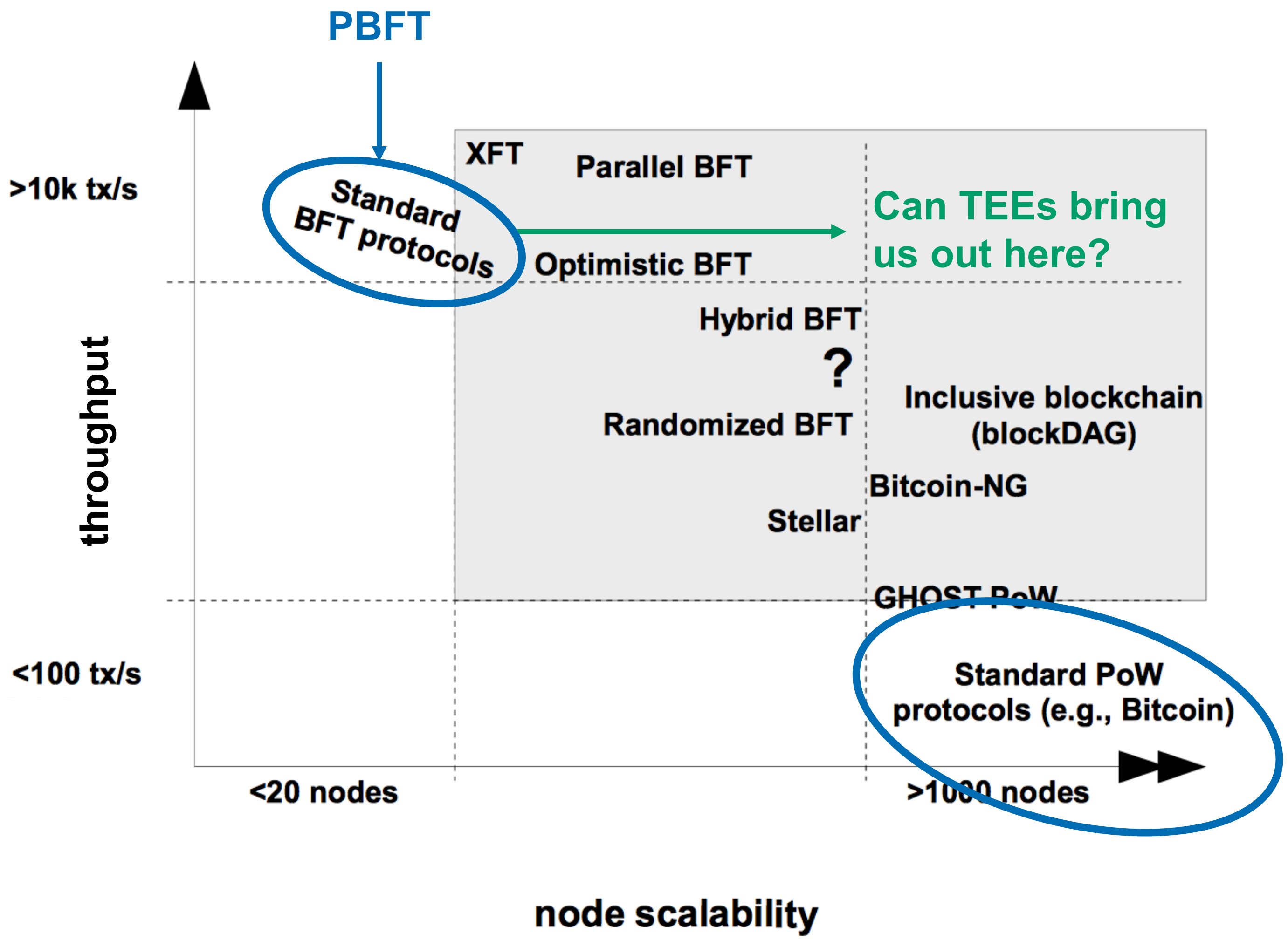


O(n^2) messages, $n = 3f + 1$

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

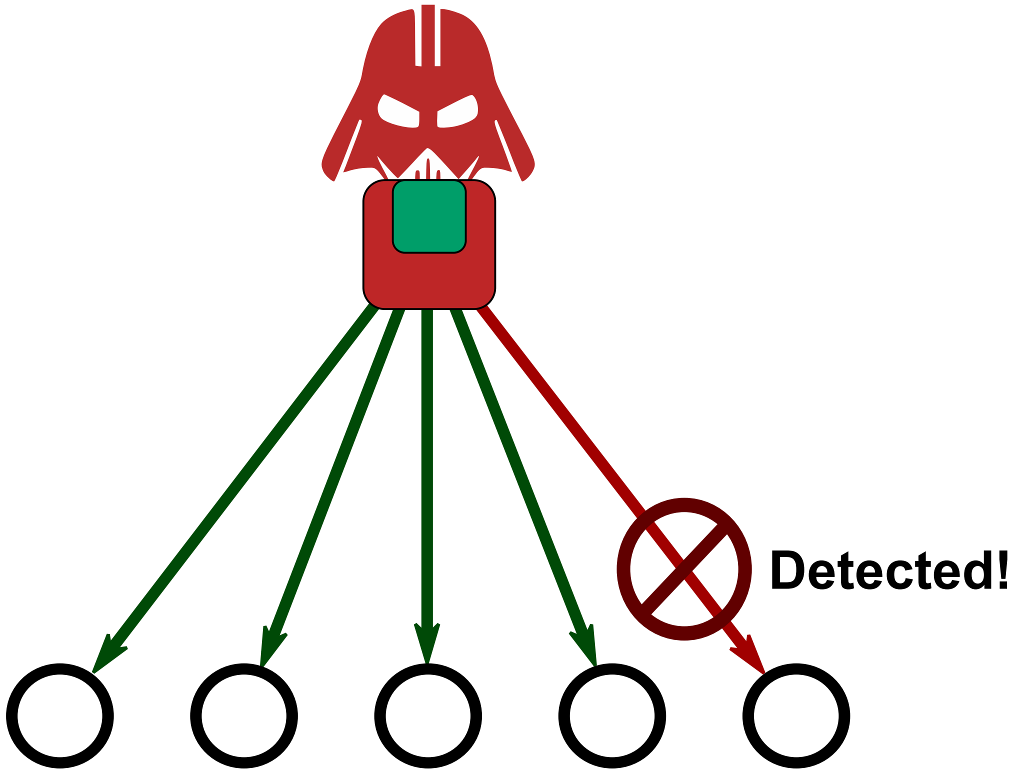
How can TEEs help design scalable consensus?

Problem: Compromised nodes can equivocate

Solution: Use attestation to prevent equivocation!

- Tolerate faults in ½ of the nodes

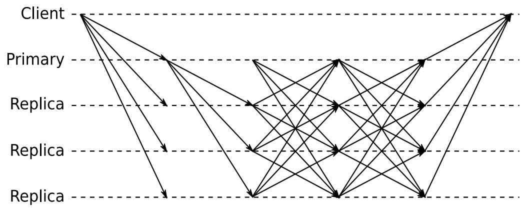
Applicability limited to permissioned settings



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

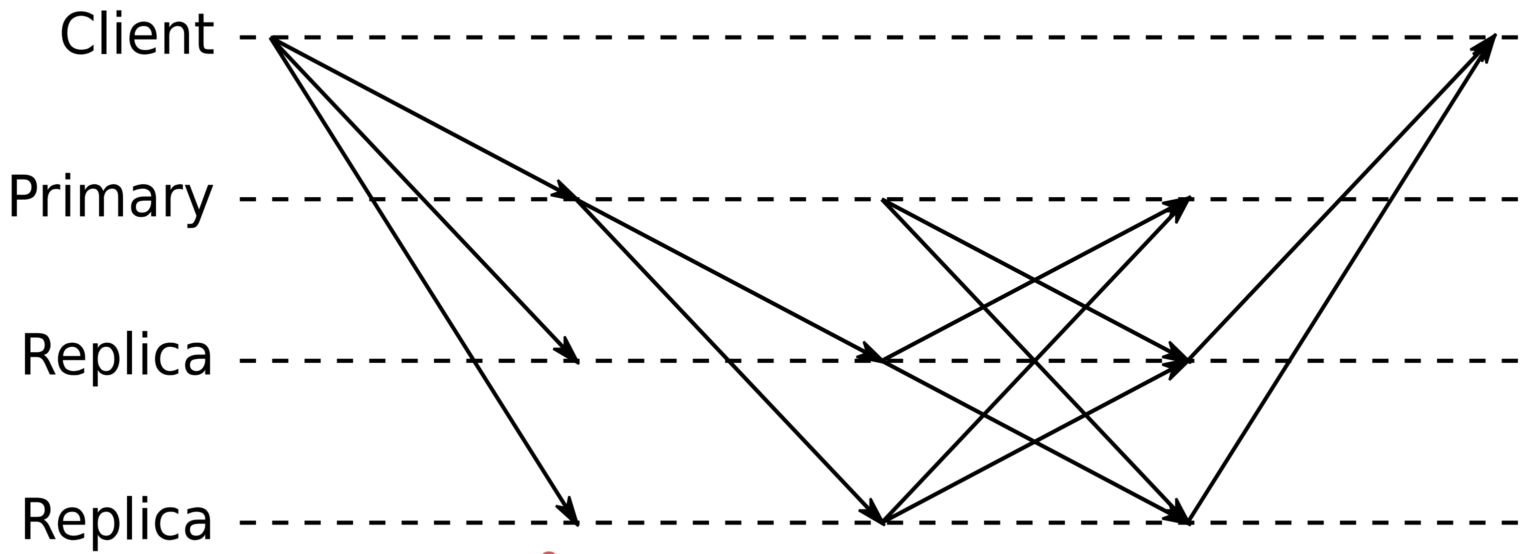
MinBFT

PBFT



Hardware-based monotonic counters

→ increase fault-tolerance

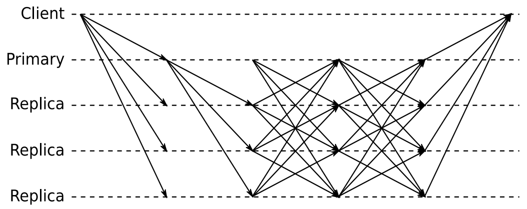


O(n^2) messages, $n = 2f + 1$

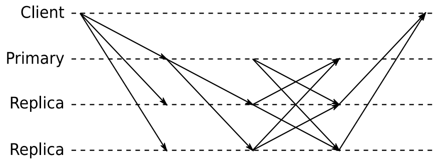
Veronese et al., "Efficient Byzantine fault-tolerance." IEEE Trans. Computers 62.1 (2013): 16–30

FastBFT

PBFT

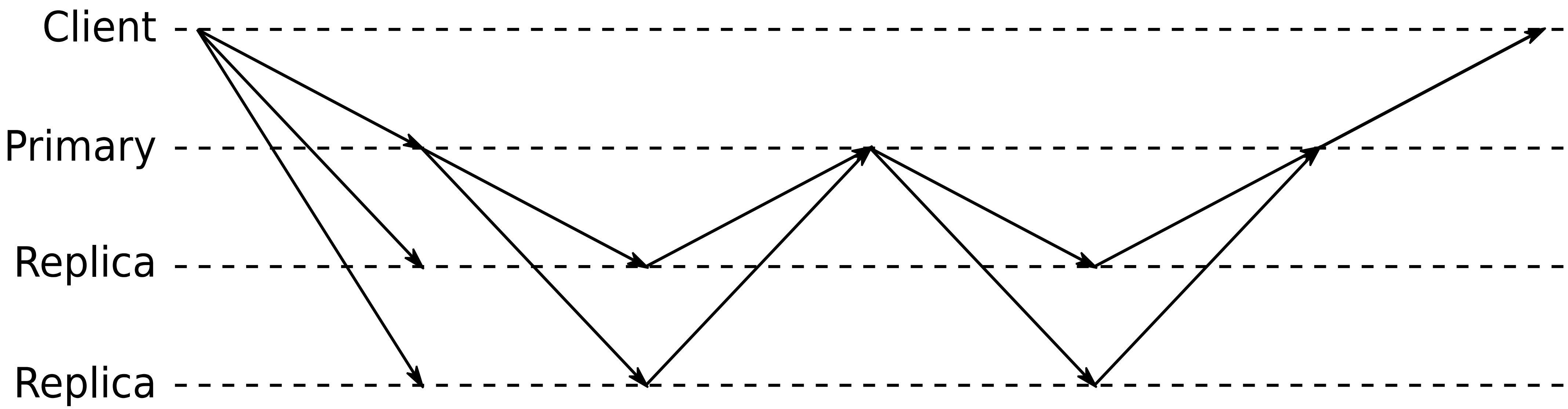


MinBFT



TEE-protected secret sharing, message aggregation

→ increase throughput



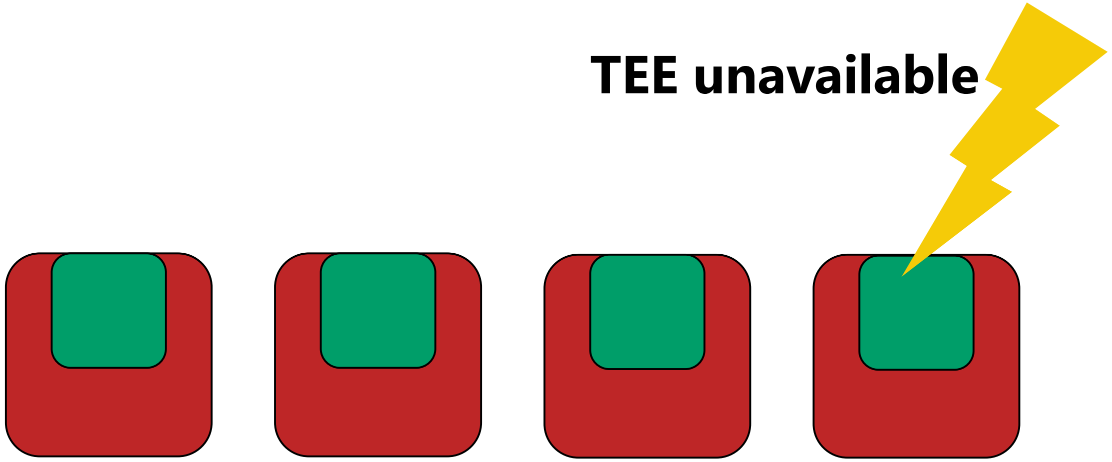
$O(n)$ messages, $n = 2f + 1$

[LLKA18] "Scalable Byzantine Consensus via Hardware-assisted Secret Sharing", IEEE Trans. Computers (2018)

Challenges

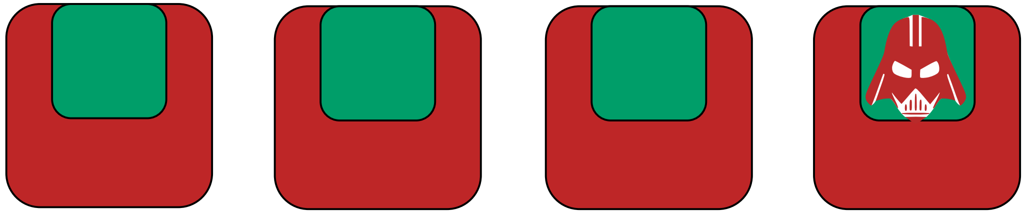
[Skip to TEE compromise](#)

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

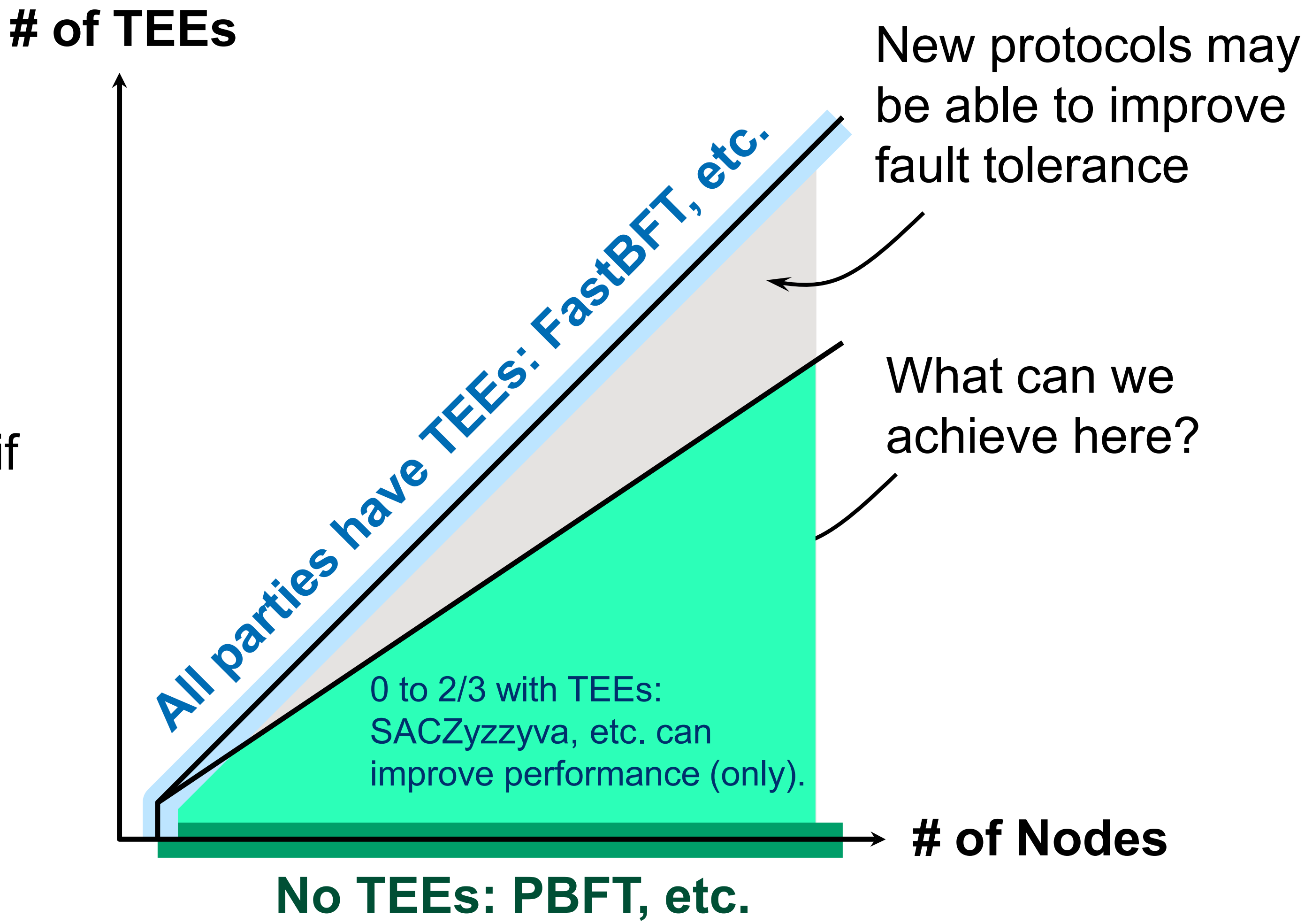
Question: Can we improve consensus protocols by adding only a few TEEs?

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



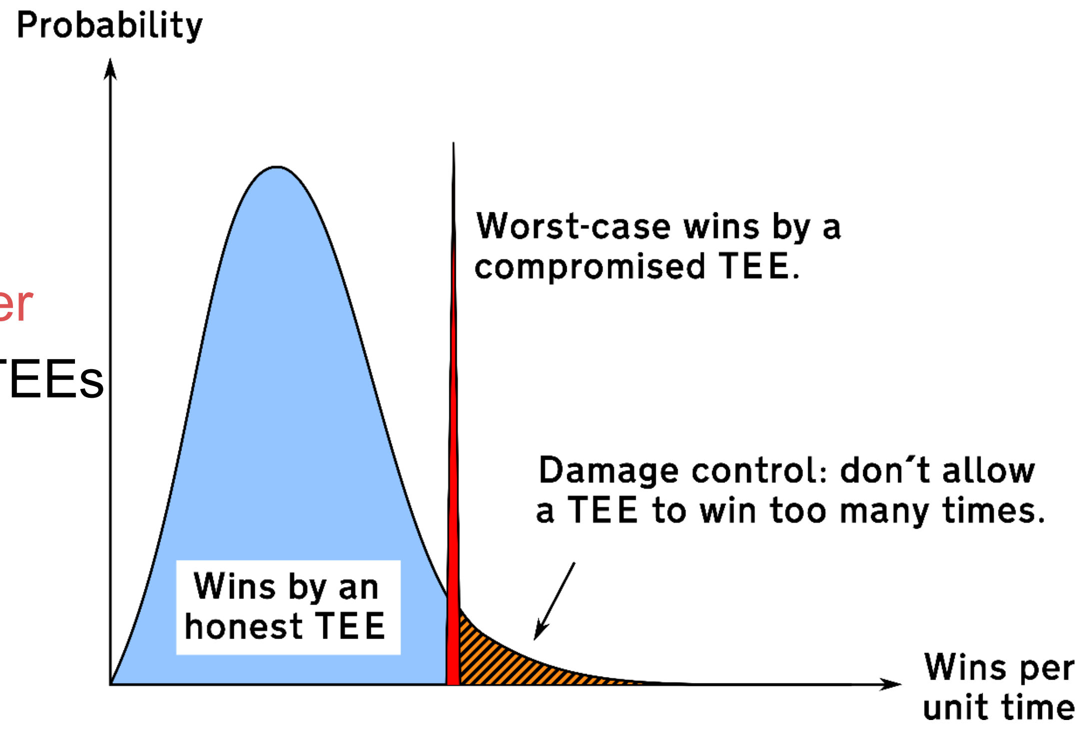
* [GLVA19] SACZyzzyva, SRDS, http://arxiv.org/abs/1905.10255

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



"Design for Failure"

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

Concerns with applicability of hardware-supported TEEs remain

But compelling common-sense applications exist

be practical; protect end-users; address everyday needs

Solutions must incorporate mitigations for:

TEE unavailability or compromise

<u>Design for failure</u>

application- or system-level mitigations possible

https://ssg.aalto.fi/research/projects/bcon/

BCon project, Academy of Finland

http://www.icri-cars.org/

ICRI-CARS, Intel

On dealing with TEE compromise

Two types of settings where TEEs are useful:

1. Improving functionality without compromising security: e.g., PoET
2. Improving security (esp. where none exists today): e.g., SafeKeeper

TEE compromise is a major concern in Type 1 settings

In Type 2 settings, TEE compromise implies returning to current situation