Secure, Confidential Blockchains Providing High Throughput and Low Latency

Lefteris Kokoris-Kogias

EPFL

Lausanne, 03-04-2019

Blockchain, Blockchain, Blockchain

- Bring Transparency in a Digital World
- Minimise the need for globally trusted third parties
- Cheeper and faster transactions

This Thesis

- **Scaling and Performance**: Scaling up blockchains to handle intensive global workloads for both permissionless decentralized blockchains, and permissioned/consortium blockchains supporting >100,000 transactions/sec.
- **Correctness by Design and Construction**: Making it easy, and even automatic, for blockchain developers to produce secure protocols and code, by utilizing (1) programming language techniques to create correct code, and (2) cryptographic protocols with security proofs.
- **Confidentiality**: Combining transparency with confidentiality in blockchains, by utilizing (1) cryptographic techniques, as well as (2) trusted hardware.
- **Authenticated Data Feeds**: Supporting a robust ecosystem of trustworthy data feeds for blockchains and contributing high-trust data feed solutions.
- **Safety and Compliance**: Enabling techniques and protocols for effective monitoring and targeted intervention in blockchains, informed by evaluations of traditional contract law and risks of crime in smart contracts.
- **Sound Migration**: Formulating practical migration paths to production blockchain deployments and enabling integration of new blockchain systems with legacy systems.

Talk Outline

- **Part I : Introduction**
- Part II : Tools for Efficient Decentralization
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC
  - Scalable Bias-Resistant Distributed Randomness
- Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding
- Part IV : Conclusion and Future Work

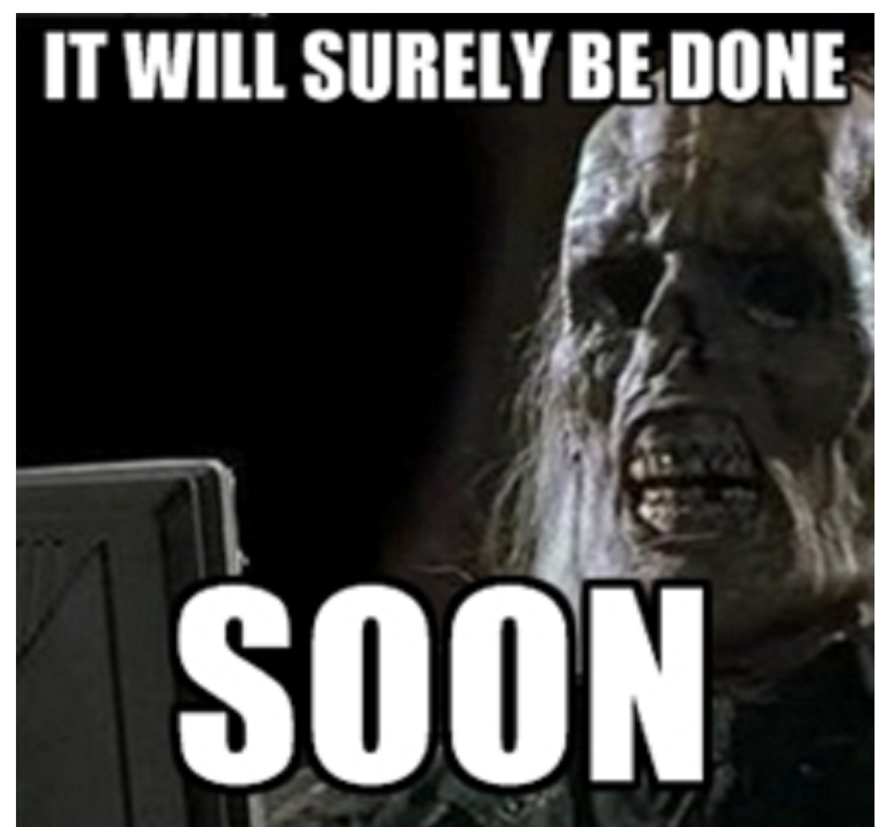
Scaling Blockchains is More Important Than Ever …

CATS RULE THE BLOCKCHAIN, TOO

The ethereum network is getting jammed up because people are rushing to buy cartoon cats on its blockchain

Drawbacks of Nakamoto Consensus

- Transaction confirmation delay
  - Bitcoin: Any tx takes >10 mins until being confirmed
- Weak consistency
  - Bitcoin: You are not really certain your tx is committed until you wait >1 hour
- Low throughput
  - Bitcoin: ~7 tx/sec
- Proof-of-work mining
  - Wastes huge amount of energy



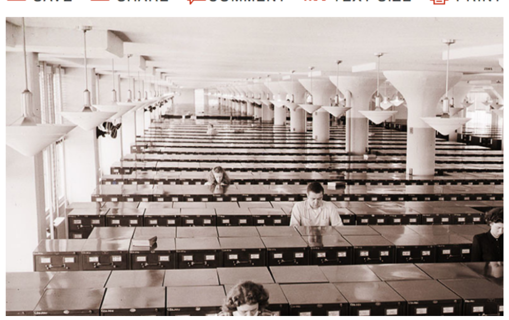
The Promise of Blockchain

The Potential for Blockchain to Transform Electronic Health Records

by John D. Halamka, MD, Andrew Lippman, and Ar

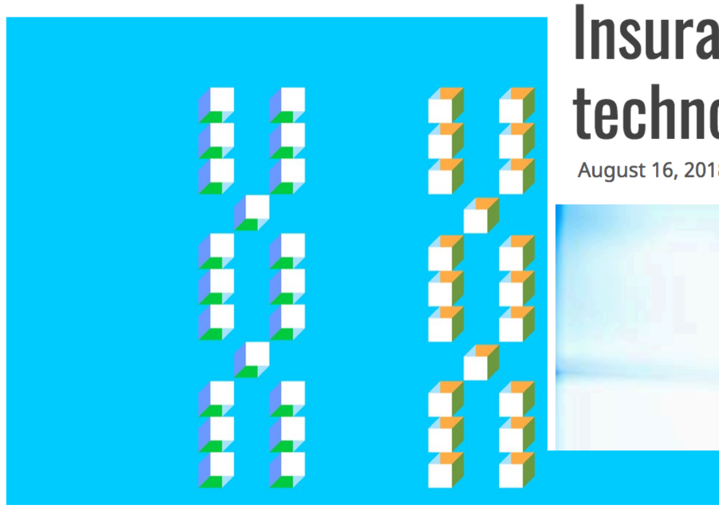
MARCH 03, 2017

SAVE SHARE 0 COMMENT TEXT SIZE PRINT



ADAM ROGERS SCIENCE 02.21.18 07:00 AM

SOLVE GENOMICS WITH THE BLOCKCHAIN? WHY THE HELL NOT



MEET THE MAN WITH A RADICAL PLAN FOR BLOCKCHAIN VOTING

A new movement says that crypto-voting can purify democracy—and eventually eliminate the need for governments altogether.

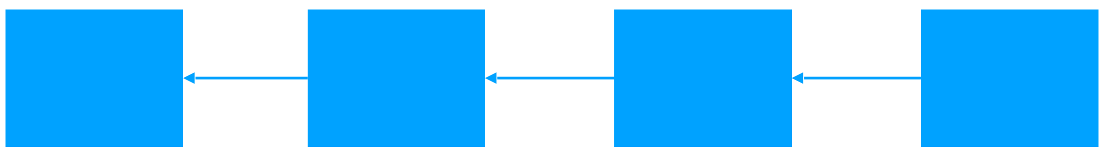
BY ANDREW LEONARD

IN A CAFÉ on the Upper East Side of Manhattan, a one-time videogame developer turned political theorist named Santiago Siri is

Insurance Companies start experimenting with Blockchain technology

August 16, 2018

The Promise of Blockchain



Transparent Decentralized Log

Post encryptions, store keys on cloud

This Thesis

	Scalability	Confidentiality
Open	Oakland ’17,’18 Sec ’16, ’17 HotPETs ‘16	Under Submission
Permissioned	ESORICS ’18	

This Thesis

	Scalability	Confidentiality
Open	Oakland ’17,’18 Sec ’16, ’17 HotPETs ‘16	Under Submission
Permissioned	ESORICS ’18	

This Talk

Talk Outline

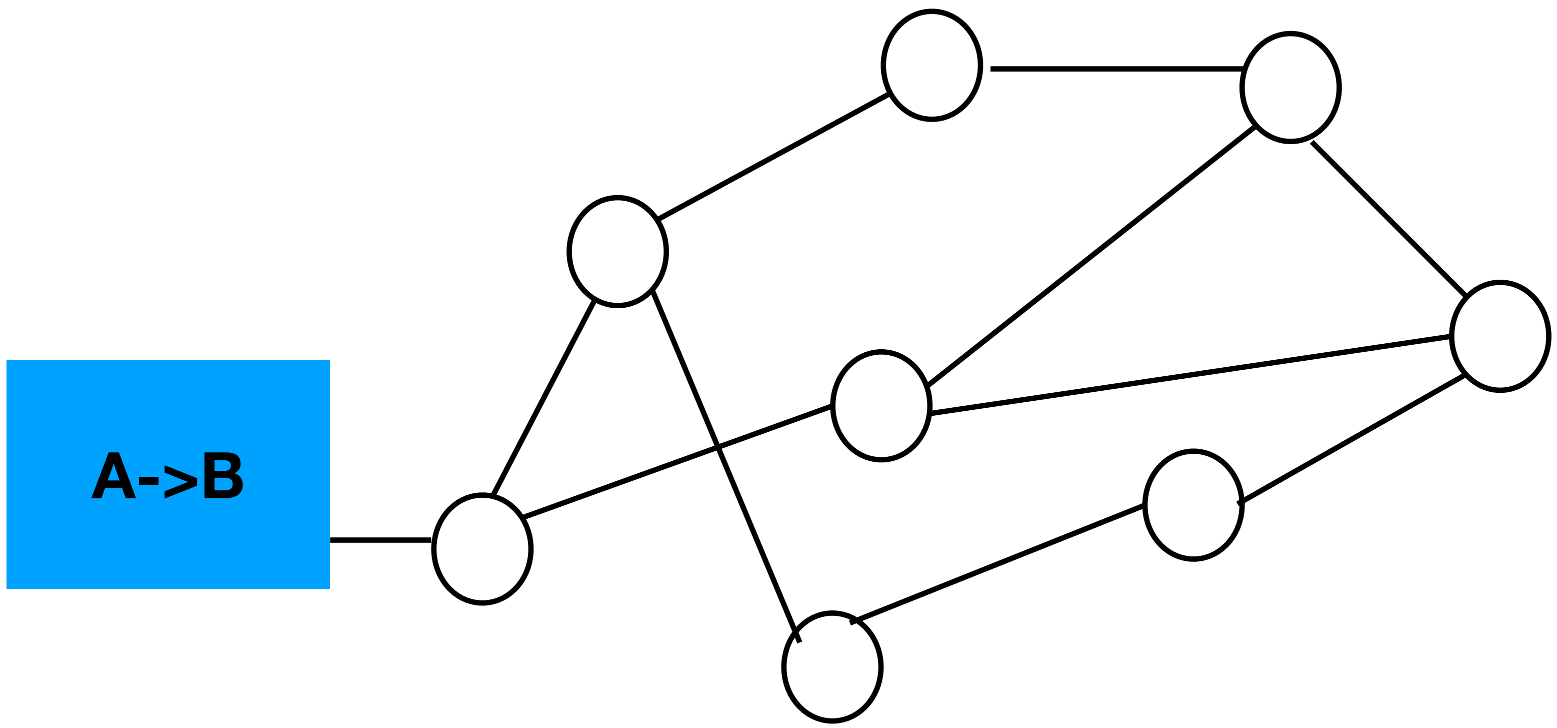
- Part I : Introduction
- **Part II : Tools for Efficient Decentralization**
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC
  - Scalable Bias-Resistant Distributed Randomness
- Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding
- Part IV : Conclusion and Future Work

Chapter Outline

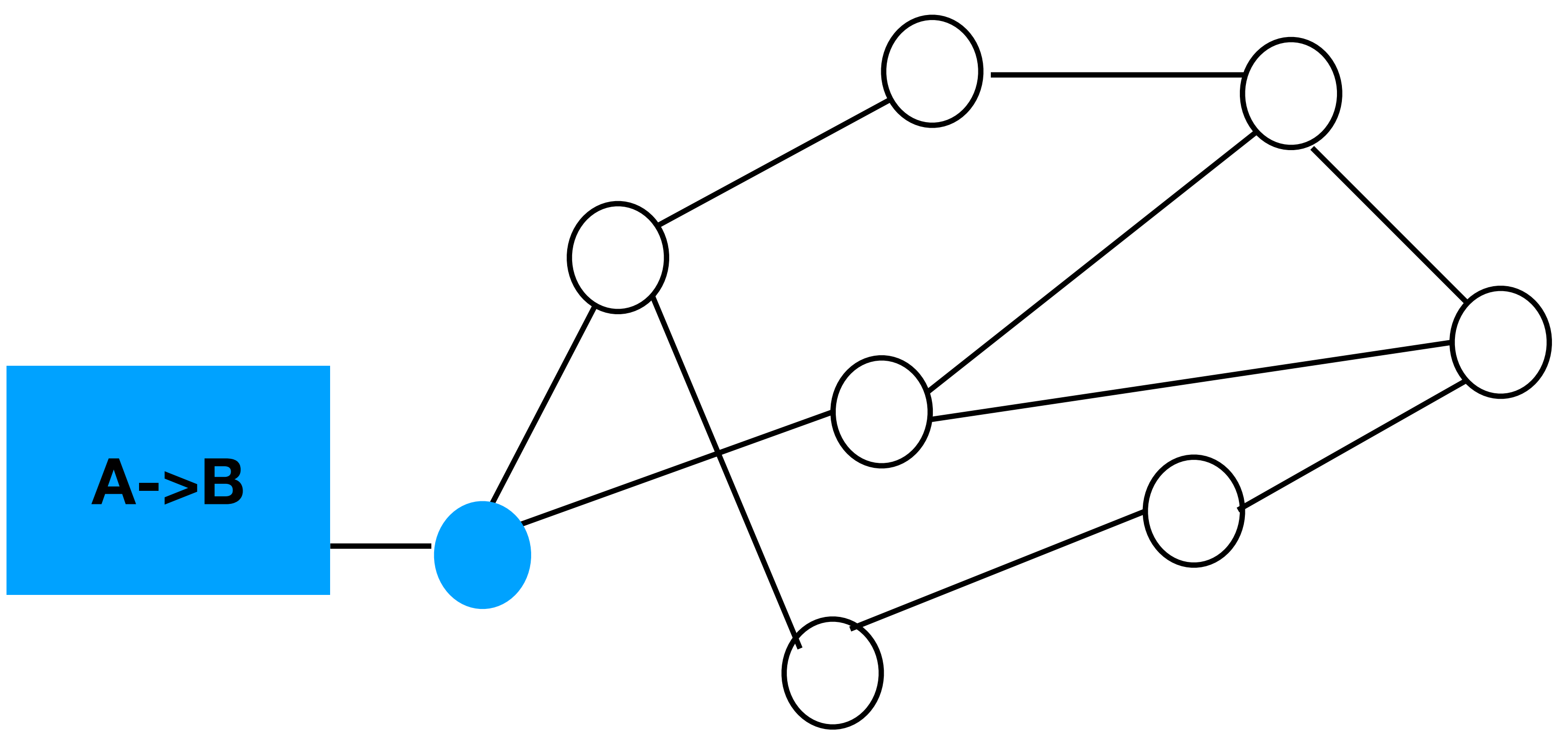
- **Bitcoin and its limitations**
- Strawman design: PBFTCoin
- Opening the consensus group
- From MACs to Collective Signing
- Decoupling transaction verification from leader election
- Performance Evaluation

*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16'

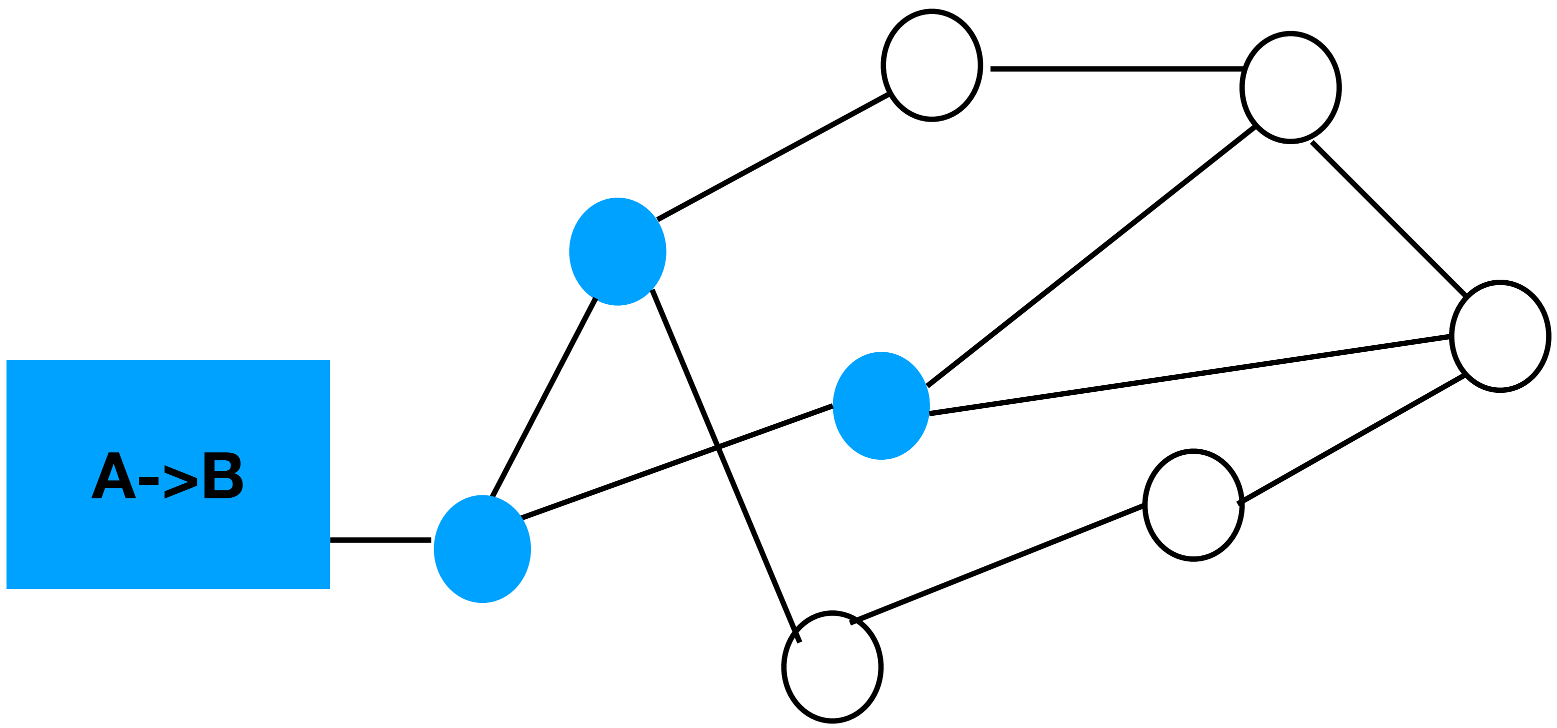
Transaction Verification in Bitcoin



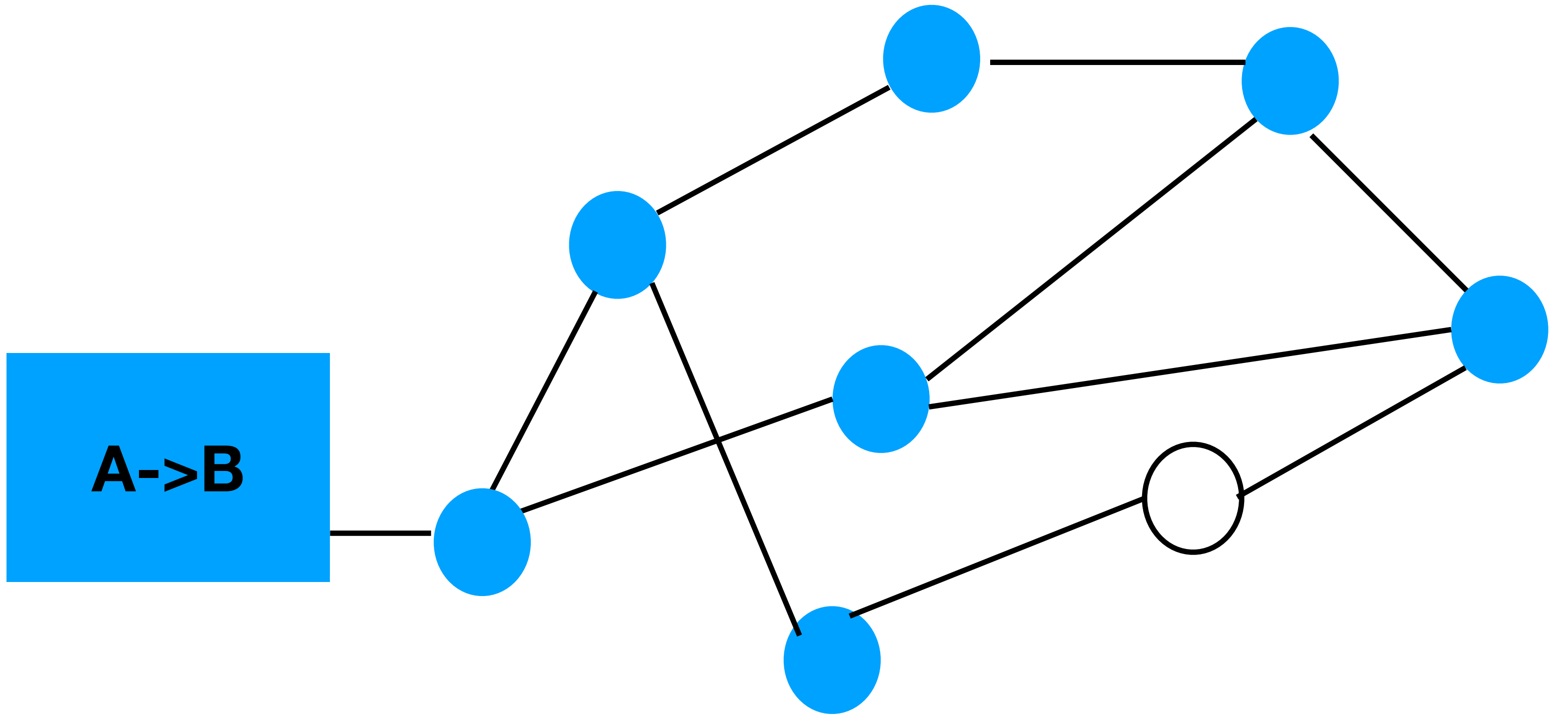
Transaction Verification in Bitcoin



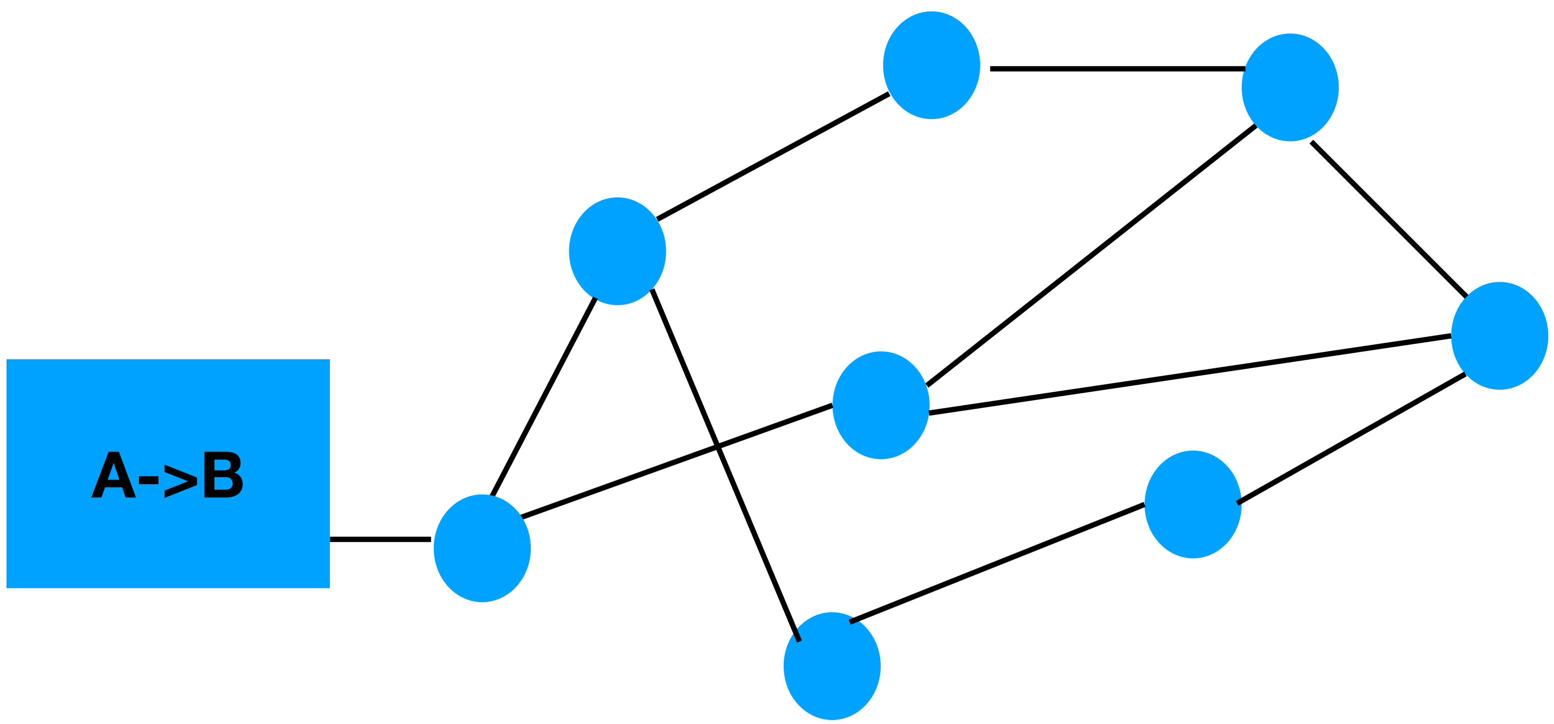
Transaction Verification in Bitcoin



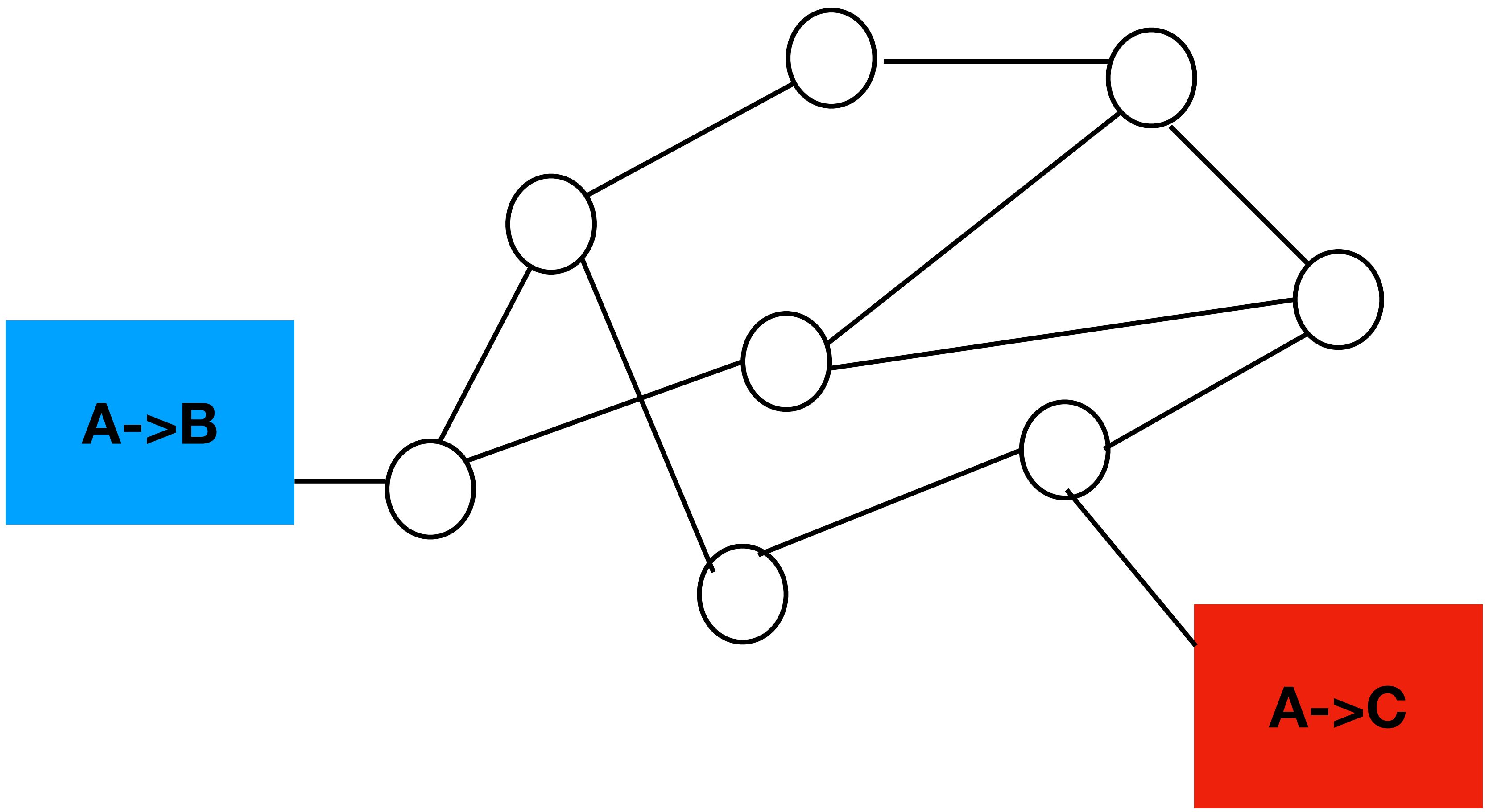
Transaction Verification in Bitcoin



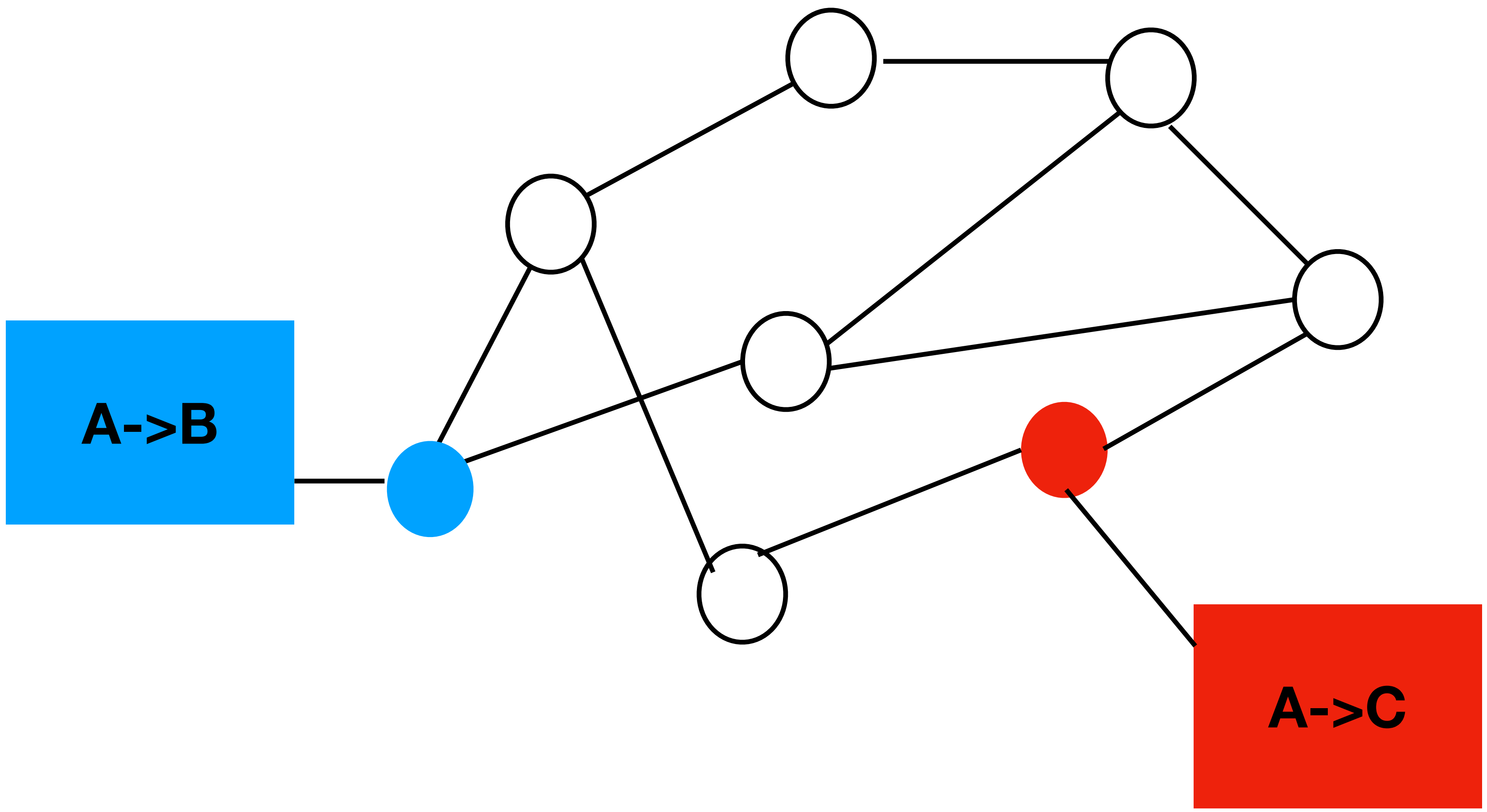
Transaction Verification in Bitcoin



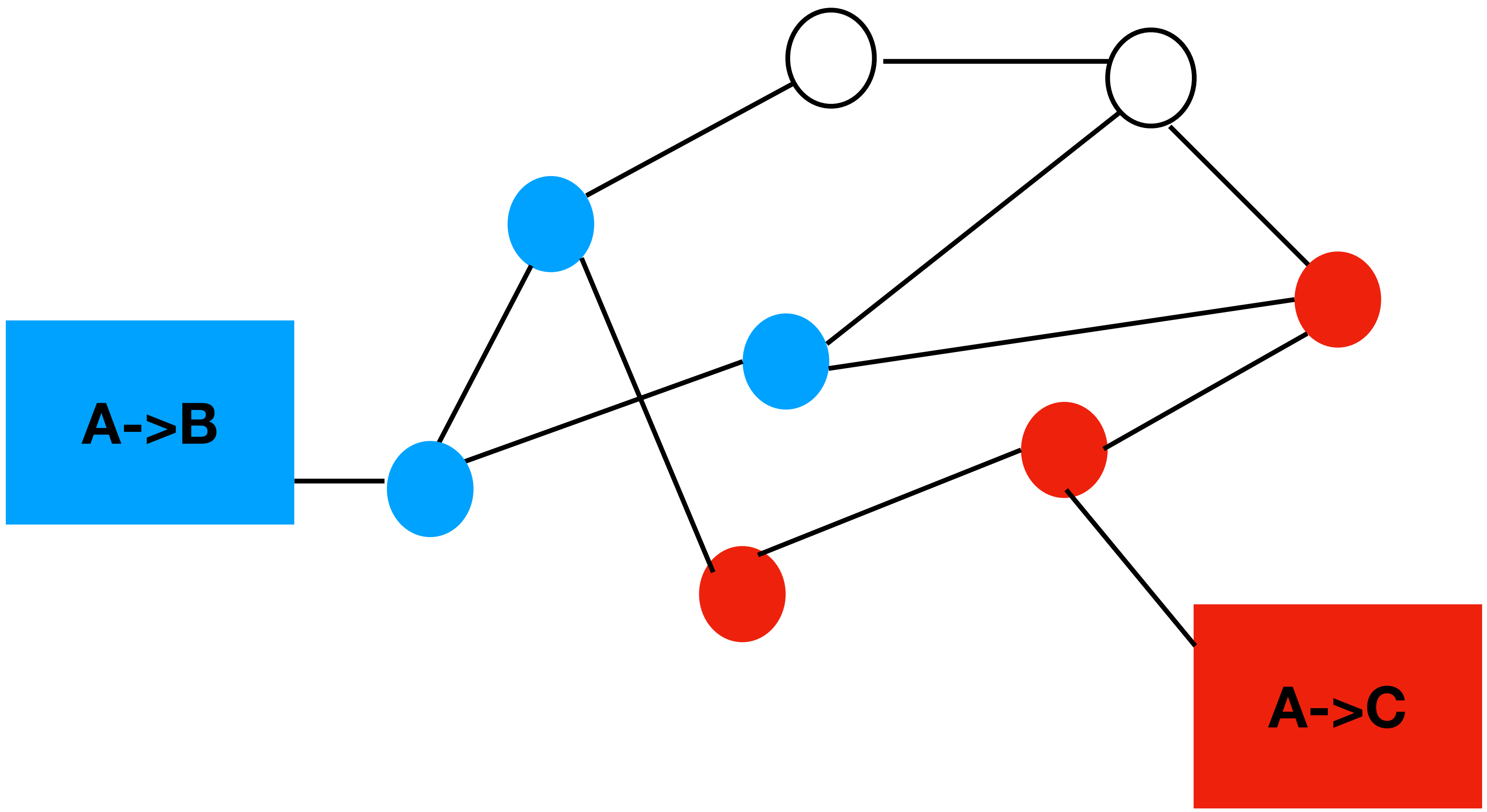
Transaction Verification in Bitcoin



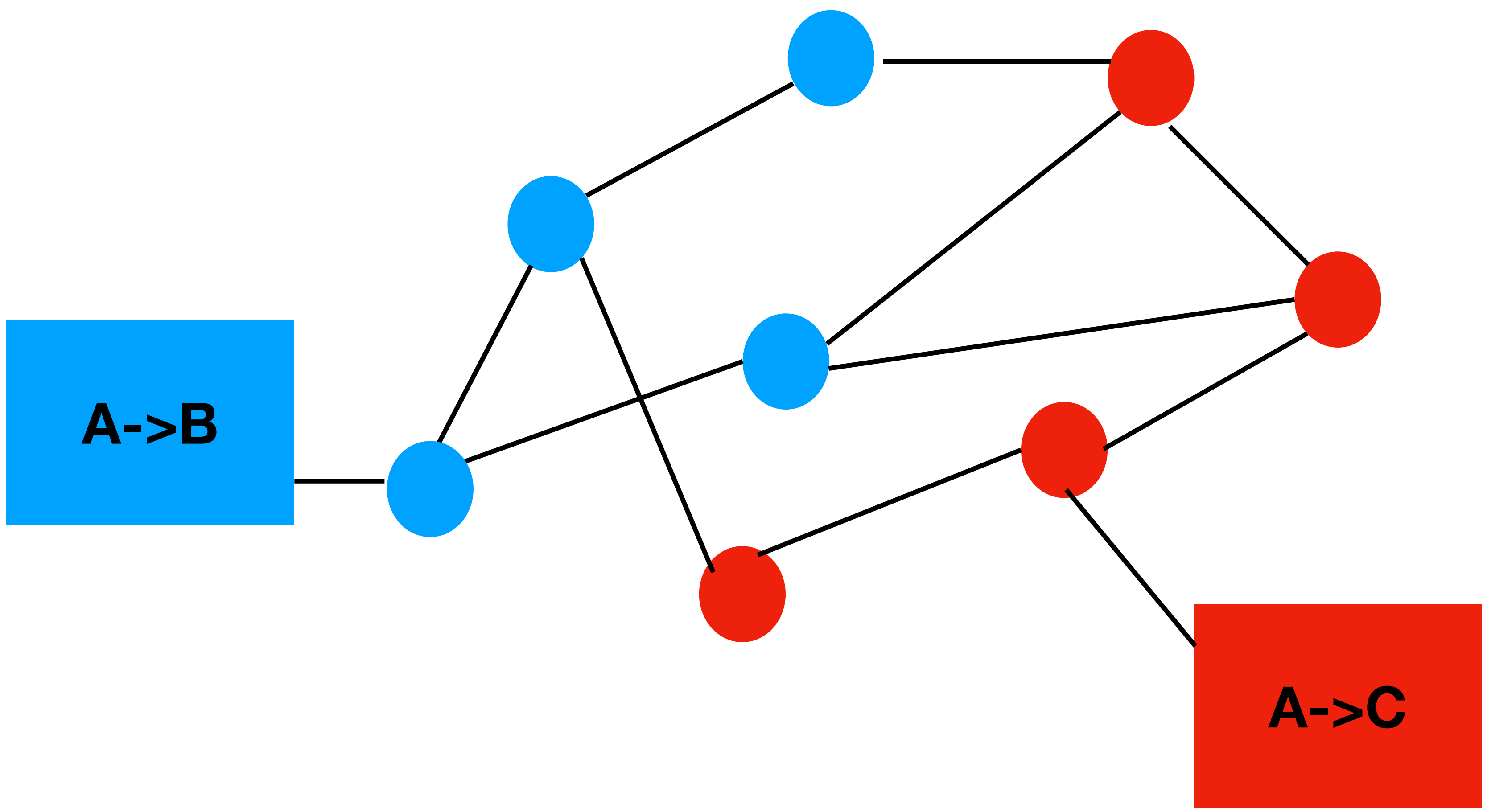
Transaction Verification in Bitcoin



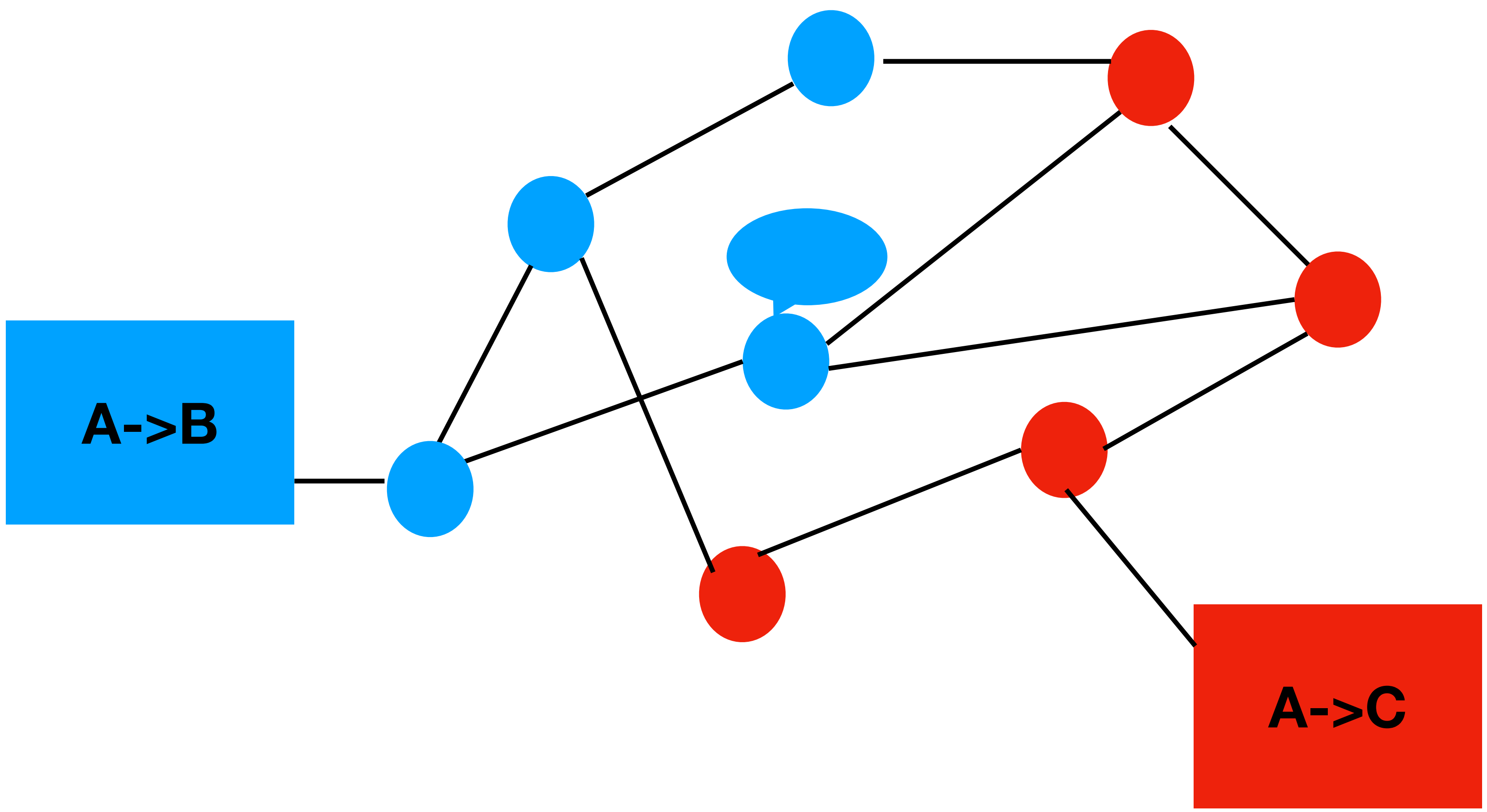
Transaction Verification in Bitcoin



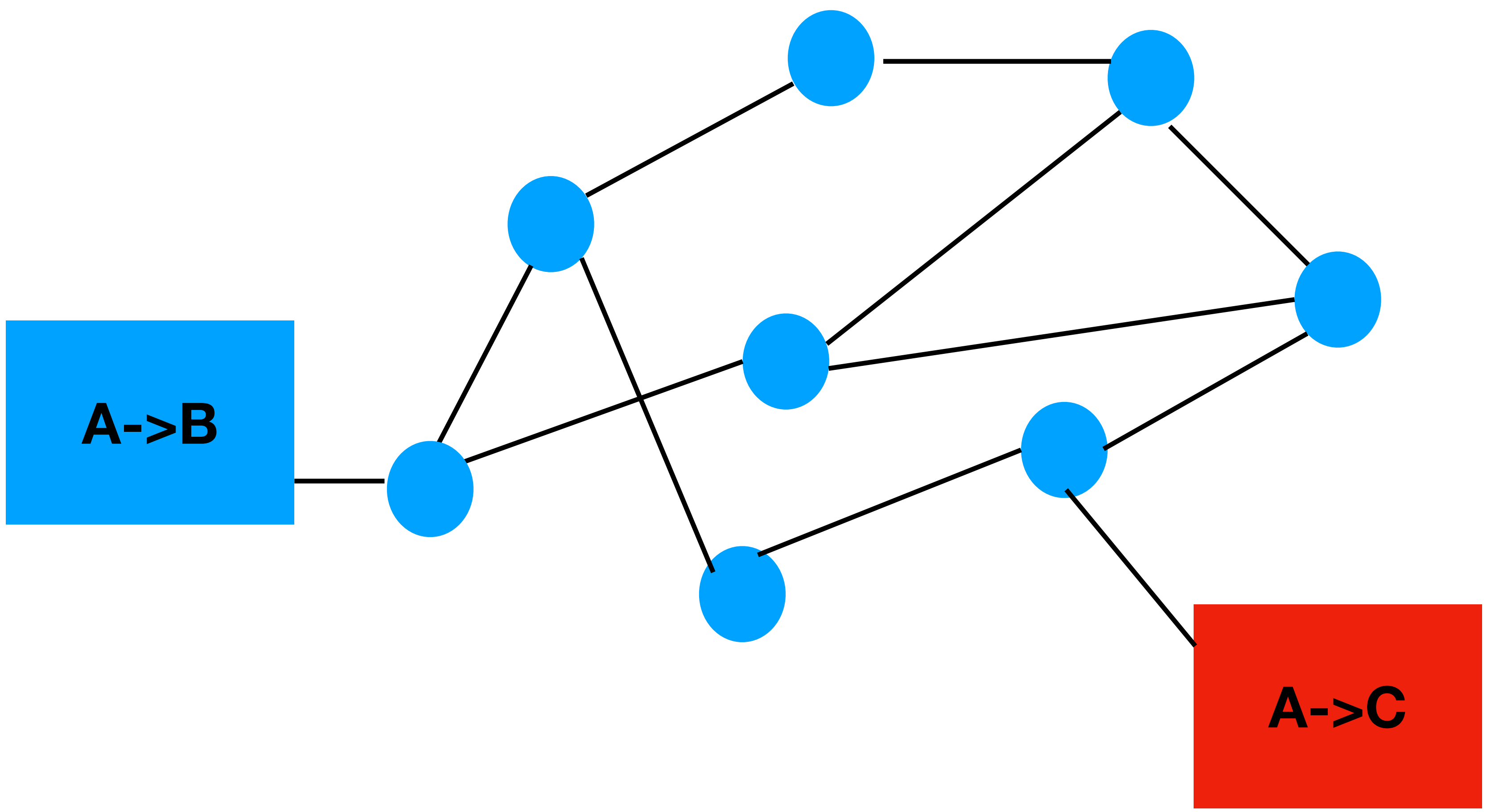
Transaction Verification in Bitcoin



Transaction Verification in Bitcoin

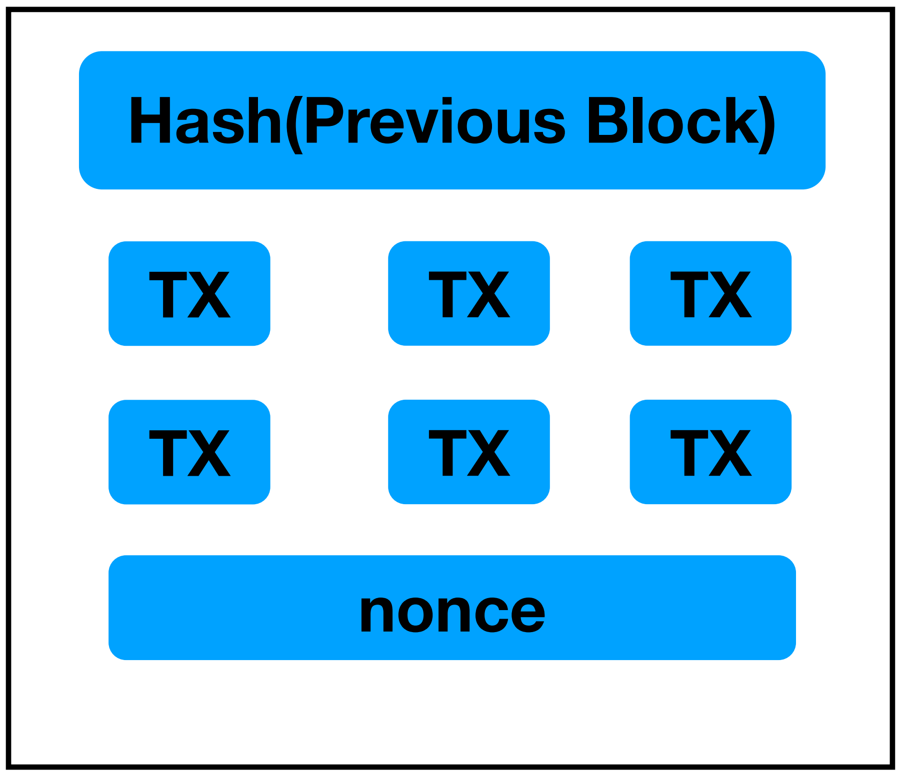


Transaction Verification in Bitcoin



Proof-of-Work

BLOCK



H(Block, nonce=0) =abc3426fe31233

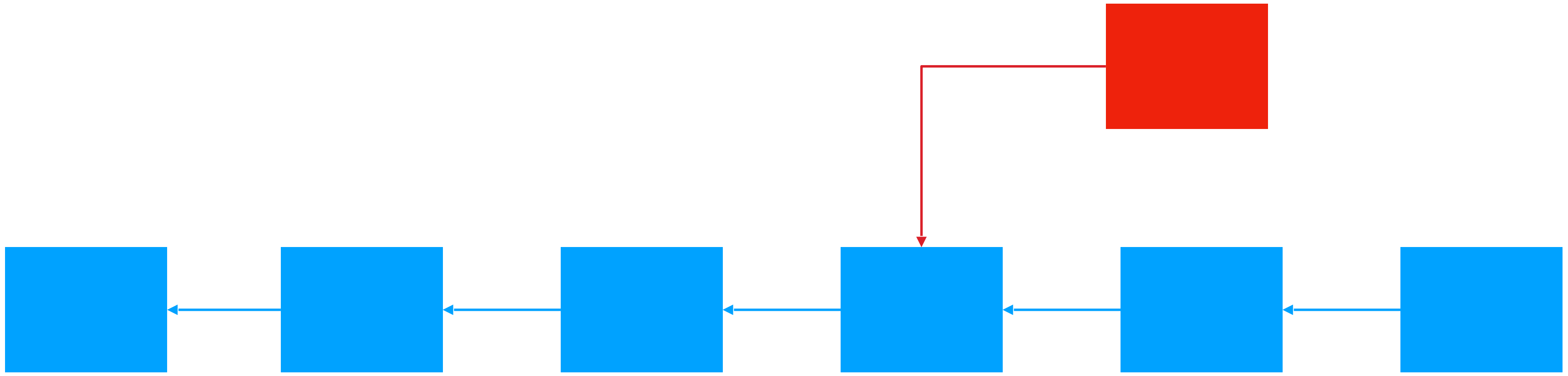
H(Block, nonce=1) =fe541200abc229

H(Block, nonce=2) =0bc3429831233

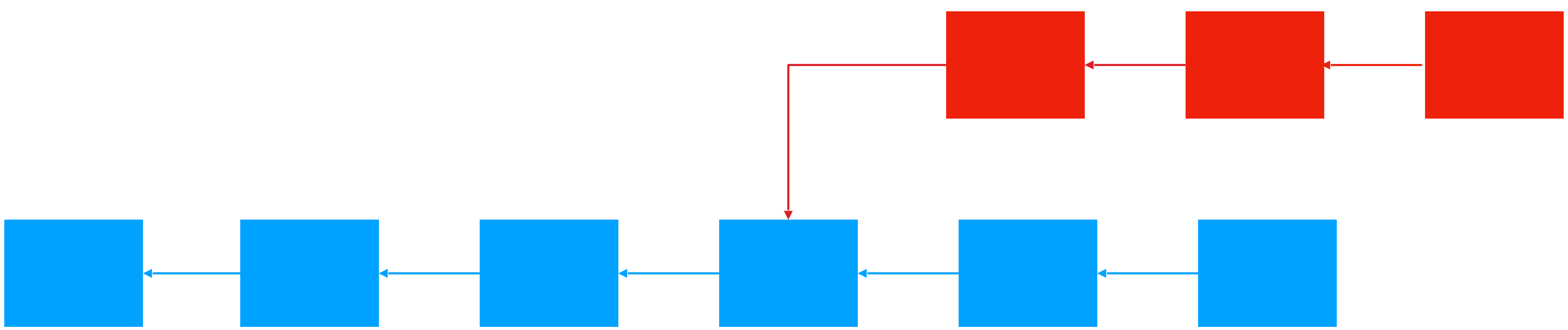
⋮

H(Block, nonce=29) =0000fed98312

The Blockchain



The Blockchain



Problem Statement

- In Bitcoin there is no verifiable commitment of the system that a block will persist
  - Clients rely on probabilities to gain confidence
  - Probability of successful fork-attack decreases exponentially

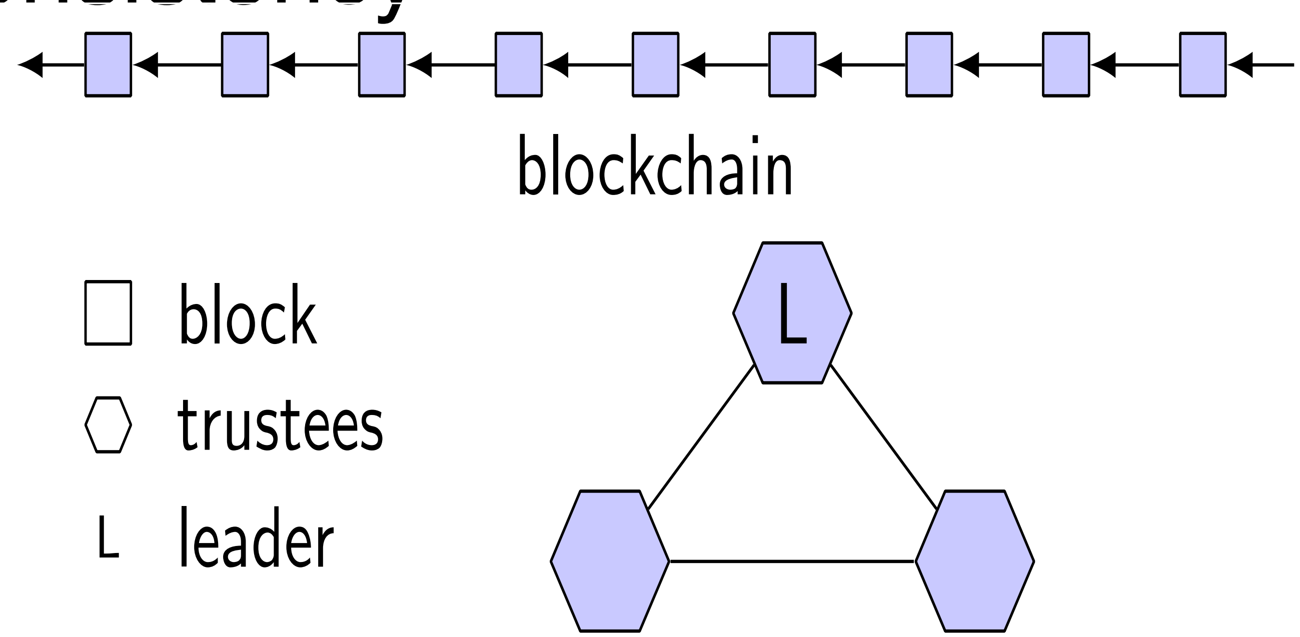
Chapter Outline

- Bitcoin and its limitations
- **Strawman design: PBFTCoin**
- Opening the consensus group
- From MACs to Collective Signing
- Decoupling transaction verification from leader election
- Performance Evaluation

*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16'

Strawman Design: PBFTCoin

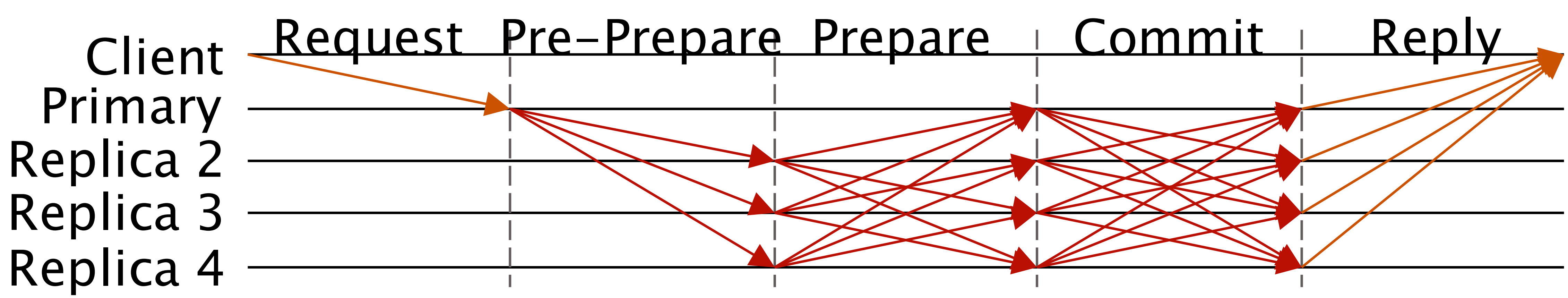
- 3f+1 fixed "trustees" running PBFT* to withstand **f** failures
- Non-probabilistic strong consistency
  - Low latency
- No forks/inconsistencies
  - No double-spending



*Practical Byzantine Fault Tolerance [Castro/Liskov]

Strawman Design: PBFTCoin

- Problem: Needs a static consensus group
- Problem: Scalability
  - $O(n^2)$ communication complexity
  - $O(n)$ verification complexity
  - Absence of third-party verifiable proofs (due to MACs)



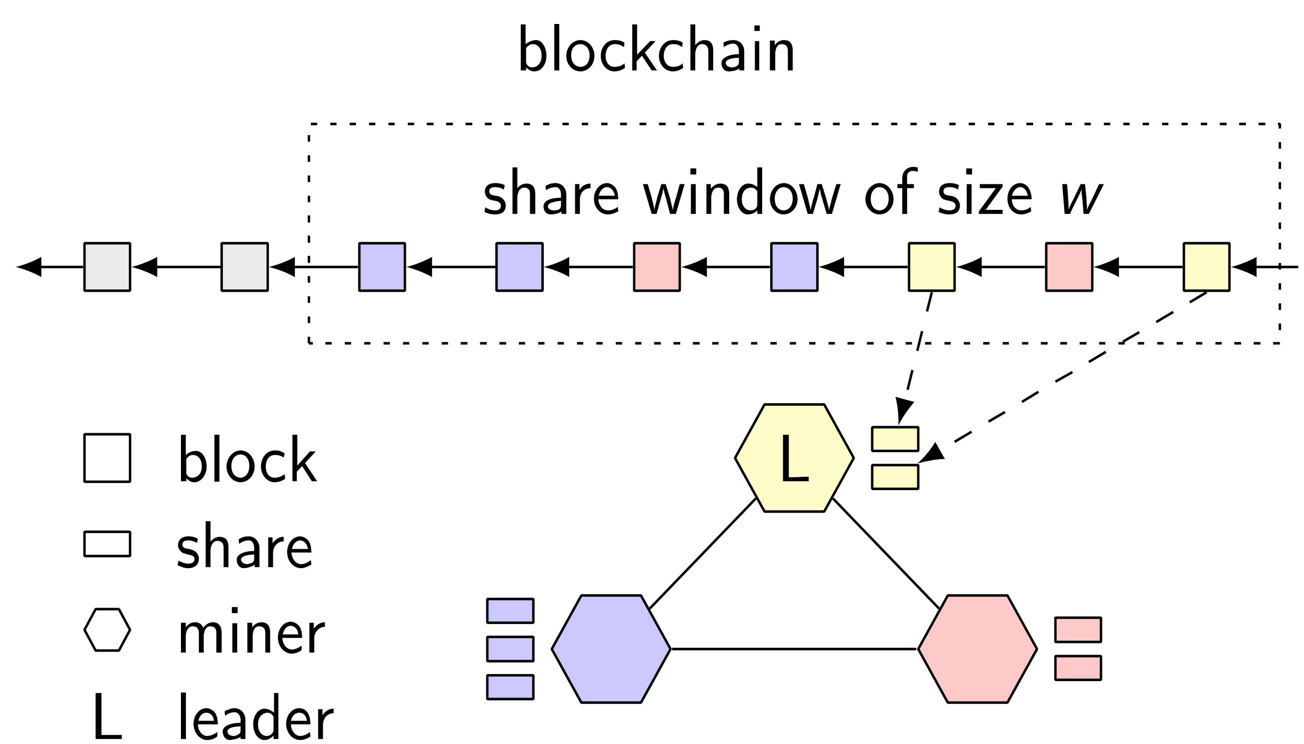
Chapter Outline

- Bitcoin and its limitations
- Strawman design: PBFTCoin
- **Opening the consensus group**
- From MACs to Collective Signing
- Decoupling transaction verification from leader election
- Performance Evaluation

*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16’

Opening the Consensus Group

- PoW against Sybil attacks
- One share per block
  - % of shares ∝ hash-power
- Window mechanism
  - Protect from inactive miners



Chapter Outline

- Bitcoin and its limitations
- Strawman design: PBFTCoin
- Opening the consensus group
- **From MACs to Collective Signing**
- Decoupling transaction verification from leader election
- Performance Evaluation

*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16’

From MACs to Signing

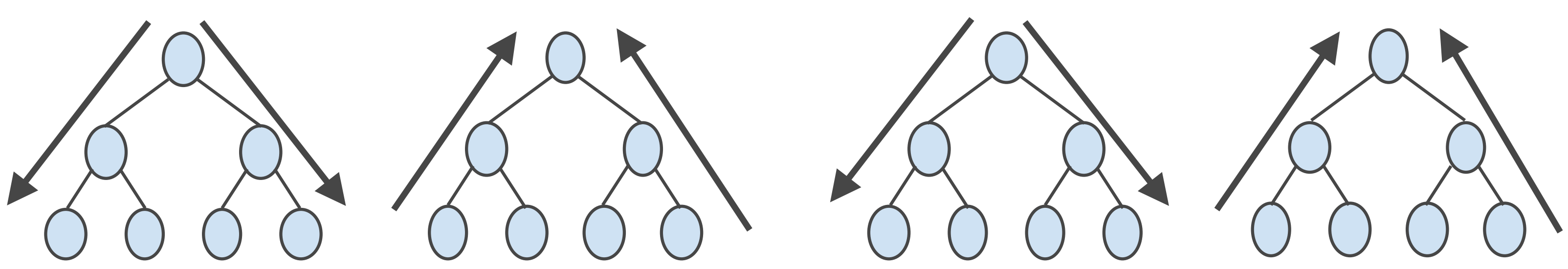
- Substitute MACs with public-key cryptography
  - Third-party verifiable
  - PoW Blockchain as PKI
  - Enables sparser communication patterns (ring or star topologies)

From MACs to Collective Signing

- Can we do better than O(n) communication complexity?
  - Multicast protocols transmit information in O(log n)
  - Use trees!!
- Can we do better than O(n) complexity to verify?
  - Schnorr multisignatures could be verified in O(1)
  - Use aggregation!!
- Schnorr multisignatures + communication trees
  = Collective Signing [Syta et all, IEEE S&P '16]

CoSi

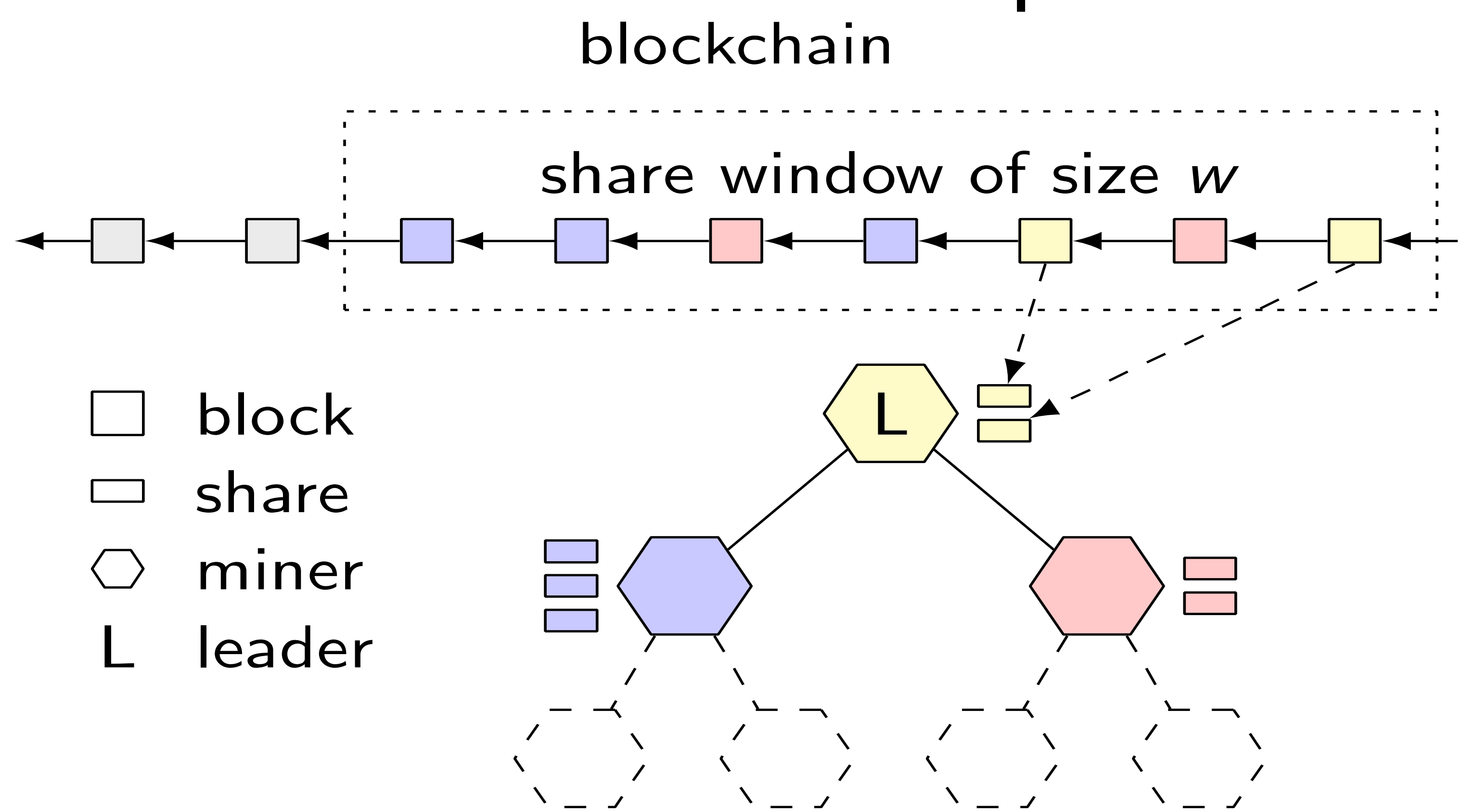
- Efficient collective signature, verifiable as a simple signature
  - 80 bytes instead of 9KB for 144* co-signers (Ed25519)



* Number of ~10-minute blocks in 1-day time window

Discussion

- CoSi is not a BFT protocol
- PBFT can be implemented over two subsequent CoSi rounds
  - Prepare round
  - Commit round



Problem Statement

- In ~~Bitcoin~~ ByzCoin there is ~~no~~ a verifiable commitment of the system that a block will persist
- Throughput is limited by forks
  - Increasing block size increases fork probability
  - Liveness exacerbation

Chapter Outline

- Bitcoin and its limitations
- Strawman design: PBFTCoin
- Opening the consensus group
- From MACs to Collective Signing
- **Decoupling transaction verification from leader election**
- Performance Evaluation

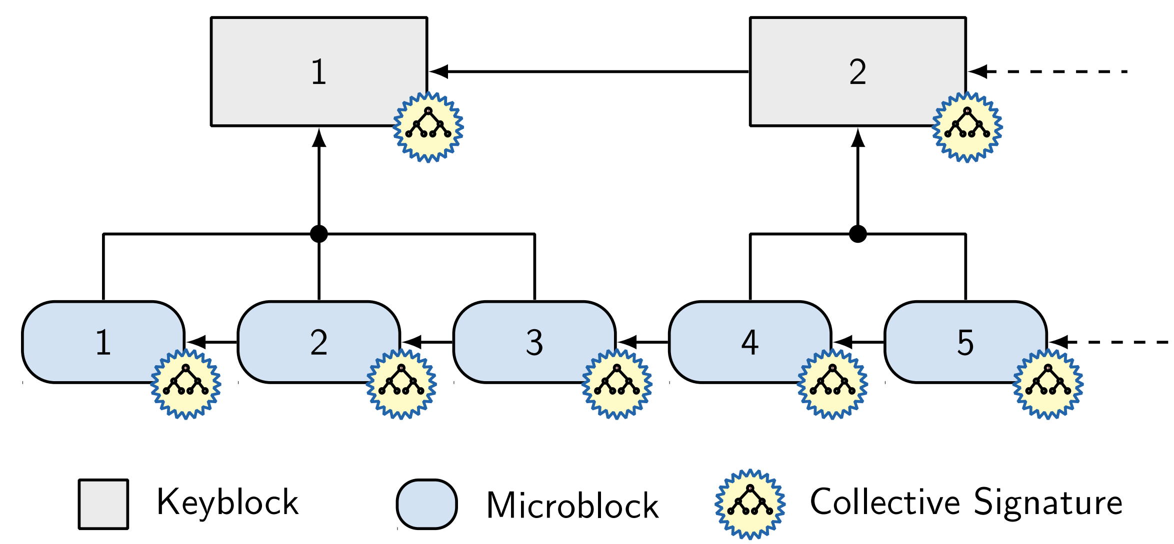
*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16’

Bitcoin-NG [Eyal et all, NSDI ’16]

- Makes the observation that block mining implement two distinct functionalities
  - Transaction verification
  - Leader election
- But, Bitcoin-NG inherits many of Bitcoin’s problems
  - Double-spending
  - Leader is checked after his epoch ends

Decoupling Transaction Verification from Leader Election

- Key blocks:
  - PoW & share value
  - Leader election
- Microblocks:
  - Validating client transactions
  - Issued by the leader



Chapter Outline

- Bitcoin and its limitations
- Strawman design: PBFTCoin
- Opening the consensus group
- From MACs to Collective Signing
- Decoupling transaction verification from leader election
- **Performance Evaluation**

*Enhancing bitcoin security and performance with strong consistency via collective signing, Sec 16'

Performance Evaluation

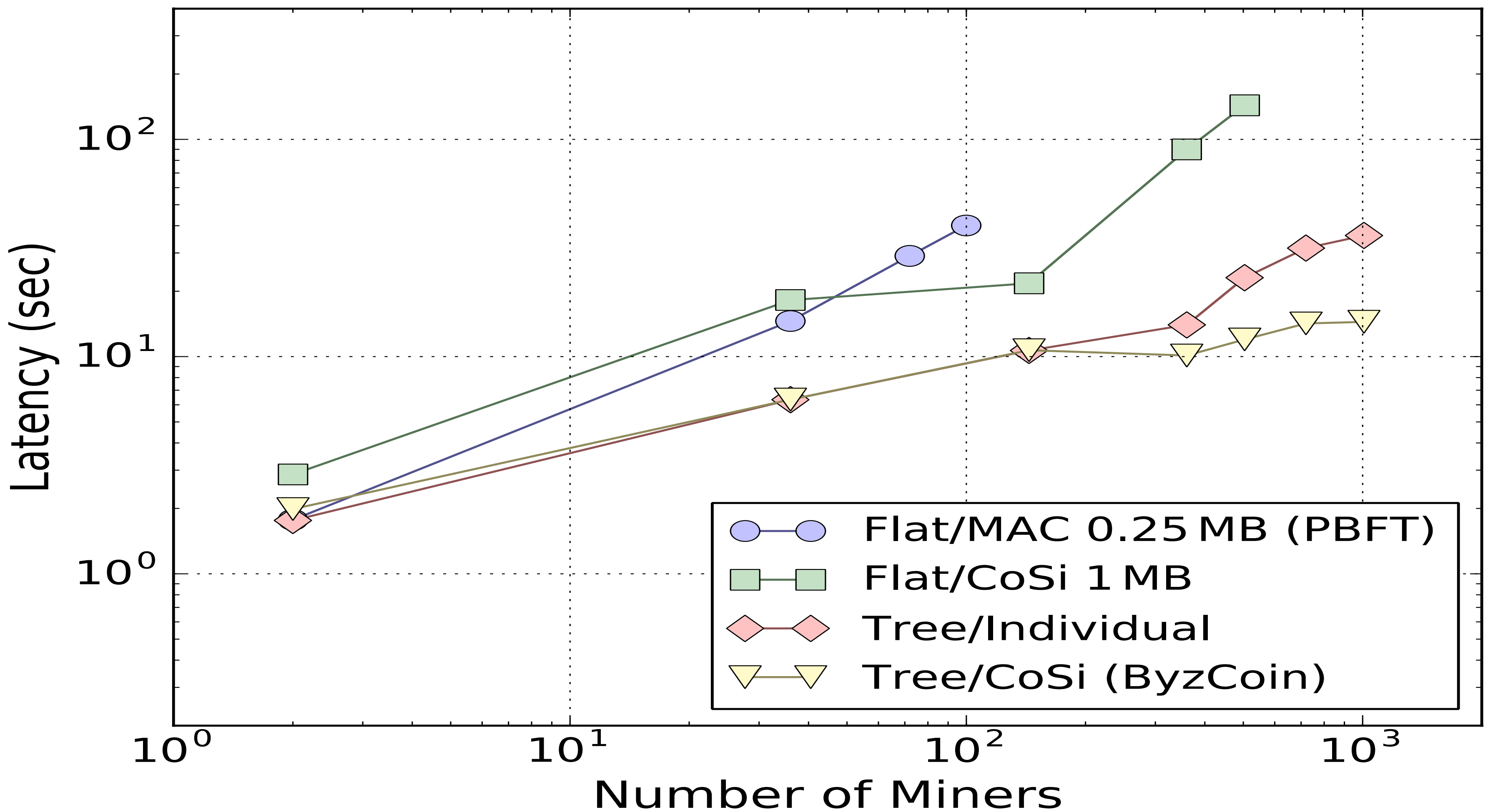
- Experiments run on DeterLab network testbed
  - Up to 1,008* miners multiplexed atop 36 machines
  - Impose 200 ms roundtrip latencies between all servers
  - Impose 35 Mbps bandwidth per miner

*** 1008 = # of ~10-minute key-blocks in 1-week time window**

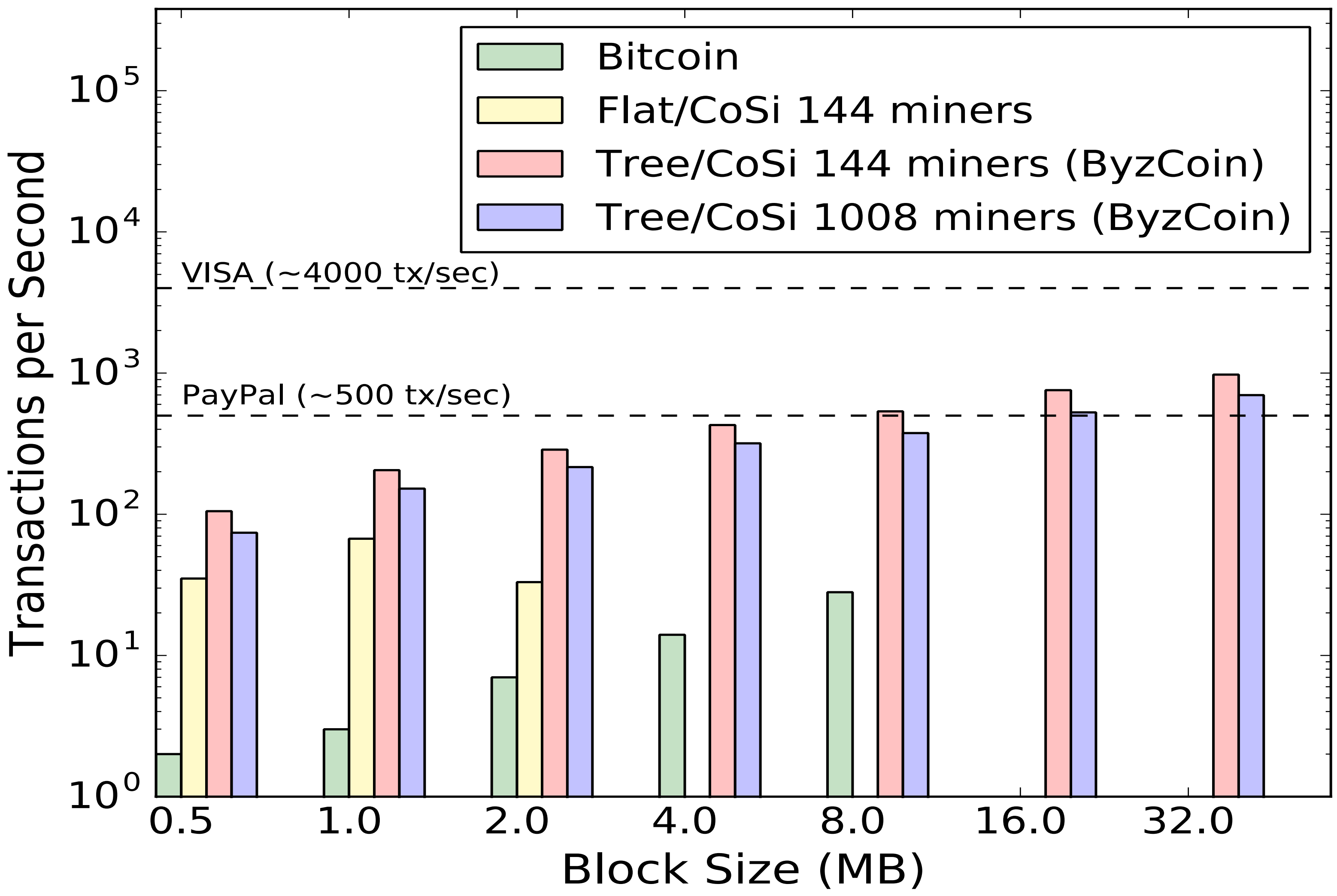
Performance Evaluation

- Key questions to evaluate:
  - What size consensus groups can ByzCoin scale to?
  - What transaction throughput can it handle?

Consensus Latency



Throughput



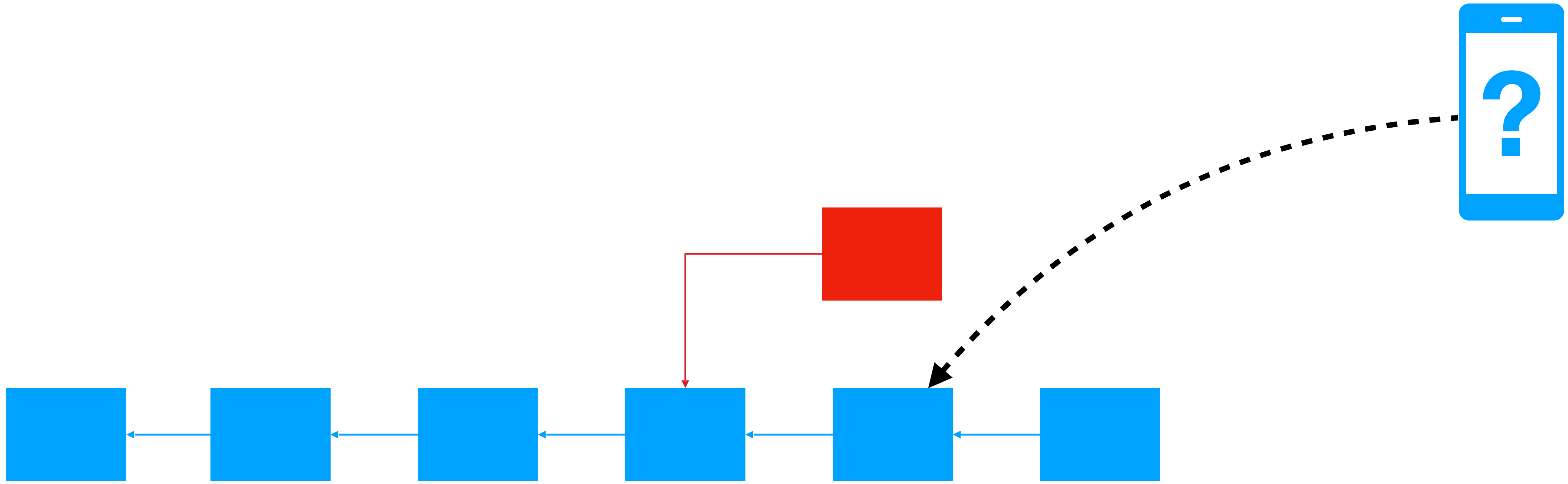
Bitcoin
Flat/CoSi 144 miners
Tree/CoSi 144 miners (ByzCoin)
Tree/CoSi 1008 miners (ByzCoin)
VISA (~4000 tx/sec)
PayPal (~500 tx/sec)
Transactions per Second
10^0
10^1
10^2
10^3
10^4
10^5
0.5
1.0
2.0
4.0
8.0
16.0
32.0
Block Size (MB)

Talk Outline

- Part I : Introduction
- **Part II : Tools for Efficient Decentralization**
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - **Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC**
  - Scalable Bias-Resistant Distributed Randomness
- Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding
- Part IV : Conclusion and Future Work

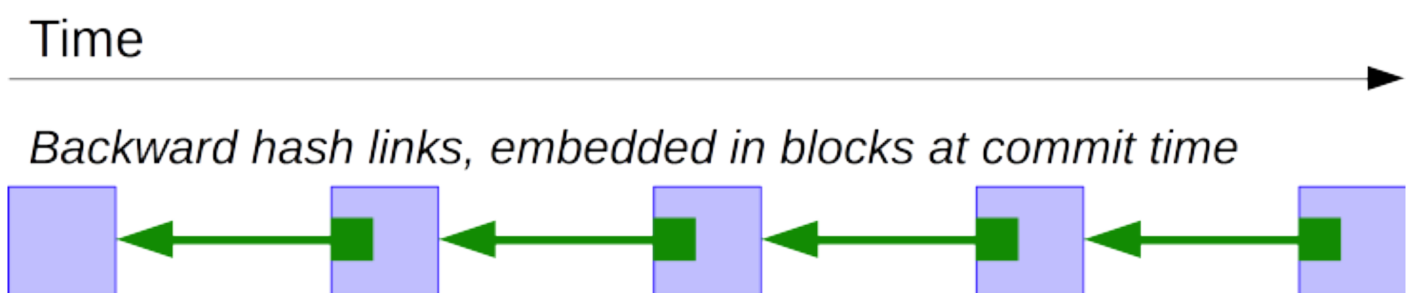
Problem: Efficient Verification

- How does a “light” (low-power, mobile) client securely confirm a recent (or old) transaction?
    - Especially after being offline for months, years?
    - Without “just trusting” central party (exchange)?

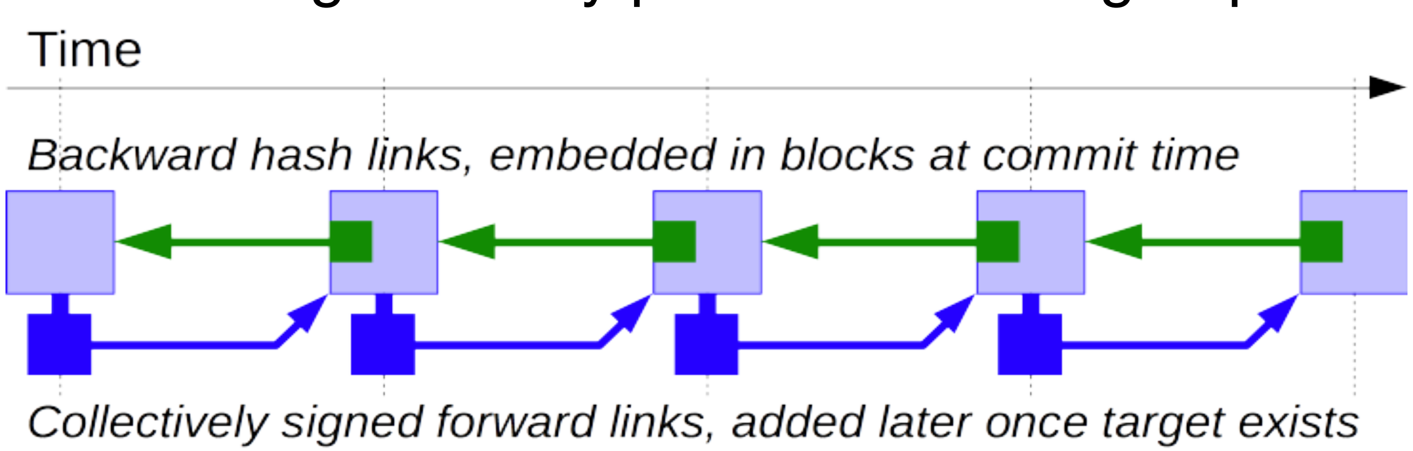


Backward and Forward Verifiability

- Standard blockchains traversable only **backward**
  - Via hash back-links from current head

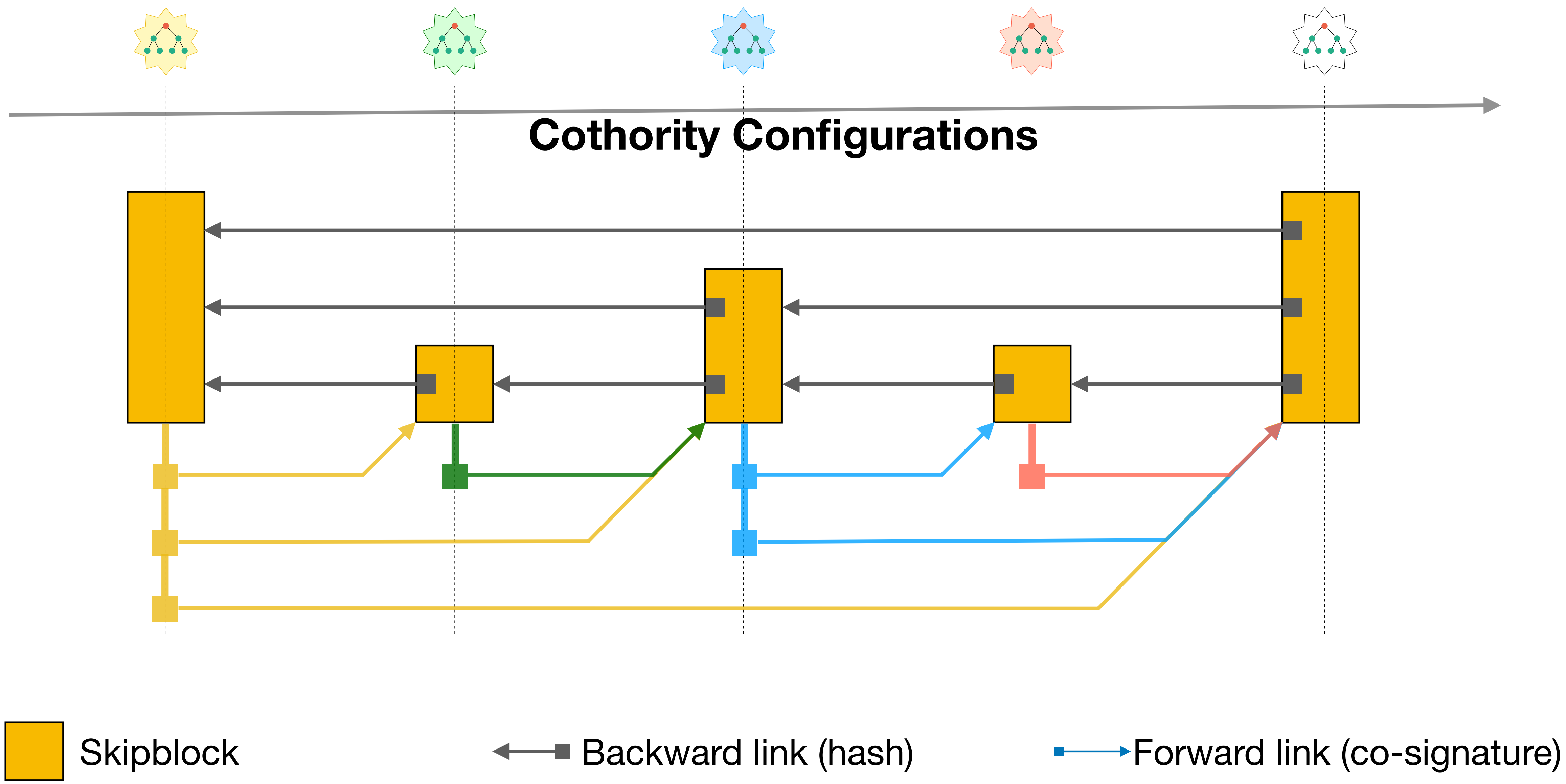


- We add traversability **forward in time***
  - Collective signature by prior consensus group



*Managing identities using blockchains and CoSi, HotPETs 16’

Skipchains


Cothority Configurations
Skipblock
Backward link (hash)
Forward link (co-signature)

Applications of SkipChains

- Enable Offline/P2P verification
  - Works even if Internet is unavailable, slow, costly
- Broad applications
  - Software/key updates
  - Blockchain-Attested Degrees, Awards, …
  - Chain-of-Custody, Bills of Lading, …

Talk Outline

- Part I : Introduction
- **Part II : Tools for Efficient Decentralization**
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC
  - **Scalable Bias-Resistant Distributed Randomness**
- Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding
- Part IV : Conclusion and Future Work

Chapter Outline

- **Motivation**
  - The need for public randomness
  - Strawman examples: Towards unbiasable randomness
- RandHound
- Implementation and Experimental Results

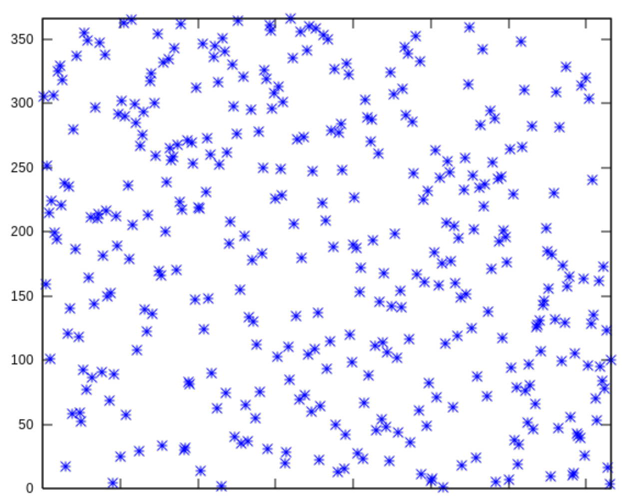
*Scalable Bias-Resistant Distributed Randomness, Oakland ’17

Public Randomness

- **Collectively** used
- Unpredictable ahead of time
- Not secret past a certain point in time
- Applications
  - **Random selection:** lotteries, sweepstakes, jury selection, voting and election audits
  - **Games:** shuffled decks, team assignments
  - **Protocols:** parameters, IVs, nonces, sharding
  - **Crypto:** challenges for NZKP, authentication protocols, cut-and-choose methods, “nothing up my sleeves” numbers

Failed / Rigged Randomness

Vietnam War Lotteries

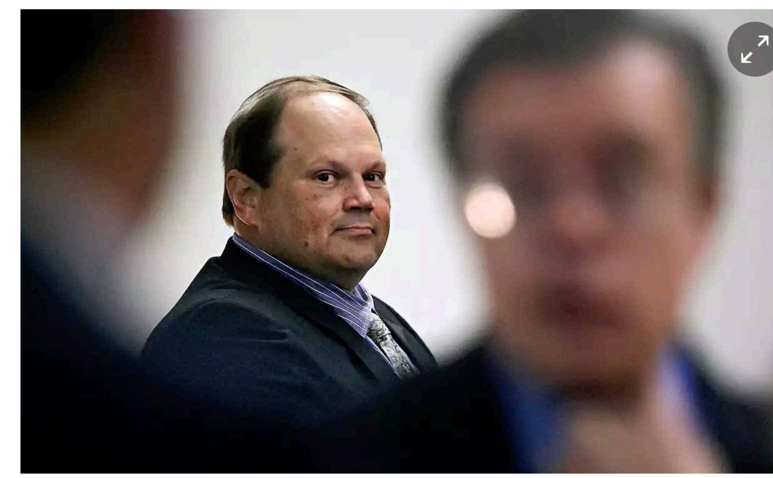


'European draws have been rigged': Ex-FIFA president Sepp Blatter claims to have seen hot and cold balls used to aid cheats

Former FIFA president Sepp Blatter said he had witnessed rigged draws for European football competitions

Man hacked random-number generator to rig lotteries, investigators say

New evidence shows lottery machines were rigged to produce predictable jackpot numbers on specific days of the year netting millions in winnings



'Computer whiz' rigged lottery number generator to produce predictable numbers a couple of times a year. Photograph: Brian Powers/AP

Goals

Decentralized, public randomness in the *(t,n)*-threshold security model

1. Availability

Successful protocol termination for up to *f=t-1* malicious nodes.

2. Unpredictability

Output not revealed prematurely.

3. Unbiasability

Output distributed uniformly at random.

4. Verifiability

Output correctness can be checked by third parties.

5. Scalability

Executable with hundreds of participants.

Assumptions: n= 3f +1, Byzantine adversary and asynchronous network with eventual message delivery

Public Randomness is Hard

	Availability	Unpredictability	Unbiasability	Verifiability	Scalability
Strawman I	−	−	−	−	−
Strawman II	−	✓	−	−	−
Strawman III	✓	✓	✓	−	−

Strawman I

- **Idea:** Combine random inputs of all participants.
- **Problem:** Last node controls output.

Strawman II

- **Idea:** Commit-then-reveal random inputs.
- **Problem:** Dishonest nodes can choose not to reveal.

Strawman III

- **Idea:** Secret-share random inputs.
- **Problem:** Dishonest nodes can send bad shares.

Public Randomness is Hard

	Availability	Unpredictability	Unbiasability	Verifiability	Scalability
Strawman I	✗	✗	✗	✗	✗
Strawman II	✗	✓	✗	✗	✗
Strawman III	✓	✓	(✓)	✗	✗
RandShare	✓	✓	✓	✗	✗

RandShare

- **Idea:** Strawman III + *verifiable secret sharing* (Feldman, 1987)
- **Problems:**
  - Not publicly verifiable
  - Not scalable: $O(n^3)$ communication / computation complexity

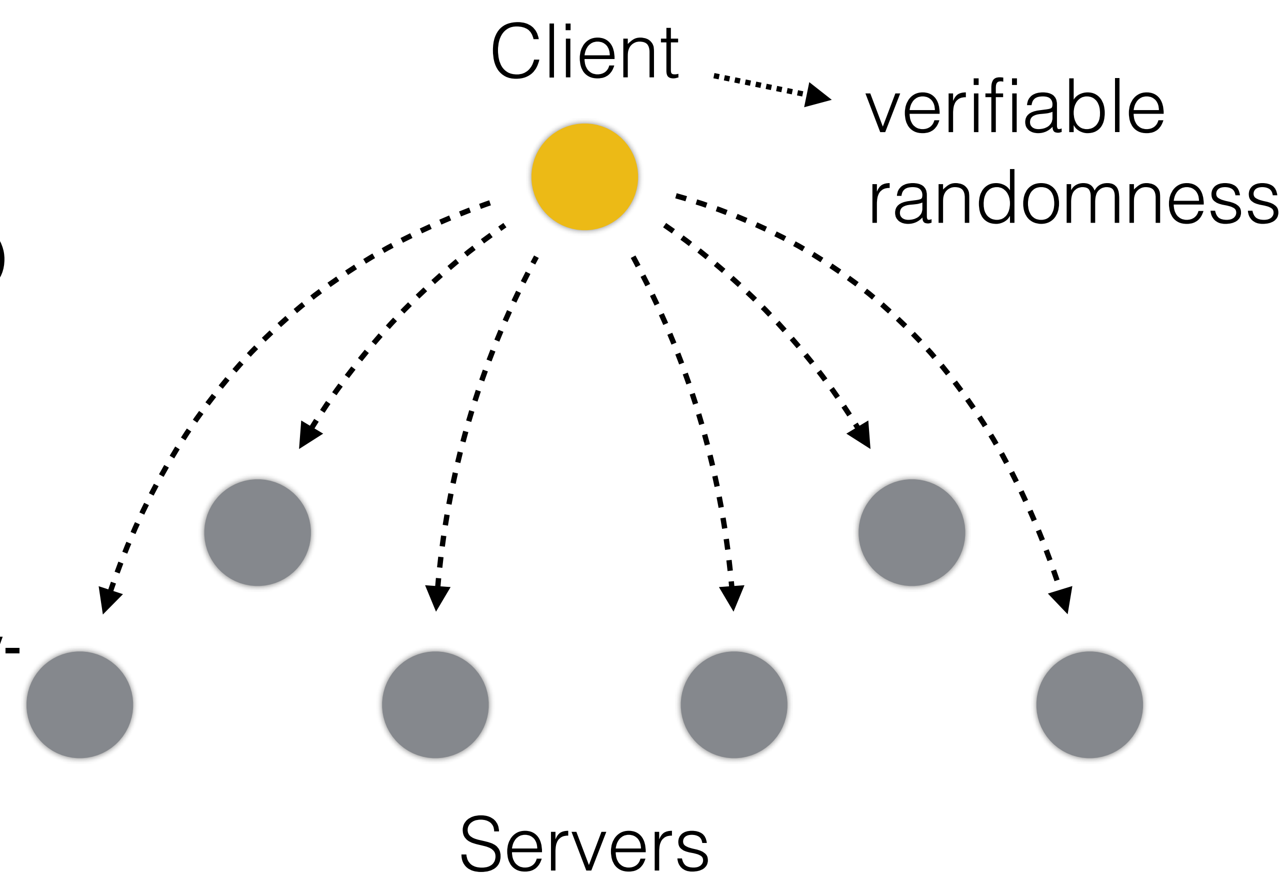
Chapter Outline

- Motivation
  - The need for public randomness
  - Strawman examples: Towards unbiasable randomness
- **RandHound**
- Implementation and Experimental Results

*Scalable Bias-Resistant Distributed Randomness, Oakland ’17

RandHound

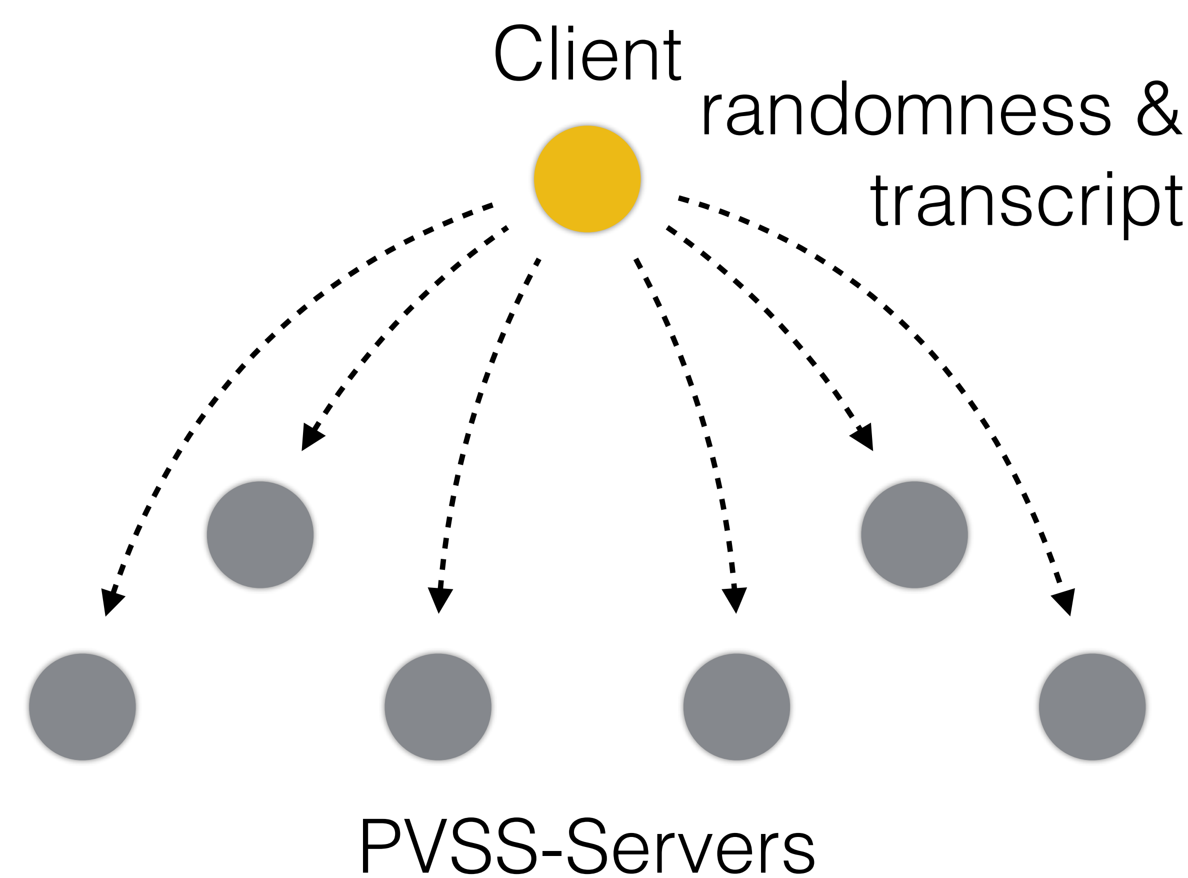
- Goals
  - Verifiability: By third parties
  - Scalability: Performance better than $O(n^3)$
- Client/server randomness scavenging protocol
  - Untrusted client uses a large set of nearly-stateless servers
  - On demand (via configuration file)
  - One-shot approach
  - Example: lottery authority



RandHound

Achieving Public Verifiability

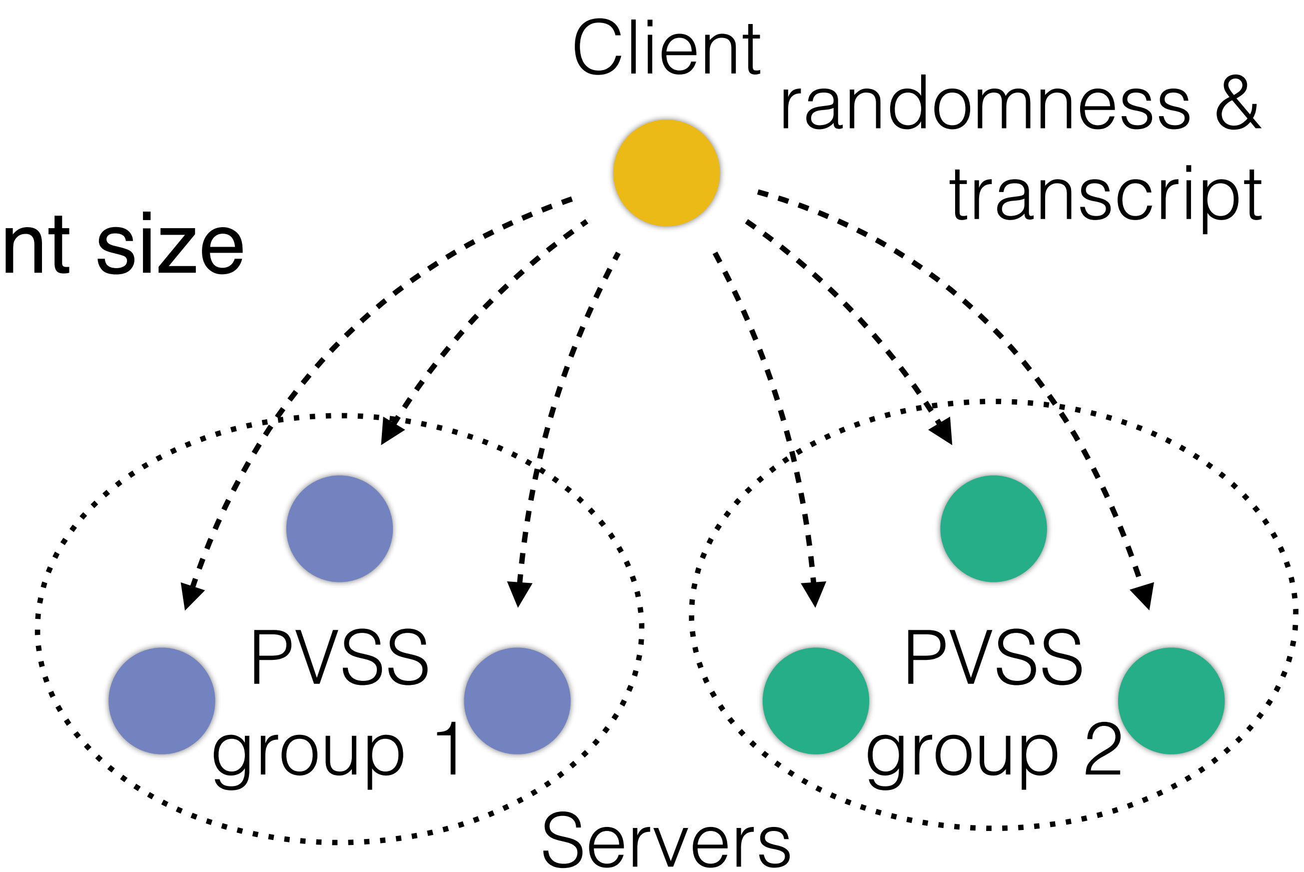
- Publicly-VSS (Schoenmakers, 1999)
  - Shares are encrypted and publicly verifiable through zero-knowledge proofs
  - No communication between servers
- Collective signing (Syta, 2016)
  - Client publicly commits to their choices
- Create protocol transcript from all sent/received (signed) messages



RandHound

Achieving Scalability

- Shard participants into constant size groups
  - Secret sharing with everyone too expensive!
  - Run secret sharing (only) inside groups
  - Collective randomness: combination of all group outputs



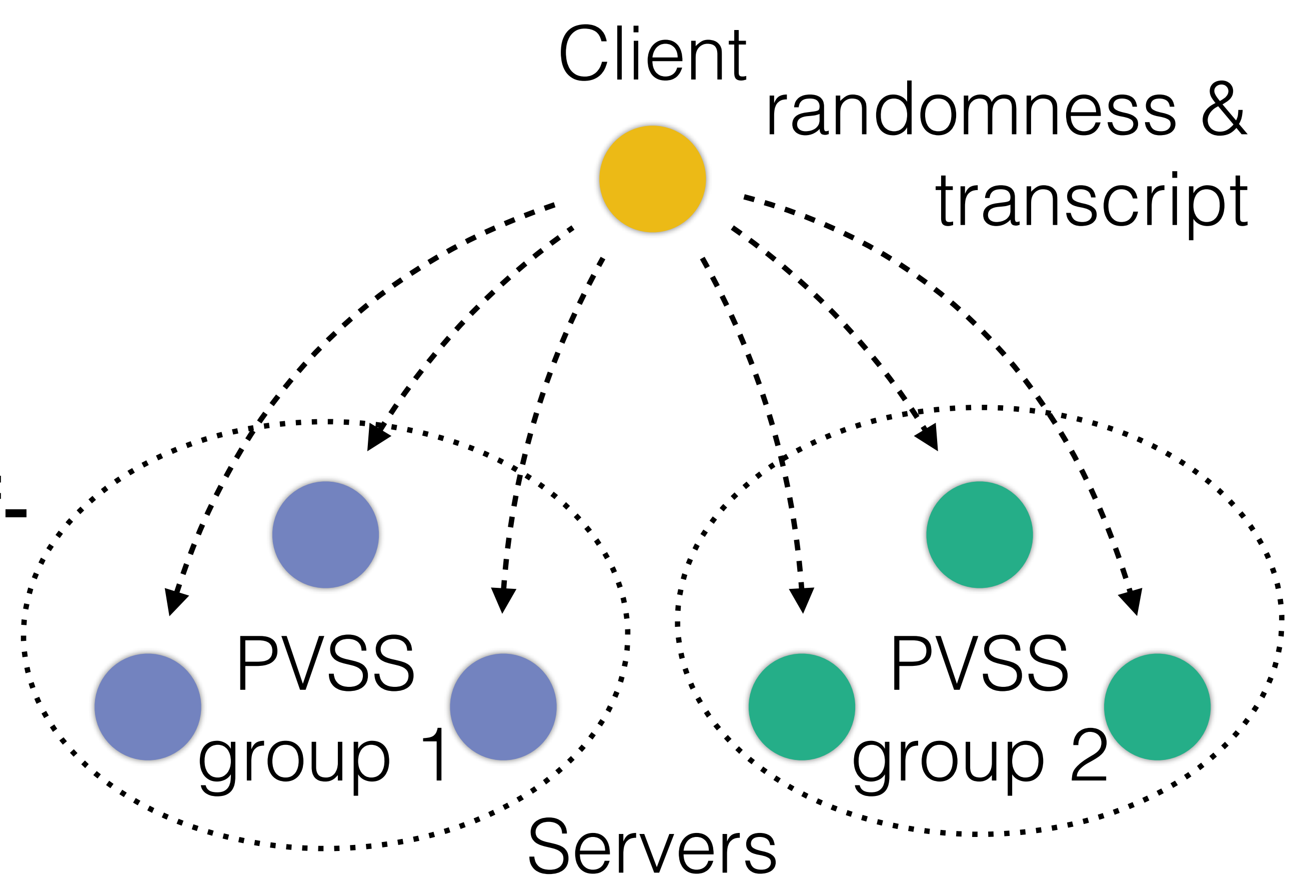
Chicken-and-Egg problem?

- How to securely assign participants to groups?

RandHound

Solving the Chicken-and-Egg Problem

- Client selects server grouping
- Availability might be affected (self-DoS)
- Security properties through
  - *Pigeonhole principle:* at least one group is not controlled by the adversary
  - *Collective signing:* prevents client equivocation by fixing the secrets that contribute to randomness



Public Randomness is (not so) Hard

	Availability	Unpredictability	Unbiasability	Verifiability	Scalability
Strawman I	−	−	−	−	−
Strawman II	−	✓	−	−	−
Strawman III	✓	✓	✓	−	−
RandShare	✓	✓	✓	−	−
RandHound	✓	✓	✓	✓	✓

Communication / computation complexity: $O(cn^2)$

Chapter Outline

- Motivation
  - The need for public randomness
  - Strawman examples: Towards unbiasable randomness
- RandHound
- **Implementation and Experimental Results**

*Scalable Bias-Resistant Distributed Randomness, Oakland ’17

Implementation & Experiments

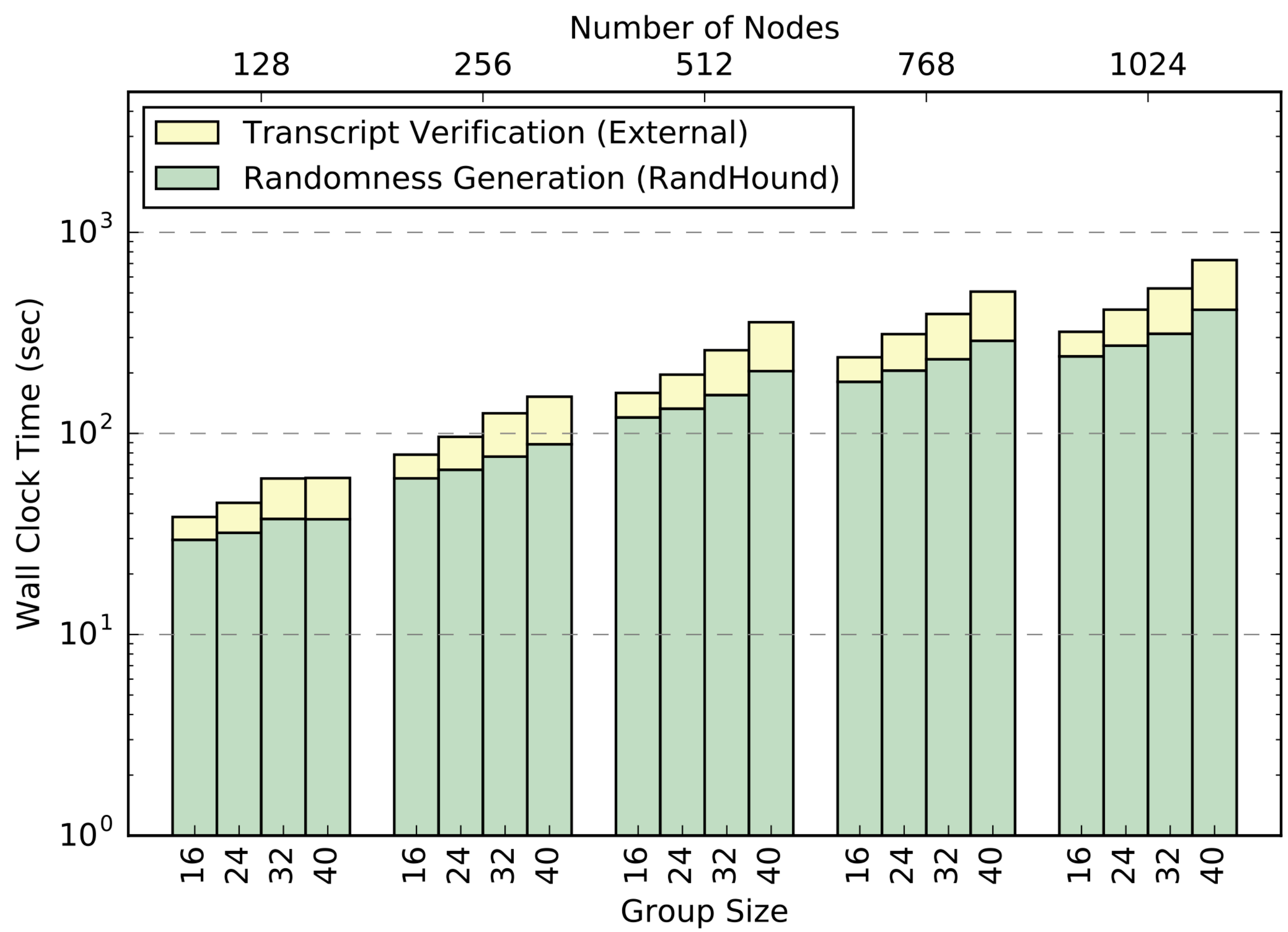
Implementation

- Go versions of DLEQ-proofs, PVSS, RandHound
- Based on DEDIS code
  - Crypto library
  - Network library
  - Cothority framework
- https://github.com/dedis

DeterLab Setup

- 32 physical machines
  - Intel Xeon E5-2650 v4 (24 cores @ 2.2 GHz)
  - 64 GB RAM
  - 10 Gbps network link
- Network restrictions
  - 100 Mbps bandwidth
  - 200 ms round-trip latency

Experimental Results



Take-away: Gen. / ver. time for 1 RandHound run is 290 sec / 160 sec with 1024 nodes, group size 32.

Talk Outline

- Part I : Introduction
- Part II : Tools for Efficient Decentralization
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC
  - Scalable Bias-Resistant Distributed Randomness
- **Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding**
- Part IV : Conclusion and Future Work

Chapter Outline

- **Motivation**
- OmniLedger
- Evaluation

*Omniledger: A secure, scale-out, decentralized ledger via sharding, Oakland ‘18

Bitcoin vs OmniLedger

	Bitcoin	OmniLedger*
Throughput	~4 TPS	~20.000 TPS
1-st Confirmation	~10 minutes	~1 second
Full Security	~60 minutes	~42 second
More Available Resources	No performance Gain	Linear Increase in Throughput

** Configuration with 1120 validators against a 12.5% adversary*

Bitcoin vs OmniLedger

	Bitcoin	OmniLedger*
Throughput	~4 TPS	~20.000 TPS
1-st Confirmation	~10 minutes	~1 second
Full Security	~60 minutes	~42 second
More Available Resources	**No performance Gain**	**Linear Increase in Throughput**

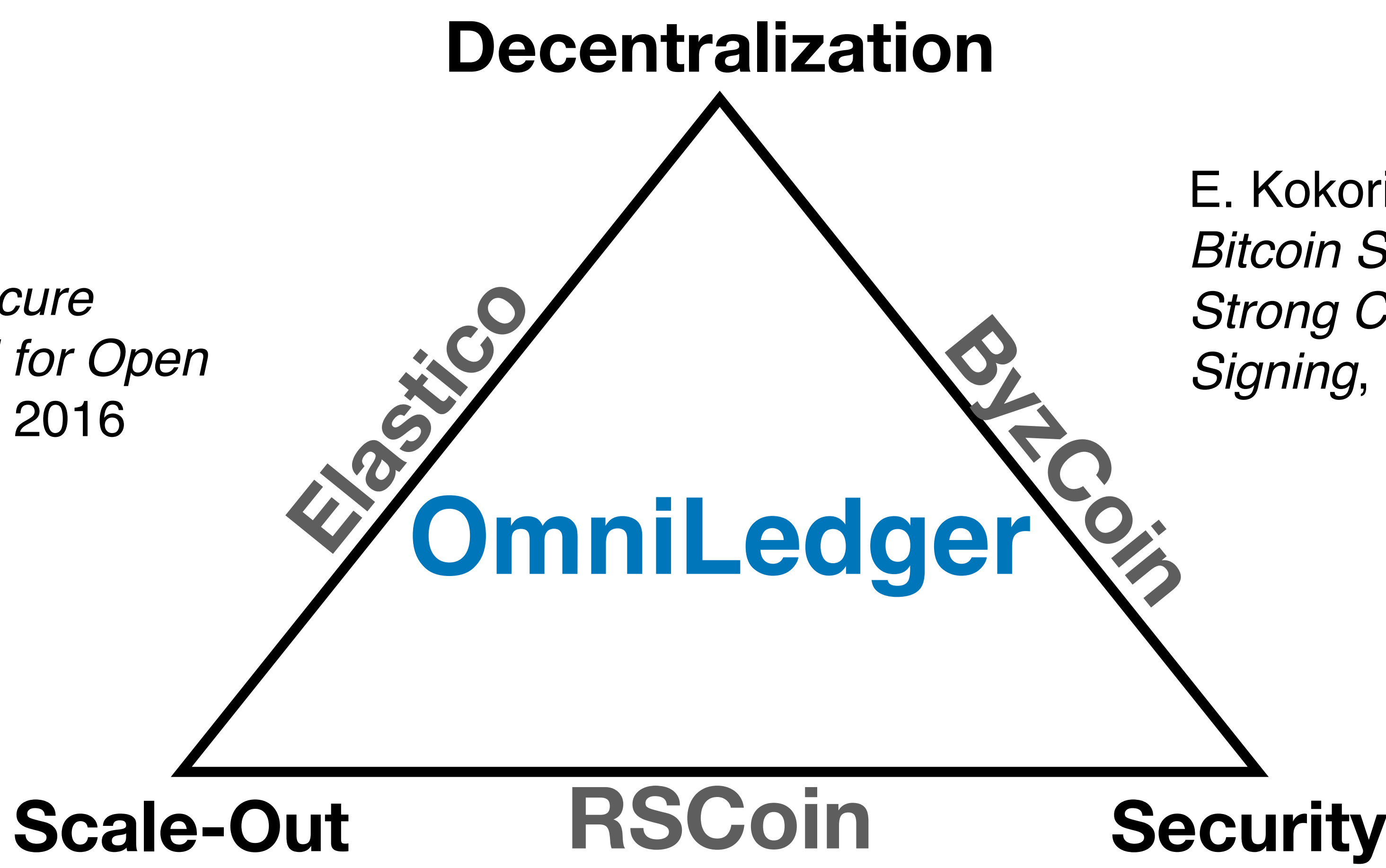
** Configuration with 1120 validators against a 12.5% adversary*

Scale-Out

… But Scaling Blockchains is Not Easy

Distributed Ledger Landscape

L. Luu et al., *A Secure Sharding Protocol for Open Blockchains*, CCS 2016



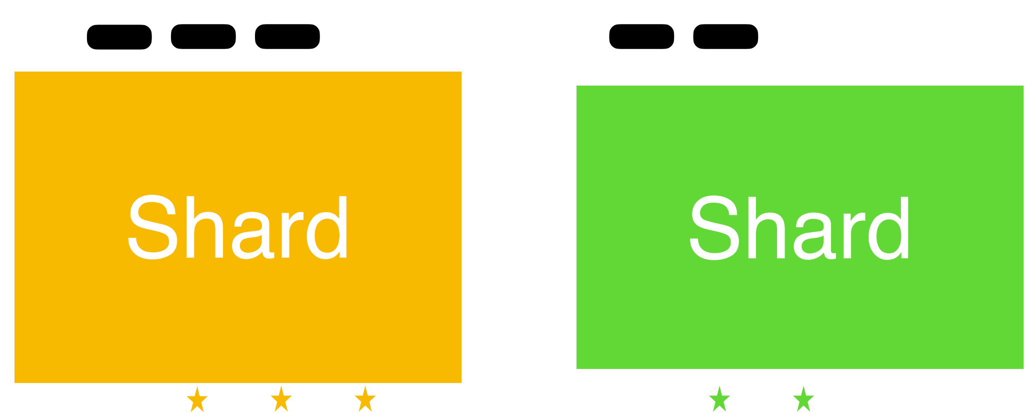
E. Kokoris Kogias et al., *Enhancing Bitcoin Security and Performance wit Strong Consistency via Collective Signing*, USENIX Security 2016

G. Danezis and S. Meiklejohn, *Centrally Banked Cryptocurrencies*, NDSS 2016

No Scale-Out (Bitcoin)

Scale-Out (OmniLedger)

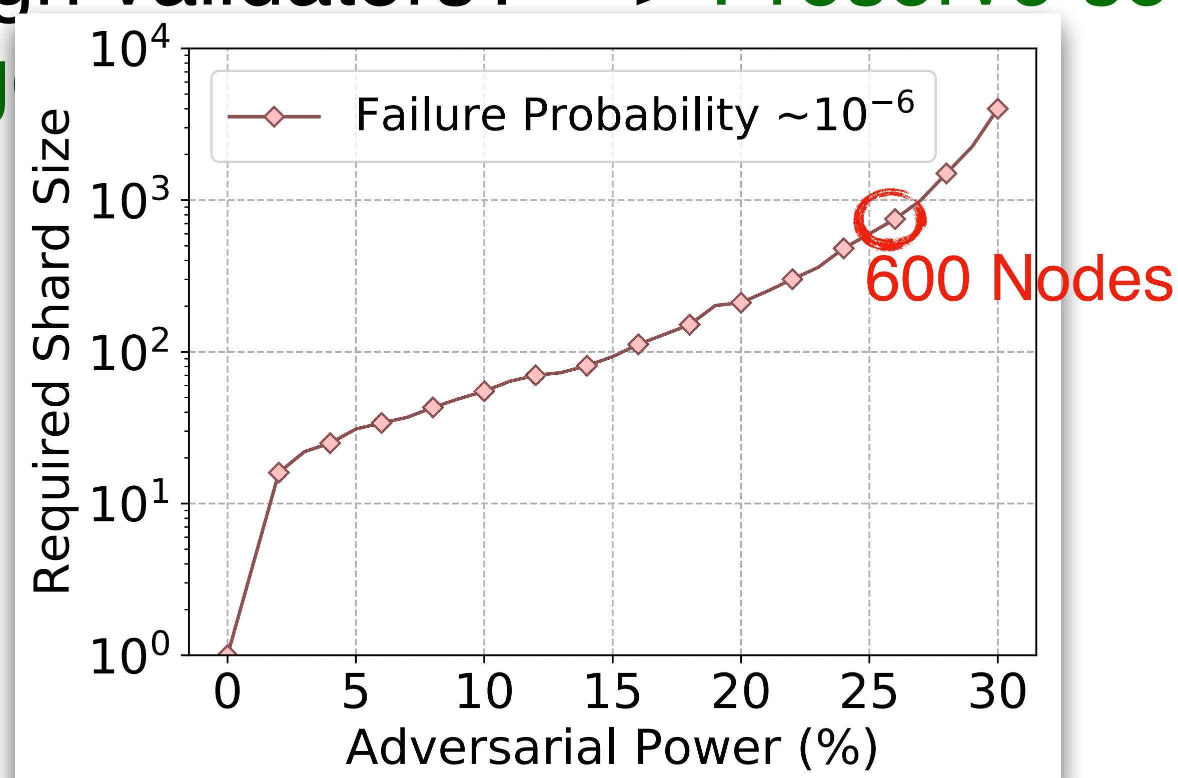
- How do validators choose which blockchain to work on?
- How can I pay a yellow vendor with greencoins?



Double Throughput

Random Validator Assignment

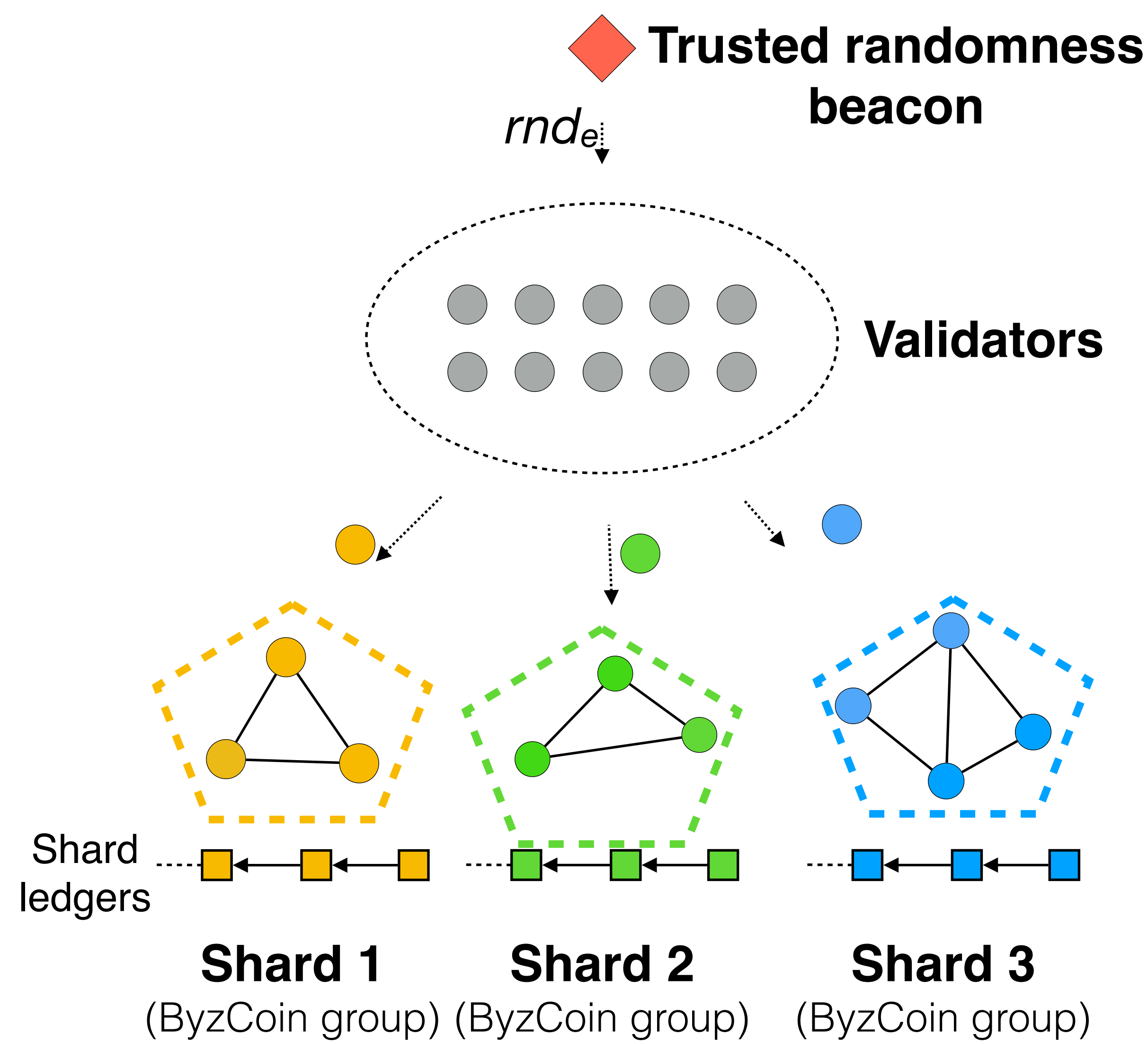
- Let validators choose? —> All malicious validators can choose the same chain
- Randomly assign validators? —> Preserve security for adequately larg



Strawman: SimpleLedger

Overview

- Evolves in epochs *e*
- Trusted randomness beacon emits random value rnd_e
- Validators:
  - Use rnd_e to compute shard assignment (ensures shard security)
  - Process tx using consensus within one shard (ByzCoin)



Strawman: SimpleLedger

Security Drawbacks

- Randomness beacon: trusted third party
- No tx processing during validator re-assignment
- No cross-shard tx support

Performance Drawbacks

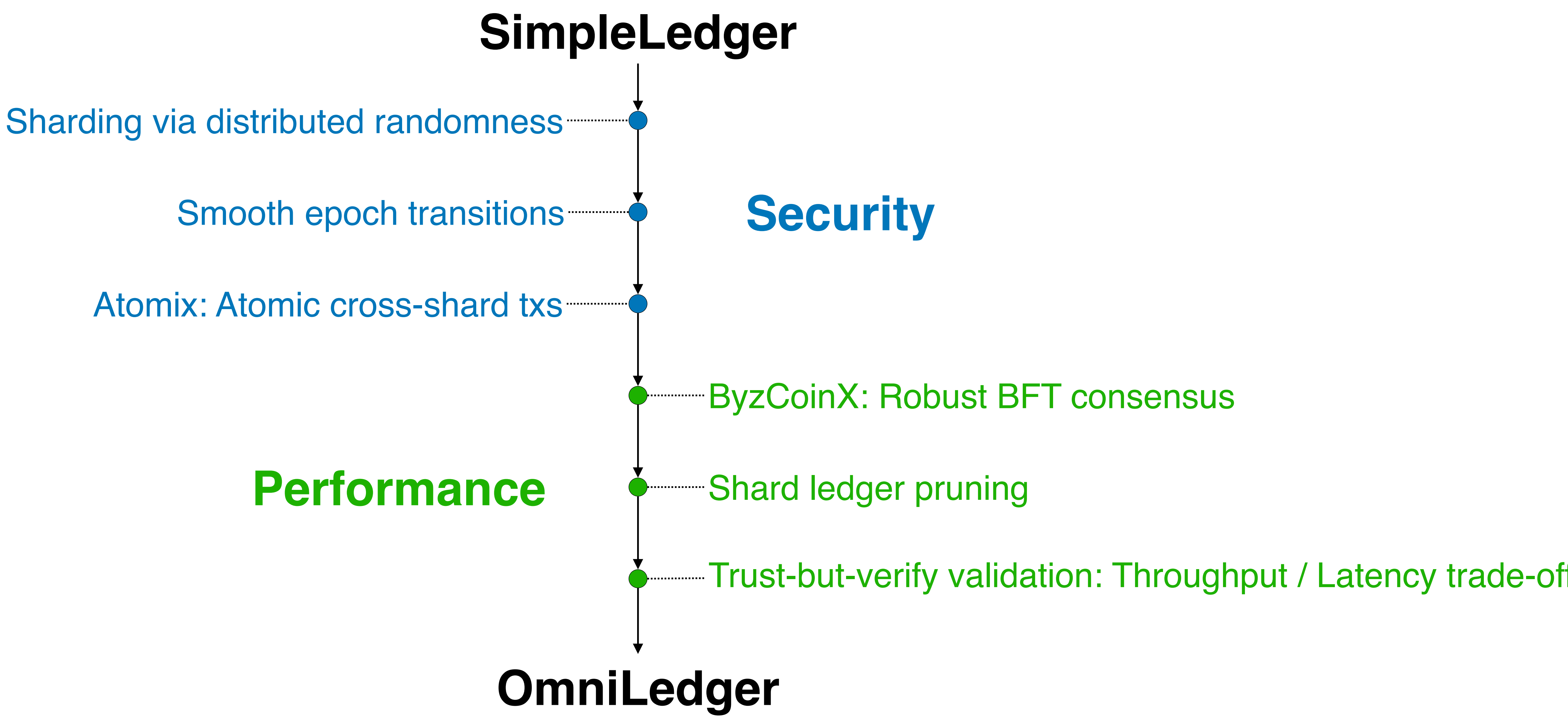
- ByzCoin failure mode
- High storage and bootstrapping cost
- Throughput vs. latency trade-off

Chapter Outline

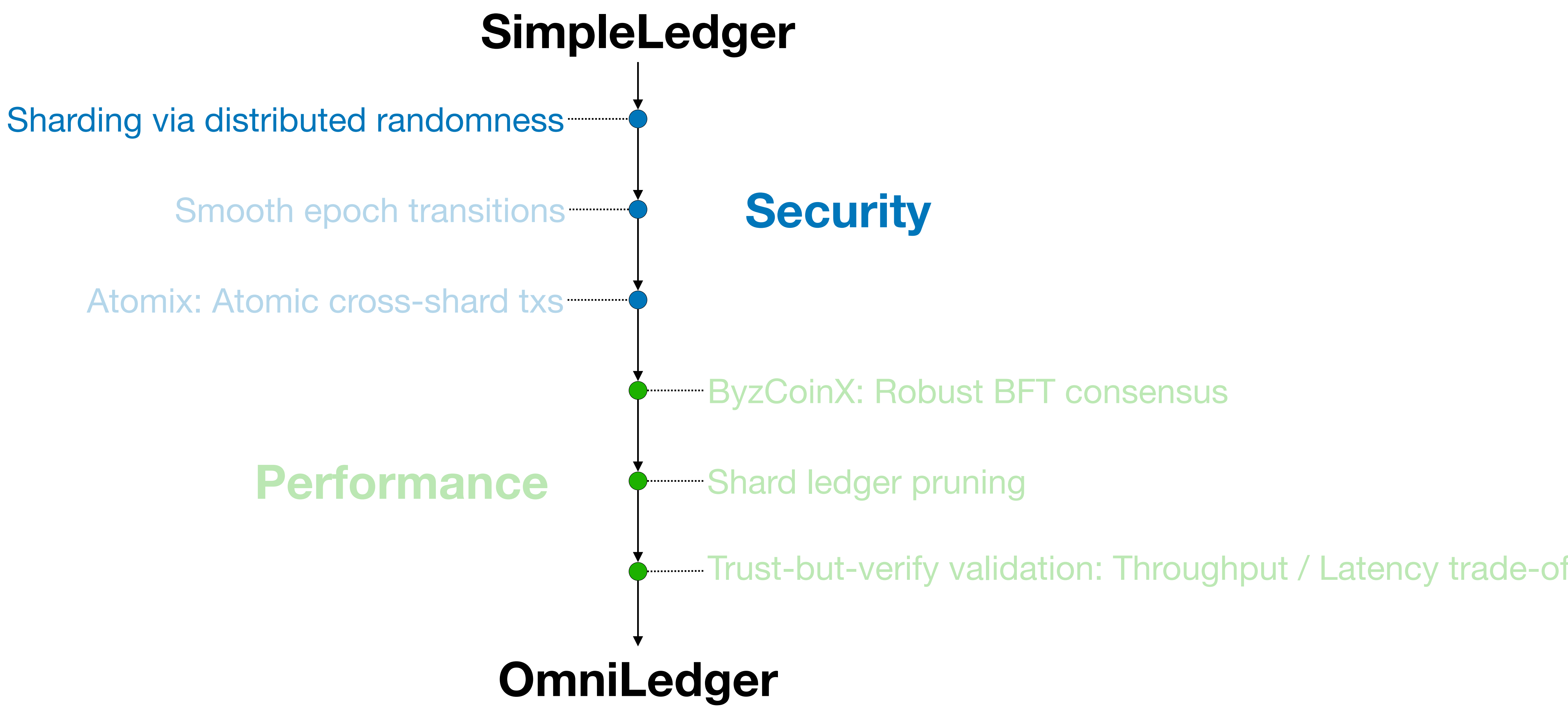
- Motivation
- **OmniLedger**
- Evaluation

*Omniledger: A secure, scale-out, decentralized ledger via sharding, Oakland '18

Roadmap


SimpleLedger
Sharding via distributed randomness
Smooth epoch transitions
Security
Atomix: Atomic cross-shard txs
ByzCoinX: Robust BFT consensus
Performance
Shard ledger pruning
Trust-but-verify validation: Throughput / Latency trade-off
OmniLedger

Roadmap

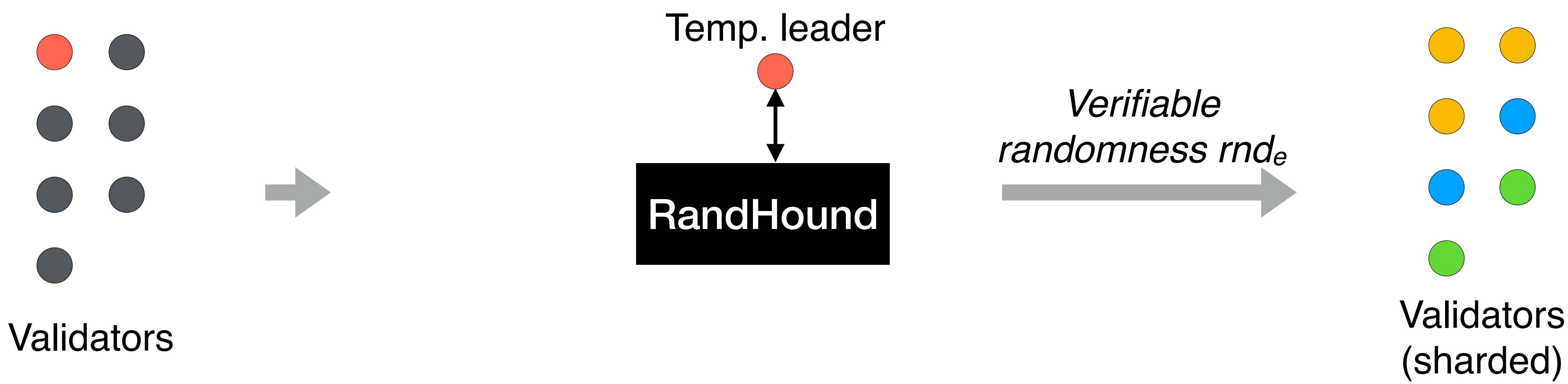

SimpleLedger
Sharding via distributed randomness
Smooth epoch transitions
Atomix: Atomic cross-shard txs
Security
ByzCoinX: Robust BFT consensus
Performance
Shard ledger pruning
Trust-but-verify validation: Throughput / Latency trade-off
OmniLedger

Shard Validator Assignment

1. Temp. leader election (Can be biased)

2. Randomness generation (Output is unbiasable)

3. Shard assignment (using rnd_e)



Roadmap

SimpleLedger

Sharding via distributed randomness

Smooth epoch transitions

Security

Atomix: Atomic cross-shard txs

ByzCoinX: Robust BFT consensus

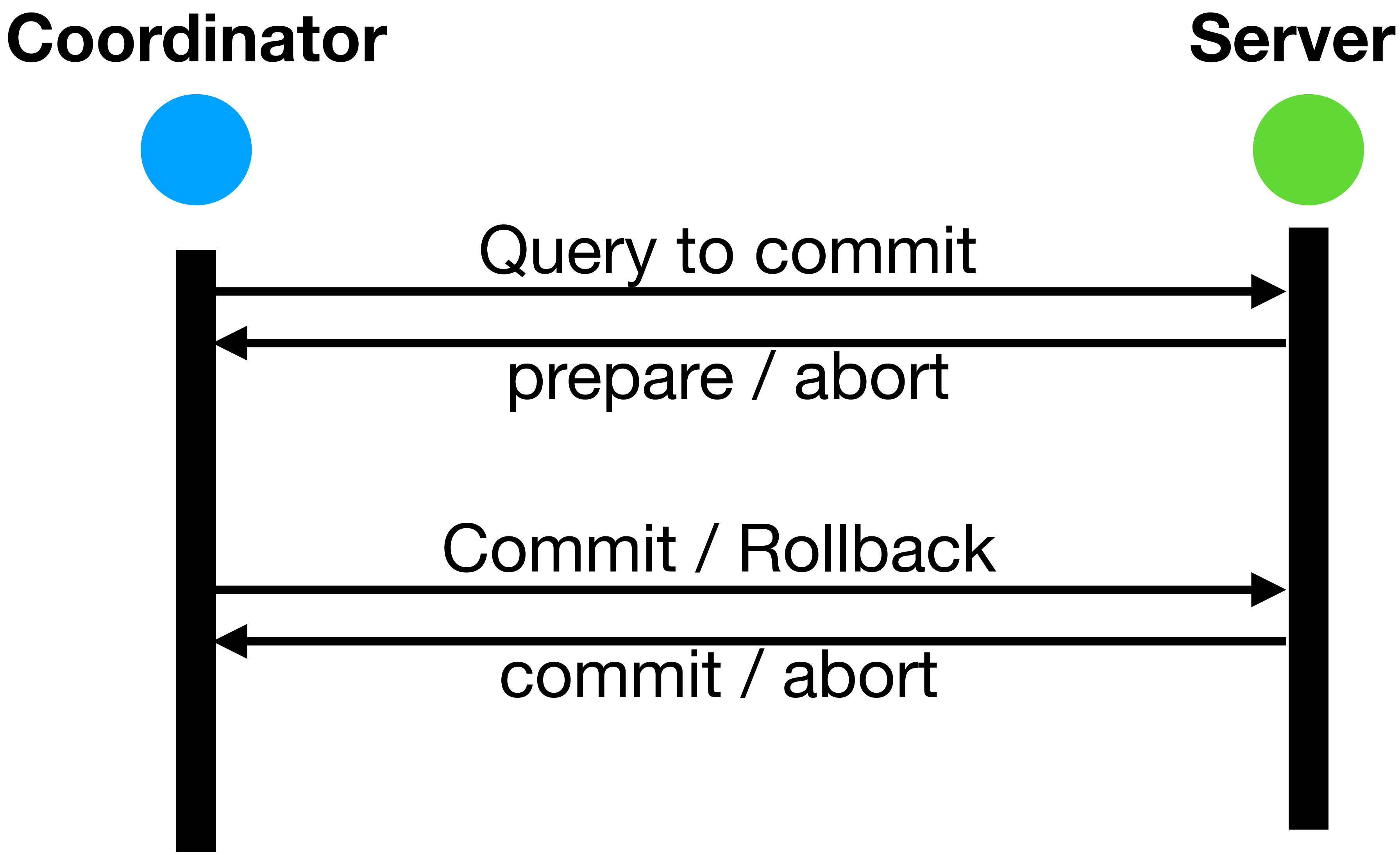
Performance

Shard ledger pruning

Trust-but-verify validation: Throughput / Latency trade-off

OmniLedger

Two-Phase Commit



Atomix: Cross-Shard Transactions

Challenge:

- Cross-shard tx commit atomically or abort eventually

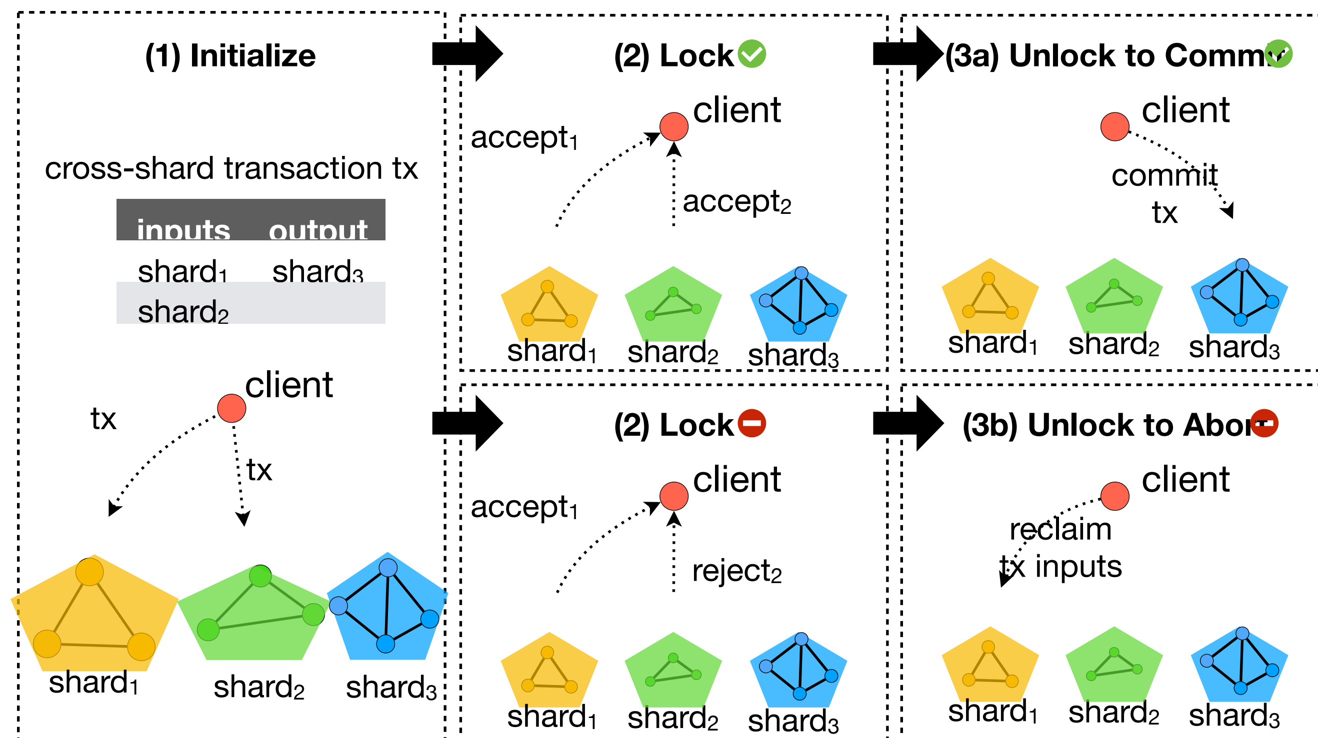
Solution: Atomix

- Client-managed protocol

    1. Client sends cross-shard tx to input shards

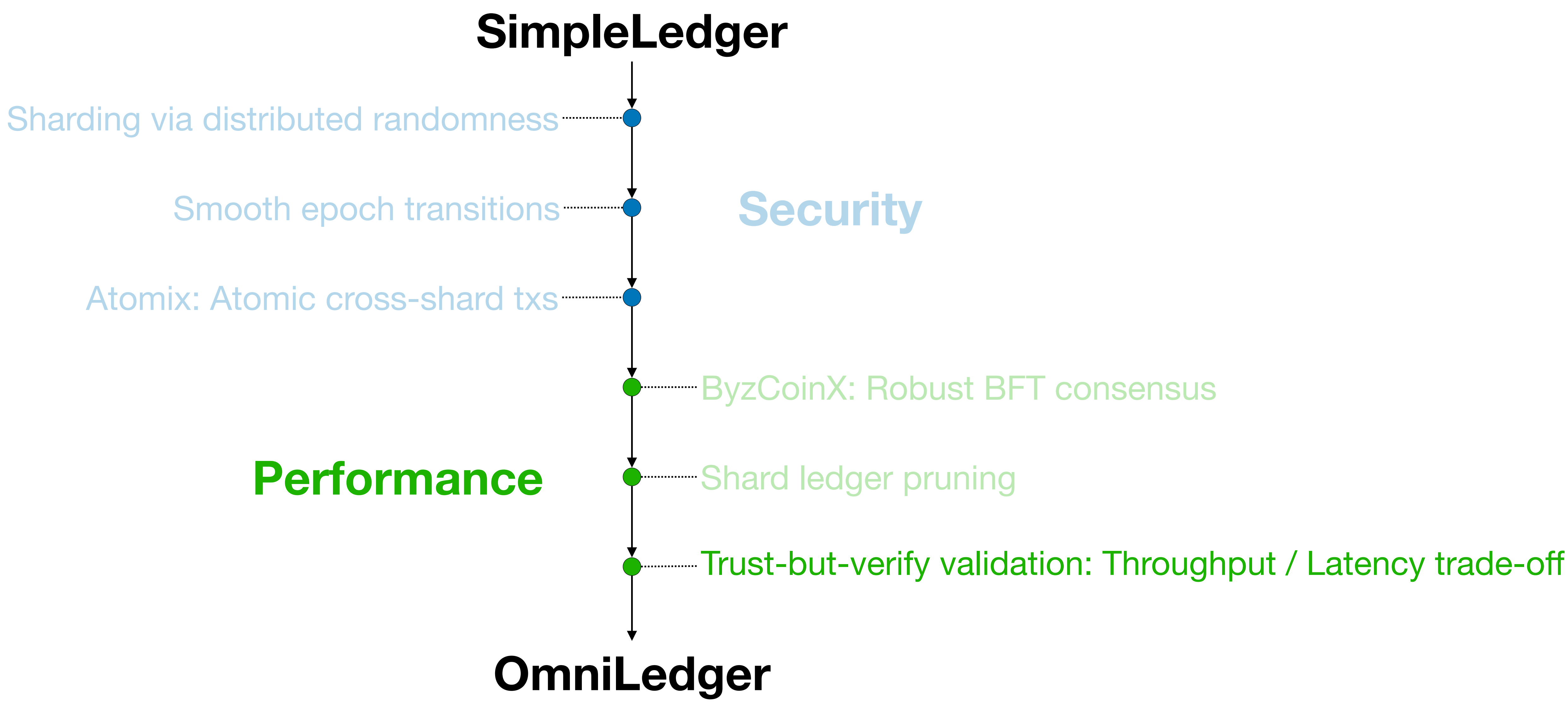
    2. Collect ACK/ERR proofs from input shards

  (a) If all input shards accept, commit to output shard, otherwise
  (b) abort and reclaim input funds



The Atomix protocol for secure cross-shard transactions

Roadmap


SimpleLedger
Sharding via distributed randomness
Smooth epoch transitions
Security
Atomix: Atomic cross-shard txs
ByzCoinX: Robust BFT consensus
Performance
Shard ledger pruning
Trust-but-verify validation: Throughput / Latency trade-off
OmniLedger

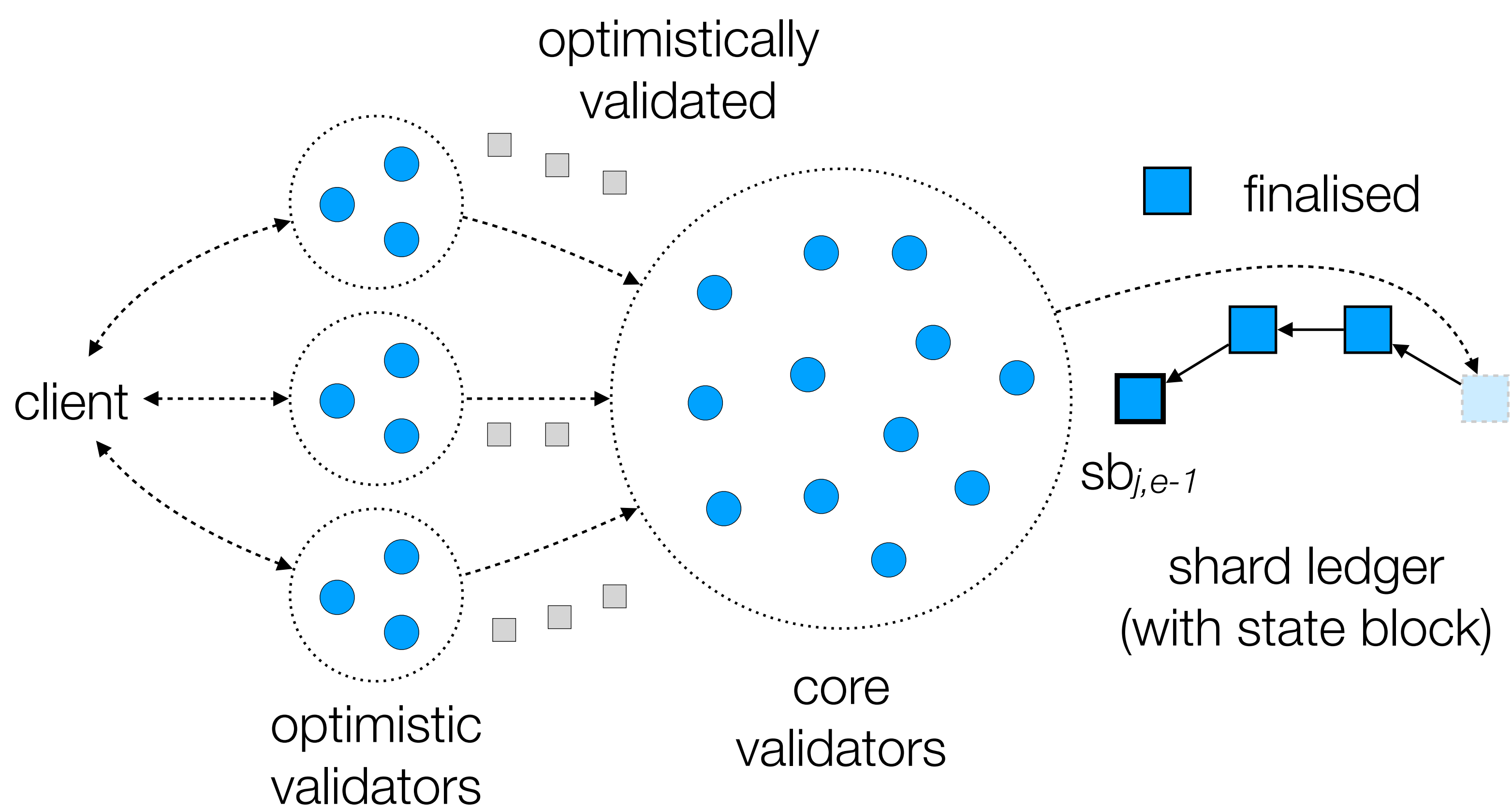
Trust-but-Verify Transaction Validation

Challenge:

- Latency vs. throughput trade-off

Solution:

- Two-level “trust-but-verify” validation
- **Low latency:**
  - Optimistically validate transactions by “insecure” shards
- **High throughput:**
  - Batch optimistically validated blocks and audit by “secure” shards



Chapter Outline

- Motivation
- OmniLedger
- **Evaluation**

*Omniledger: A secure, scale-out, decentralized ledger via sharding, Oakland ‘18

Implementation & Experimental Setup

Implementation

- OmniLedger and its subprotocols (ByzCoinX, Atomix, etc.) implemented in Go
- Based on DEDIS code
  - Kyber crypto library
  - Onet network library
  - Cothority framework
- https://github.com/dedis

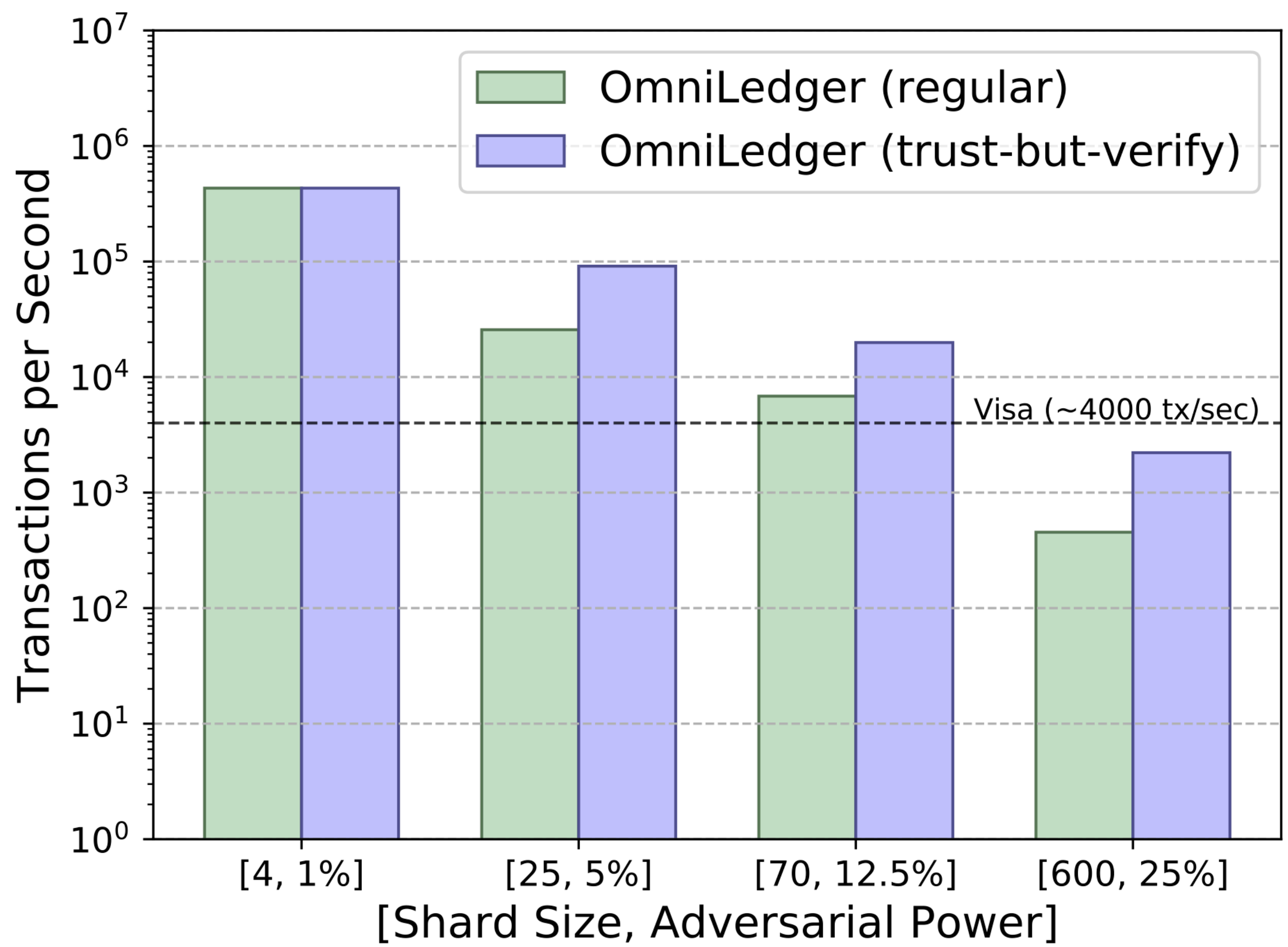
- **DeterLab Setup**
- **48 physical machines up to 1800 clients**
  - Intel Xeon E5-2420 v2 (6 cores @ 2.2 GHz)
  - 24 GB RAM
  - 10 Gbps network link
- **Network restrictions (per client)**
  - 20 Mbps bandwidth
  - 200 ms round-trip latency

Evaluation: **Scale-Out**

#validators	**70 (1)**	**140**	**280**	**560 (8)**	**1120**
OmniLedger (tx/sec)	439	869	1674	3240	5850
Bitcoin (tx/sec)	~4	~4	~4	~4	~4

Scale-out throughput for 12.5%-adversary and **shard size 70** and 1200 validators

Evaluation: **Throughput**



Results for 1800 validators

Evaluation: **Latency**

Transaction confirmation latency in seconds for regular and mutli-level validation

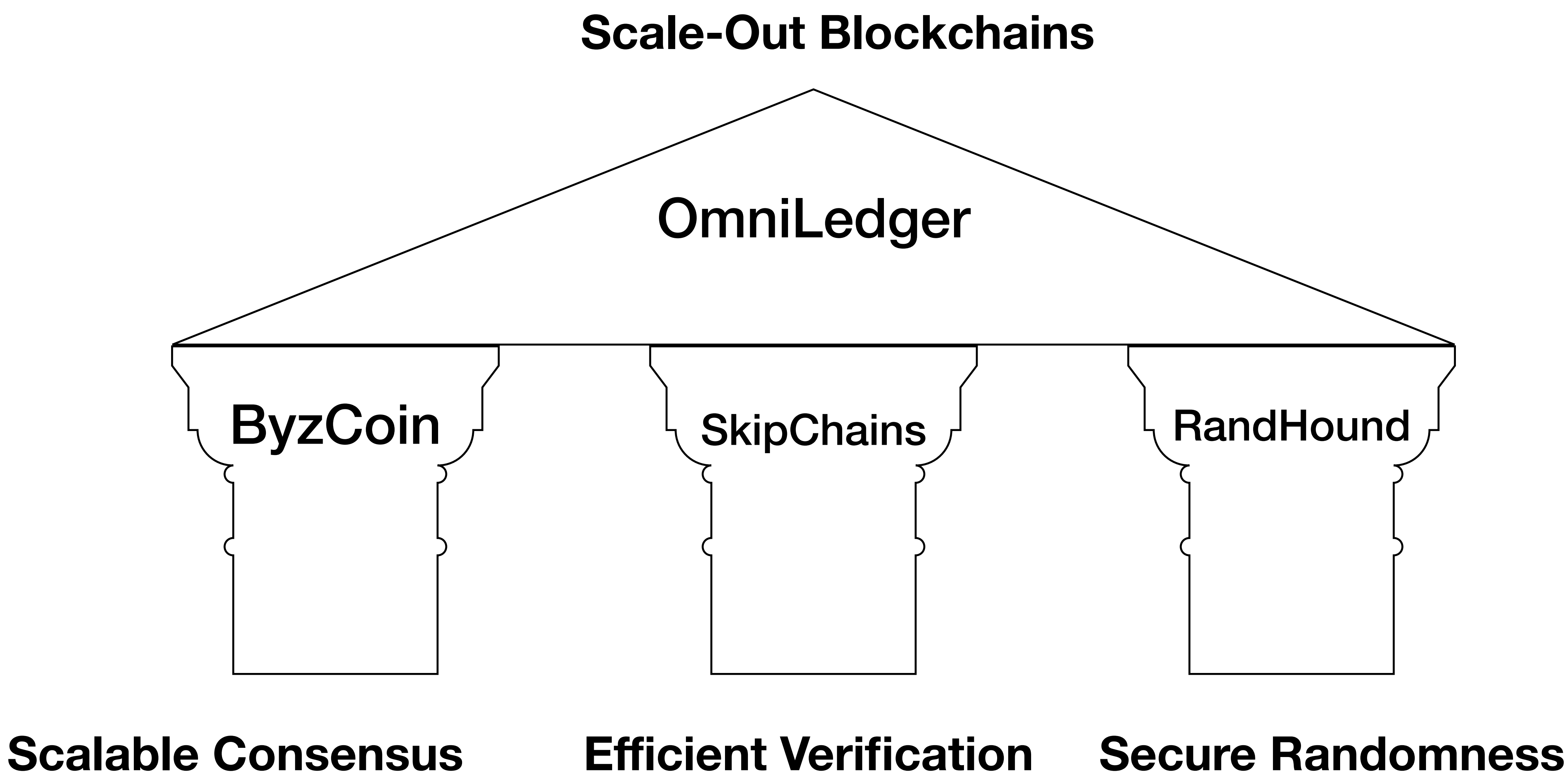
#shards, adversary	4, 1%	25, 5%	70, 12.5%	600, 25%	
regular validation	1,38	5,99	8,04	14,52	1 MB blocks
1st lvl. validation	1,38	1,38	1,38	4,48	500 KB blocks
2nd lvl. validation	1,38	55,89	41,89	62,96	16 MB blocks
Bitcoin	600	600	600	600	

latency increase since optimistically validated blocks are batched into larger blocks for final validation to get better throughput

Talk Outline

- Part I : Introduction
- Part II : Tools for Efficient Decentralization
  - Scalable, Strongly-Consistent Consensus for Bitcoin
  - Decentralized Timeline-Tracking and Long-Term Relationships using SKIPCHAINIAC
  - Scalable Bias-Resistant Distributed Randomness
- Part III : OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding
- **Part IV : Conclusion and Future Work**

Conclusion



Scale-Out Blockchains
OmniLedger
ByzCoin
SkipChains
RandHound
Scalable Consensus
Efficient Verification
Secure Randomness

Future Work

- **Scaling and Performance**: Scaling up blockchains to handle intensive global workloads for both permissionless decentralized blockchains, and permissioned/consortium blockchains supporting >100,000 transactions/sec.
- **Correctness by Design and Construction**: Making it easy, and even automatic, for blockchain developers to produce secure protocols and code, by utilizing (1) programming language techniques to create correct code, and (2) cryptographic protocols with security proofs. *
- **Confidentiality**: Combining transparency with confidentiality in blockchains, by utilizing (1) cryptographic techniques, as well as (2) trusted-hardware.
- **Authenticated Data Feeds**: Supporting a robust ecosystem of trustworthy data feeds for blockchains and contributing high-trust data feed solutions.
- **Safety and Compliance**: Enabling techniques and protocols for effective monitoring and targeted intervention in blockchains, informed by evaluations of traditional contract law and risks of crime in smart contracts.
- **Sound Migration**: Formulating practical migration paths to production blockchain deployments and enabling integration of new blockchain systems with legacy systems.

*Protean: A modular architecture for general-purpose decentralized computing. HotOS XVII