Harmony

Open consensus for 10B People

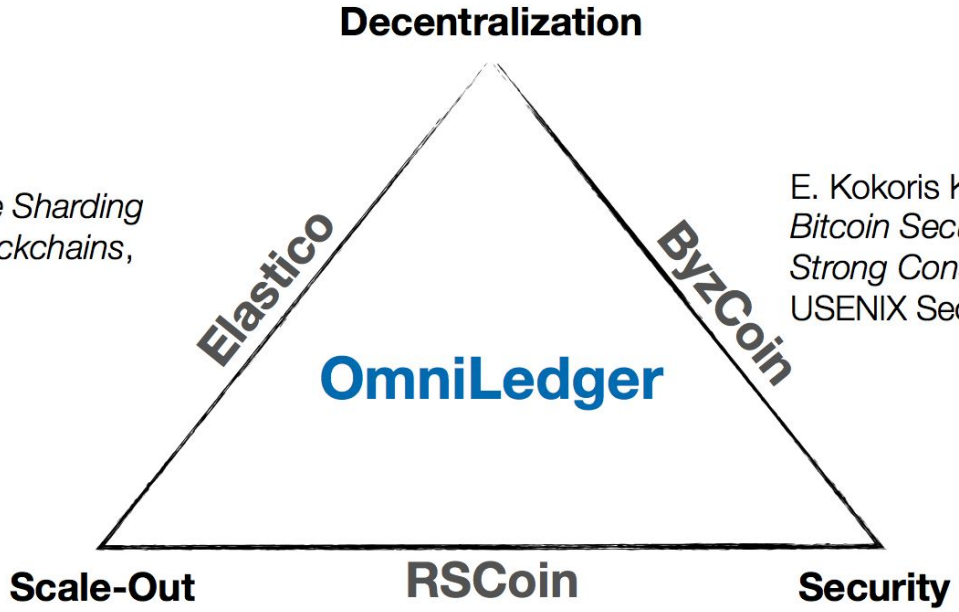
Scaling Smart Contracts via Systems and Language Design

Stephen Tse

harmony.one/law

Bring Research Results to Production!

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

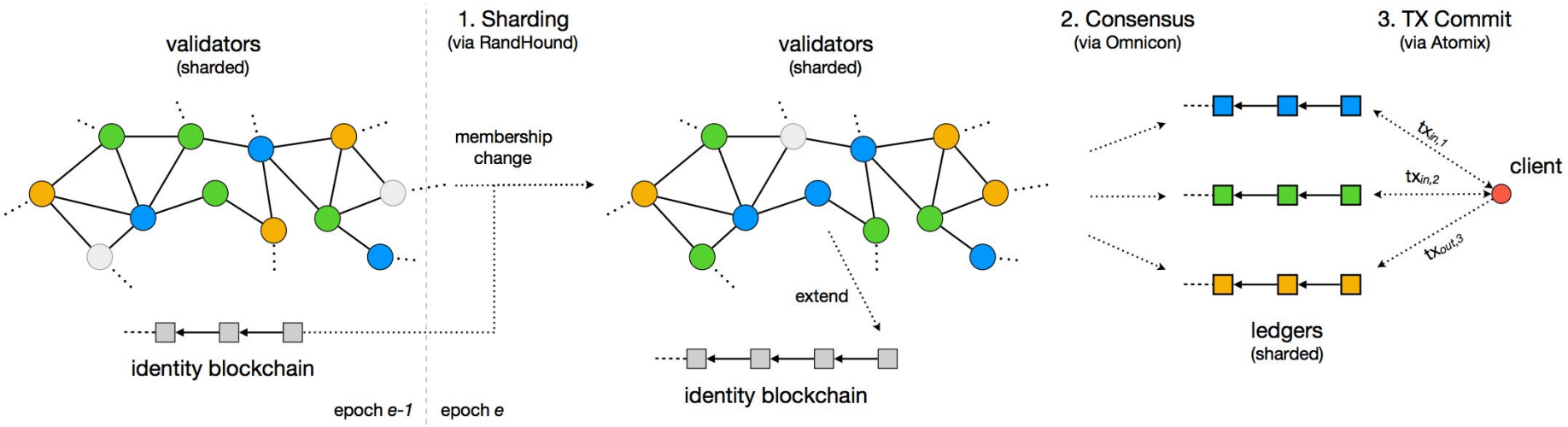


E. Kokoris Kogias et al., *Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing*, USENIX Security 2016

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

Backtesting with historical data?
Adversarial simulation, parameter optimizations?

Harmony: Production Network of 100k Nodes



A high-performance blockchain demands ***10x innovations*** in transport networks, consensus protocols & systems tools.

We bring proven innovations to *large-scale* production [OmniLedger*, RapidChain, Blockmania, Avalanche].

Million Users, Billion Actions

Fair Games

Own your game assets; all *game mechanics* are open in public for everyone to check **against cheating**

Private Lives

Every place you visit, things you eat to be recorded with privacy; aggregated models gives *personalized recommendation*

Equal Matches

Control your match making in dating & job search; disclose your *photos, bio or salary* only to the chosen ones

Harmony facilitates building *real* applications for millions.
To bring **fairness, privacy and equity** to all.

AI Data Marketplaces: Wellness News & Job Recruit

N-to-N Transactions

Bitcoin 1-to-1 for transfer, Ethereum 1-to-N for tokenize; versus *millions of consumer data* to thousands of models

Store & Compute

Storage providers with analytics and *structured queries;* use **influence functions** in training for data appraisal

Access Revocation

Right to be Forgotten #GDPR via key burning, *auditable access control* with privacy-preserving computation

Harmony is a new ***public chain*** redesigned with top performance and integrated feature layers.

For real-world decentralized applications.

Scaling Trust to 10B people & 100B devices

Can we *agree* now but foresee **inconsistency** in practice?

Can we *rely* on **intuition** when expressing terms & conditions?

Can we *iterate* without lawyers, regional laws, or **penalties**?

Can smart contracts be safe, easy, fast?

Trusting Contracts: As **SAFE** as Java

Java Virtual Machine makes web applications so *boringly* reliable for teams of 100+ developers

Trusting Contracts: As **SAFE** as Java

Run-time checks maintain **global constraints** (termination, resource consumption, balance flow)

Trusting Contracts: As **SAFE** as Java

Formal verification via dependent types (Twelf, ProVerif, Coq) guarantees *hacker-proof* before deploy

Trusting Contracts: As **SAFE** as Java

Java Virtual Machine makes web applications so *boringly* reliable for teams of 100+ developers

Run-time checks maintain **global constraints** (termination, resource consumption, balance flow)

Formal verification via dependent types (Twelf, ProVerif, Coq) guarantees *hacker-proof* before deploy

Tse/TOPLAS: Verified interoperable implementations of security protocols
Tse/IEEE-SP: Run-time principals in information-flow type systems

Writing Contracts: As **EASY** as Python

Minimal syntax (no distraction!), human has limited cognitive focus for abstractions such as *invariants* & *isomorphism*

Writing Contracts: As **EASY** as Python

Functional and *process calculi* to avoid managing states, type inference as theorem proving to tame structures & complexity

Writing Contracts: As **EASY** as Python

Understand theories behind dependencies (information flow) & **parametricity** (higher-order polymorphism) vs “fat languages”

Writing Contracts: As **EASY** as Python

Minimal syntax (no distraction!), human has limited cognitive focus for abstractions such as *invariants* & *isomorphism*

Functional and *process calculi* to avoid managing states, type inference as theorem proving to tame structures & complexity

Understand theories behind dependencies (information flow) & **parametricity** (higher-order polymorphism) vs “fat languages”

Tse/Penn: Concise concrete syntax (generalized LR parser), min-lang.com
Tse/ICFP: Translating dependency into parametricity

Running Contracts: As **FAST** as OCaml

Compile to native code & run tests in 200 ms

Tezos, Zilliqa's Scilla, Coda's SNARK, **Coq** are written in OCaml

Declarative style leaves freedom for memory management, parallelism strategies, graph executions, sharding algorithms

Beautiful and Secure Code

```
black_scholes
    s : ℝ # stock price
    x : ℝ # strike price
    t : ℝ # expiration time in years
    r : ℝ # risk-free interest rate
    σ : ℝ # volatility
    : ℝ
= s ϕ(d1) - x e^(-r t)ϕ(d2) @
 ϕ = Normal.cdf
 d0 = log s/x + (r + σ²/2)t
 d1 = d0 / σ√t
 d2 = d1 - σ√t
```

harmony.one/type-checks

Join **Harmony** team! Bring Empathy, Passion, Excellence

Rust engineers, protocol researchers, compiler writers to **s@harmony.one**

See harmony.one/law, /sharding and /tgi

Stephen: security protocols PhD

Nicolas: VR startup founder

Alok: **Apple Siri** ML

Rongjian: Google search

Minh: **Google infrastructure**

Nick: Stanford AI masters

Sahil: Harvard MBA

Eugene: Amazon networking

Leo: **Amazon Phone OS**

Hakwan: Rhode Scholars

Kayuet: Oxford PhD

Roadmap for 2019 and Beyond

Communities

Developers for protocols & apps, evangelize regional groups, *exchange listing in Q1*, tours

Contracts & Apps

Sharding contract executions, WASM compatible, verified security, incentivized consumer apps

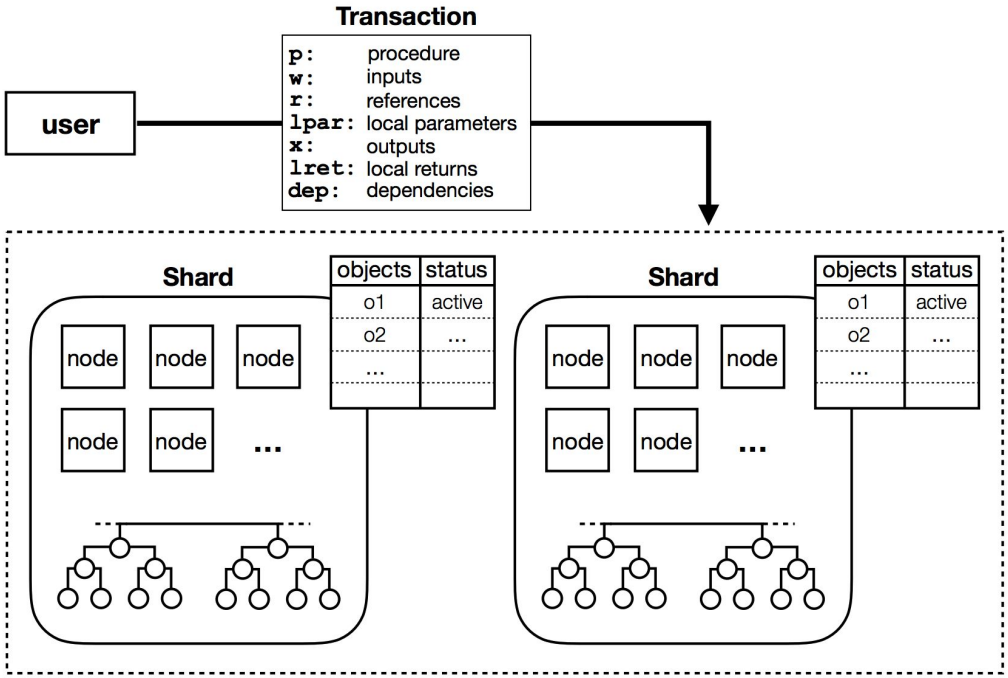
More Scaling

Optimize network parameters, *simulate attacks*, implement RapidChain, launch **mainnet in Q2**

Harmony maintains a long-term view of product building tying *people, vision and technology* together.

The value of a network is **Strength in Numbers**.

Sharding Contracts with Static Types



Age of Open Development: a *core team* to **productionize** research with protocol + application *communities*

[Chainspace*, Scilla, Fraud Proofs].

OmniLedger: Principles & Optimizations for Scaling

Representative sharding

O(1)-size multi-signatures for 10k nodes vs 16-node PBFT. Crypto sortition via randoness from multi-party computation and commit-then-reveal step.

Gradual transition

Sybil-resistant identities to maintain liveness when swapping. A sliding window from a fixed permutation to ensure ⅔ honest majority.

Atomic shard-commit

Each shard uses O(log n) *multicast tree-based BFT* to unanimously accept cross-shard transactions with O(1)-size *coordination.*

Parallelizing blocks

Acyclic graphs to capture transaction *dependencies transitively*. Divide each shard into groups to replace faulty nodes with a view-change.

Pruning checkpoints

State blocks for storage and bootstrapping against Byzantine DoS. Multi-hop, collectively signed back -pointers, 100x space savings.

Optimistic confirms

Trust but verify low-value transactions with shard deposits. Guarantee finality in ~1s with *penalty linear to loss* and detection in minutes.