Accélérateur
de la transformation
numérique

STAKECUBE

Combining Sharding and Proof-of-Stake to build Fork-free Secure Permissionless Distributed Ledgers

Antoine Durand (IRT-SystemX)
Emmanuelle Anceaume (CNRS/IRISA)
Romaric Ludinard (IMT Atlantique)

Advance Blockchain | Initiative BART (2/3)

BART
Blockchain
Advanced Research
& Technologies

Technologie Blockchain (axes 1, 3 et 6) :

- Modélisation et automatisation de processus métiers
- Algorithmes de consensus pour l'IoT (Thèse #2: SystemX / TPT / IMT / Atos)
- Evolutivité et volumétrie des Blockchains

Blockchain de consortium (axe 4) :

- Modèles et règles de gouvernance
- Modèles économiques et d'incitations
- Gestion décentralisée des identités et certification

Données et sécurité (axes 2 et 5) :

- Confidentialité et anonymisation des données (transactions)
- Analyse, visualisation et valorisation des données (Thèse #4: SystemX / INRIA)
- Calcul multipartite sécurisé (Thèse #3: SystemX / INRIA)

Régulation (axe 4) :

- Blockchain vs. GDPR (règlement général sur la protection des données)
- Blockchain vs. relations contractuelles de droit privé (Thèse #1: SystemX / INRIA / UVSQ)

Outline

- **Achievements**
  - Model
  - Properties
- **A set of ingredients**
  - Tools and protocols
  - PeerCube
- **Protocol description**
  - Overview
  - Credential System
  - Shard composition update
  - Producing the next block
- **Future work**

- **Sharded Ledger**
  - Motivated by uses case where efficiency concerns are a priority
  - Per-block agreement approach in the UTXO model
- **Stake-bounded Weakly dynamic adversary**
  - Corruption threshold and synchrony hypothesis are offloaded to building blocks
  - Corruptions are moving but subject to a delay
- **Probabilistic guarantees**
  - All properties holds with probability $1 - negl(\kappa)$

Properties (informal)

- **Safety**
  - If honest users i and j accepts respectively block B_h^i and B_h^i at height h , then $B_h^i = B_h^j$
- **Liveness**
  - Every submitted transaction is eventually confirmed by all honest users

Properties (informal)

- **Scalability**
  - Overall communication cost is $O(nc_1+c_2)$
  - c_1 and c_2 are polynomial in the security parameter κ
- **Efficiency**
  - Each block takes a constant number of rounds
  - Transaction confirmation takes one block

- **Cryptographic primitives**
  - Standard cryptographic tools
  - Verifiable Random Function (VRF)
- **Vector consensus**
- **Random Beacon**
- **Verifiable Byzantine Agreement**

PeerCube[1]

Sharded Distributed Hash Table

- Require random, globally verifiable and short-lived identifiers
- Creates shards based on the XOR distance function
- Each shard is randomly splited into two sets
  - The core set runs all procedures and has a fixed size
  - The spare set only follows the core set
- Statistical security against byzantine adversaries
  - Probability of corrupting a shard negligible in the core set size
  - Actually tolerates up to $\mathcal{F}$ corrupted shards

[1] Anceaume, E., Ludinard, R., Ravoaja, A., & Brasileiro, F. Peercube: A hypercube-based p2p overlay robust against collusion and churn. SASO 2018, IEEE.

- **Each block has an associated random seed**
  - Bootstrap the global credential system
  - Credentials are given to PeerCube, which maintains shards
- **All shards update their composition in parallel**
  - Handle join requests and core set election
- **New block and randomness generated by a subset of shards**
  - Leverage low corrupted shard bound $\mathcal{F}$

- **A random credential $\boldsymbol{\sigma_i(h)}$ is given to unspent output $\boldsymbol{pk_i}$ at height $\boldsymbol{h}$**
  - The randomness is drawn from the block random seed
- **Credentials are renewed every $\boldsymbol{T}$ blocks**
  - The first credential is given T blocks after transaction inclusion

$$\sigma_i(h) = hash(pk_i \parallel seed(B_{h'}))$$

$$h' := h_0 + \left\lceil \frac{h - h_0}{T} \right\rceil T$$

- **Composition of a shard $\mathcal{S}(h) = (\mathcal{S}_c(h), \mathcal{S}_s(h))$**
- **Join requests are locally stored in a buffer $\boldsymbol{b_i}$**
- **Upon reception of block $\boldsymbol{B_h}$**
  - Expiring credentials are removed
  - Users in $\mathcal{S}_c(h-1)$ run a vector agreement with input b_i
    - Defines the set of new spares
  - Users in $\mathcal{S}_c(h-1)$ run a random beacon
    - Seeds the core election
  - $\mathcal{S}_c$(h) is sent to all shards

Block creation process

- **A subset of shards runs the verifiable byzantine agreement**
  - The subset size ensure correctness despite $\mathcal{F}$ corrupted shards
  - Benefits from (leader-based) optimistic protocols
  - The algorithm is adapted to be run by shards instead of nodes
- **Block are proposed by shards**
  - The transactions are obtained from the vector consensus
  - The randomness is derived from the vector consensus with VRF as input

Future work

- **Signatures aggregation**
- **Storage sharding**
- **Security/efficiency improvements**
  - Tighter statistical bounds
  - Posterior corruption security
  - Stake-based weights
- **Implementation & benchmarking**

Accélérateur
de la transformation
numérique

Thanks for your attention

Any questions?

www.irt-systemx.fr

PeerCube[1] in details

- **Statistical security against byzantine adversaries**
  - Malicious users are scattered across all shards
  - Shard assignment to byzantine credentials follows an multivariate hypergeometric law
  - Probability of corrupting a single shard core is bounded by Hoeffding bound
  - Overall corruption probability with the union bound

$$\mathbb{P}[\forall \mathcal{S}, \#byz(\mathcal{S}) > \mu C_{size}] < e^{-(2\varepsilon_{\mu}^{2} C_{size} - \ln(N_{shard}))}$$

- **Actually tolerates up to $\mathcal{F}$ corrupted shards**

[1] Anceaume, E., Ludinard, R., Ravoaja, A., & Brasileiro, F. Peercube: A hypercube-based p2p overlay robust against collusion and churn. SASO 2018, IEEE.

Simplifications

- **Each UTXO has a bounded amount of stake**
- **Each output contains a public key**
- **Clean recovery from corruption**
- **New users can safely connect to the network**

Composition update in details

- **Composition of a shard $\mathcal{S}(h) = (\mathcal{S}_c(h), \mathcal{S}_s(h))$**
- **Join requests are locally stored in a buffer $\boldsymbol{b_i}$**
- **Upon reception of block $\boldsymbol{B_h}$**
  - Users in $\mathcal{S}_c(h-1)$ run a vector agreement with input b_i
  - Users in $\mathcal{S}_c(h-1)$ run a random beacon
  - $\mathcal{S}(h)$ is computed such that
    - All expiring credentials are removed
    - Joining request are added in the spare set
    - The core set is filled using the beacon randomness
  - $\mathcal{S}_c$(h) is sent to all shards