

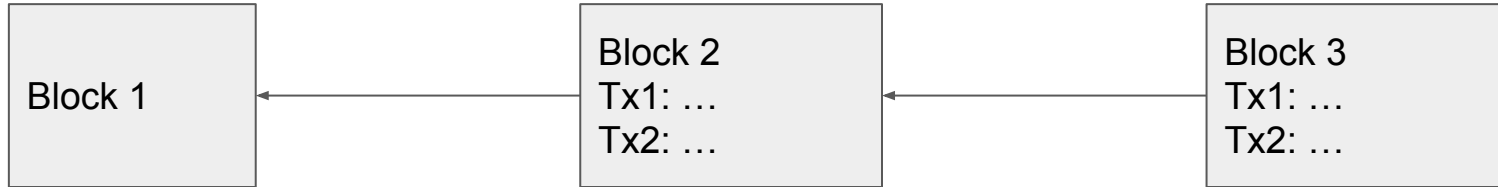
# Erasure Coded Sharding

Bribery-resistant sharding for scalable blockchains

Jessica Taylor ([jessi.liu.taylor@gmail.com](mailto:jessi.liu.taylor@gmail.com))

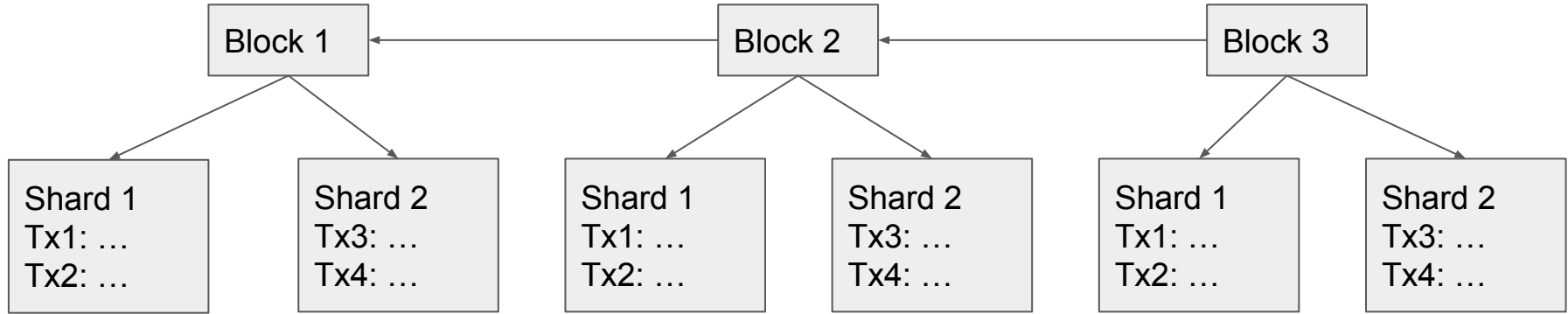
[gigascaling.net](http://gigascaling.net)

# Regular blockchains (Bitcoin, Ethereum)



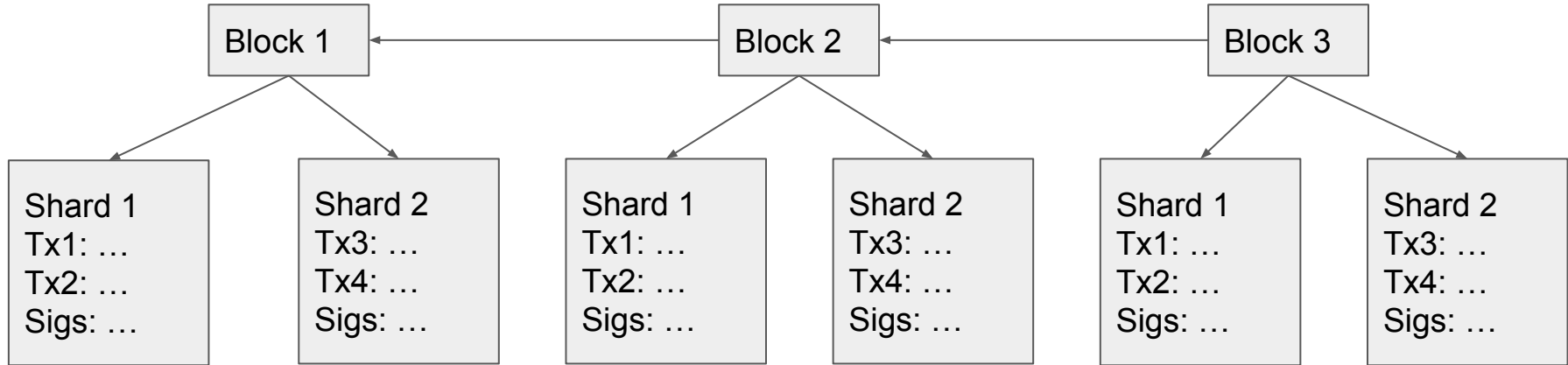
- “A  $\rightarrow$  B” means “A includes hash code of B”
- Proof-of-work or proof-of-stake (details omitted)
- All full nodes download and verify every transaction
- Problem: this limits how many transactions per second there can be

# Sharded blockchain






- Regular full nodes only download top chain
- Shard-level verifiers download data for a given shard
- Each shard is responsible for a subset of accounts
- Split transactions into “send” and “receive” (like actor model)
- Problem: how do regular full nodes know shard-level data is valid?

# Random validator pools

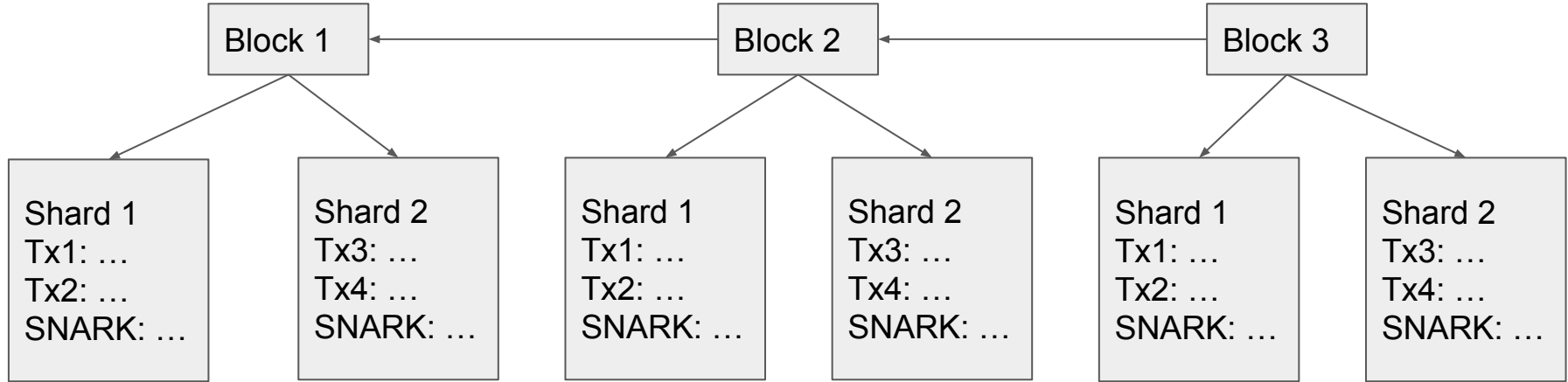


- Call together a validator committee by selecting ~1000 random stakers
- If  $> \frac{2}{3}$  of stakers are honest, with high probability  $> \frac{1}{2}$  of the committee is honest
- Regular full nodes only download signatures, not transactions
- Can be iterated multiple levels (Inductive Consensus Tree Protocol, [ictp.io](https://ictp.io))
- Problem: bribery

# 3 types of validators

-  Altruistic: Always honest
-  Malicious: Trying to sabotage the system, even at cost to self
-  Greedy: Honest unless bribed sufficiently
- If  $> \frac{1}{3}$  of stakers are malicious or greedy, committees are likely to be bribable.
- Only one committee must be bribed to mint fake currency; bribery is cheap.

# zk-SNARK verification




- Each valid shard contains a zero-knowledge proof of its validity
- Full nodes download and check SNARKs, not transactions
- Can be iterated multiple levels (and compressed) with recursive SNARKs
- Problem: data availability attacks (reveal SNARK, withhold data)

# Data availability attacks



- Shard-level validators may provide a SNARK showing the shard is valid, but withhold the shard data (incl. transactions) from others
- This prevents individual accounts from proving account states or unspent transactions, which prevents the account's funds from being accessible
- Can we use random validator committees to mitigate this? They can be bribed, knocking out lots of account data
- Solution concept: Can we force a large *percentage* of committees to be knocked out to make *any* shard's data unavailable?
- (inspired by [PolyShard](#), algorithm details differ; PolyShard requires all validators to download all new blocks)

# Erasure code





- 
- $k$  blue squares = original data chunks (each the same # bytes), 5 in this case
- Redundancy factor  $\alpha$ , 3 in this case
- $k(\alpha - 1)$  green squares = data augmentation
- Can recover  $k$  blue squares given *any*  $k$  blue or green squares!
- Reed-Solomon code works by polynomial interpolation: interpret blue squares as polynomial coefficients in a finite field, get green squares by evaluating polynomial at more points, recover blue squares by fitting a polynomial
- Encoding is  $\sim O(\alpha k^2)$ , decoding is  $\sim O(\alpha^2 k^3)$  for Reed Solomon codes
- Encoding is  $\sim O(\alpha k)$ , decoding is  $\sim O(\alpha k)$  for Raptor codes (need  $\sim 10$  extra chunks)
- Reed-Solomon parallelizes well, tractable to do in a SNARK







# Erasure coding shard data

- Shard 1 data: 
- Shard 2 data: 
- Etc, for x shards
- The data consists of  $k\alpha$  chunks for each shard ( $xk\alpha$  total chunks)
- Split into  $k\alpha$  columns; we can recover all data from k full columns
- Split stakers into  $k\alpha$  equal-sized pools, each responsible for storing 1 column
- If enough honest stakers store all their data, we can recover everything!
- Storage per staker = x chunks ( $1/k$  of original data), it scales decently
- Use SNARKs to prove that enough signatures exist that a significant fraction of stakers would have to lie about their storage for data to be unrecoverable
- In case some chunks are dropped, use a different SNARK for each shard

# Reducing storage by erasure coding each column

- Shard 1 data: 
- Shard 2 data: 
- Redundant 1: 
- Redundant 2: 
- Secondary redundancy factor  $\beta$  (2 in this case)
- We can recover a column with  $x$  of  $\beta x$  chunks
- We can recover everything with  $(\alpha + \beta - 1)xk$  of  $\alpha\beta xk$  chunks
- Split stakers into  $\alpha\beta xk$  equal-sized pools, each responsible for storing 1 chunk
- If enough honest stakers store their chunk, we can recover everything!
- Storage per staker = 1 chunk ( $1/(xk)$  of original data), it scales very well

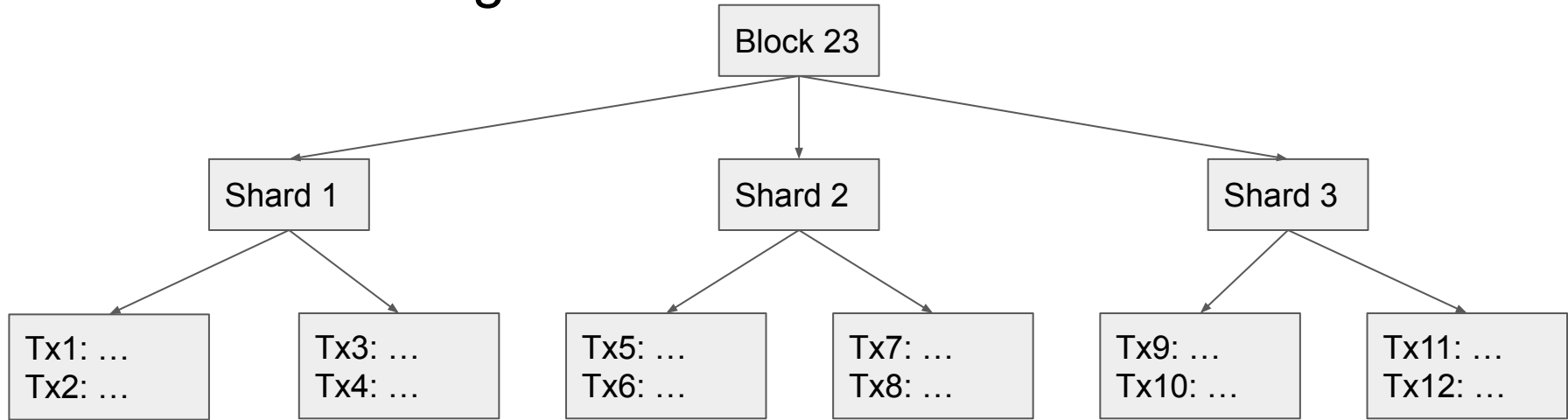
# Security analysis

- Set threshold  $\gamma$ , proportion of chunks that must be *asserted* to be stored
- Proportion that must *actually* be stored is  $(\alpha + \beta - 1)/(\alpha\beta)$
- Secure if less than  $\gamma - (\alpha + \beta - 1)/(\alpha\beta)$  proportion of stakers act maliciously
- Malicious includes  and bribed 
- Cost of bribery  $\approx$  stake amount (easy to prove chunk unavailability & punish)
- Functional if more than  $\gamma$  proportion of stakers are honest most rounds
- Acting honestly includes  and  (bribery won't happen most rounds)
- E.g.  $\gamma = 2/3$ ,  $\alpha = \beta = 6 \rightarrow$  secure if less than 36.1% of stakers act maliciously

# Privacy and smart contracts

- Privacy is easy since we're already using SNARKs
- Transaction data for a private transaction consists of account ID and hash code of new private state (64 bytes total)
- SNARK for a shard proves SNARKs exist for each account state transaction
- 2 types of smart contracts: per-account and independent
- Per-account smart contracts modify account data, including private data
- A single account can partake in multiple per-account smart contracts
- E.g. tokens which are similarly private to the base currency
- Independent contracts are like their own account, data is public

# Multi-level sharding



- 3 instead of 2 levels, reduced branching factor
- Reduces work per node
- Use 2d erasure code for each shard, 3d overall (decreases data efficiency, increases compute efficiency of Reed Solomon encoding/decoding)