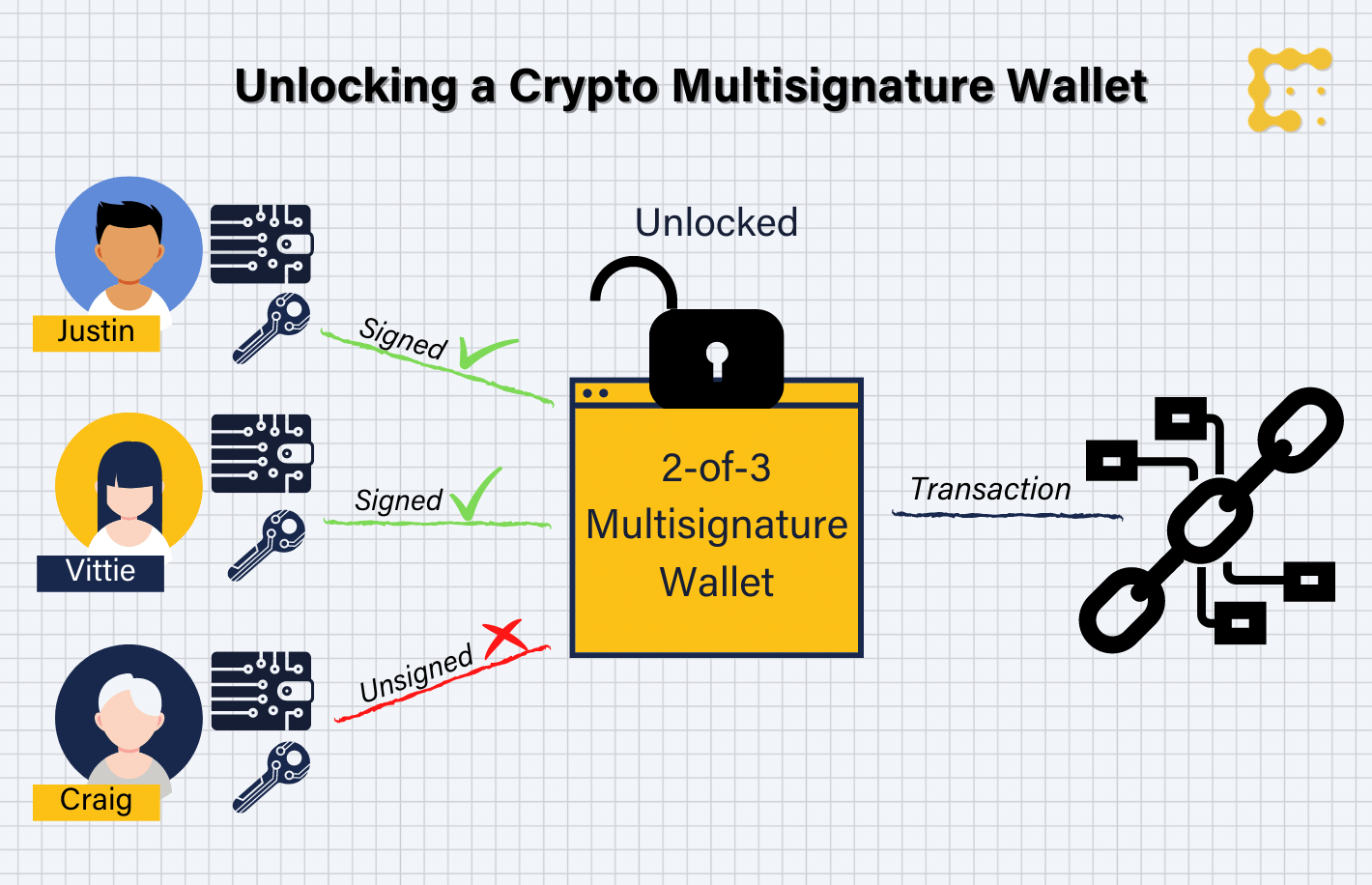
**Multisig**

A [multi-signature wallet](https://en.bitcoin.it/wiki/Multi-signature) (multisig for short) is one where, instead of an account, **a group of accounts** are required to **approve wallet operations.**

Is based on the well-known security principle of **split control.**

The goal is to eliminate a **single point of failure**, whether it's via a human error, insider breach or external compromise.



### **Armory Wallet for Bitcoin**

[Armory](https://www.btcarmory.com/) is a **full node, open-source Bitcoin wallet** with both cold storage and multisig support.

It uses a **deterministic wallet**; all addresses are derived from the same root.

Armory is designed conservatively, with a primary focus on security, evident from the choice of cryptographic schemes adopted.

### **Gnosis Safe for Ethereum**

### [Gnosis Safe](https://gnosis-safe.io/) is the most popular multisig wallet for **Ethereum**, period.

The core smart contracts are **open-source and formally verified**, which is generally a very high bar for security.

You can create and access a safe directly from your **browser**, but Gnosis is also available as a **desktop** (Windows, MacOS, Linux) and mobile application (iOS, Android).

It supports **Ethereum** (Mainnet, Rinkeby), xDai, Arbitrum, EWC, Volta, Polygon and BSC blockchains, and works with a wide range of wallets, including MetaMask, Ledger, Trezor and others.

It supports **non-fungible tokens** (NFTs) and seamlessly integrates into popular decentralised finance (DeFi) protocols to trade and manage digital assets.

### 

### **Cashmere Wallet for Solana**

[Cashmere](https://www.cashmere.finance/) is a platform to **create and manage multisig wallets** for digital assets on the **Solana** blockchain.

Cashmere is relatively new, without a lot of documentation, community or a roadmap, and it's unclear whether the smart contracts have been audited, so do exercise caution.

### **Multisig Safe for Harmony One**

[Harmony Multisig Wallet](https://multisig.harmony.one/) is a platform to **create and manage multisig wallets/safes** for digital assets on the Harmony One blockchain.

It allows you to connect existing **MetaMask and 1Wallet wallets**, and is based on Gnosis Safe smart contracts.

Harmony is a **fast and secure blockchain** for decentralized applications. The production mainnet supports 4 shards of 1000 nodes, producing blocks in 2 seconds with finality.

**Effective Proof-of-Stake (EPoS)** reduces centralization while supporting stake delegation, reward compounding and double-sign slashing.

**Key Innovations Harmony One**

**Fully scalable Architecture**

Harmony fully scalable on all three aspects of the blockchain: **network, storage and transaction processing.**

**Secure Random Sharding**

Harmony’s sharding process is provably secure against **shard attacks** because the network validators are randomly assigned and shuffled among shards.

The randomness used in the sharding is obtained with a distributed randomness generation algorithm (based on VRF and VDF) which is **unpredictable, un-biased, verifiable and scalable.**

**Cuckoo Rule** - cuckoo assign a random number to a new node that wants to join the network, the position of that random number indicates the shard number for the node and then *cuckoo (move)* nodes that are “close” to that random number (“existing eggs”), to new random numbers which give them new shards.

**Efficient and fast consensus**

Harmony’s consensus algorithm is called ***Fast Byzantine Fault Tolerance or FBFT.***

FBFT is a highly **efficient and speedy consensus algorithm** built upon the famous PBFT (Practical Byzantine Fault Tolerance) algorithm

Harmony’s FBFT is able to confirm blocks within 2 seconds - the adoption of aggregated BLS (Boneh–Lynn–Shacham) signature.

FBFT is also highly optimized in network message processing and block proposal pipelining so that the consensus can scale to hundreds of validators at the same time.

**Effective proof of stake**

Harmony is a **Proof-of-Stake blockchain** which is energy efficient and low-cost for node runners.

The process to elect validators is called **Effective Proof-of-Stake** (EPoS) which is the first sharding-focused PoS mechanism that prevents stake centralization.

In EPoS, validators with a **large amount of staked tokens** are obligated to **run more nodes** to support the network while validators with less stake run fewer nodes.

Besides, EPoS is able to randomly and evenly distribute the stakes among all shards so no shard is less secure than other shards.

## **Scalable Networking Infrastructure**

Harmony’s network layer is based on the **industry-leading p2p protocol named libp2p.**

We use libp2p’s gossip protocol for **network message broadcasting and stream protocol** for decentralized state synchronization.

To achieve high performance, we adopt **RaptorQ fountain code and use Adaptive Information Dispersal Algorithm** to quickly and efficiently broadcast large blocks.

Harmony also features a design where Kademlia routing is used to achieve cross-shard transactions that scale logarithmically with the number of shards.

## **Asynchronous Cross-Shard Transactions**

Harmony supports **cross-shard transactions** to achieve composability of assets and smart contracts between shards.

**Key Features**

**Secure, Random State Sharding**

Sharding is proven to scale blockchains without compromising security and decentralization.

We divide not only our network nodes but also the **blockchain states into shards, scaling linearly** in all three aspects of machines, transactions and storages.

To prevent **single shard attacks**, we must have a sufficiently large number of nodes per shard and cryptographic randomness to re-shard regularly.

Each shard has 1/4 of nodes for strong security guarantee against Byzantine behaviors.

use **Verifiable Random Function (VRF)** for un-biased and unpredictable shard membership.

# **Fast Consensus w/ Instant Finality**

Harmony has innovated on the battle-tested **Practical Byzantine Fault Tolerance (PBFT)** forfast consensus of block transactions.

Our Fast BFT (FBFT) leads to **low transaction fees and 1-block-time finality** in Harmony Mainnet.

We use **Boneh–Lynn–Shacham (BLS)** constant-sized signatures to commit blocks in a single round of consensus messages. We achieve **2-second block tim**e with view changes in production against adversarial or unavailable leaders.

# 

# **Effective PoS & Token Economics**

Harmony has designed a novel **Proof-of-Stake (PoS) mechanism** for network security and economics.

Our Effective Proof-of-Stake (EPoS) reduces centralization and distributes rewards fairly to thousands of validators.

Our staking mechanism supports **delegation and reward compounding**.

**Harmony’s FBFT consensus involves the following steps:**

1. The **leader constructs the new block** and broadcasts the block hash to all validators. This is called the **“announce” phase.**
2. The **validators** check the validity of the message, **sign the block has**h with a BLS signature, and send the signature back to the leader.
3. The leader waits for **valid signature**s with more than 2/3 voting power from validators (including the leader itself) and aggregates them into a **BLS aggregate signature(Boneh–Lynn–Shacham)**. Then the leader broadcasts the new block and the aggregated signature along with a **bitmap** indicating which validators have signed. Together with Step 2, this concludes the “prepare” phase.
4. The validators check that the aggregate signature has at least **2/3 of total voting power,** verify the new block, sign the received block from Step 3, and send it back to the leader.
5. The leader waits for **valid signatures** with more than 2/3 voting power (can be different signers from Step 3), aggregates them together into a **BLS aggregate** signature, and creates a **bitmap for all the signers**. Finally, the leader commits the new block with the aggregate signatures and bitmaps into local DB, and broadcasts the aggregate signature and bitmap for all validators to confidently **commit the block** too. Together with Step 4, this concludes the “commit” phase.

**State Sharding**

In Harmony, each shard maintains it's **own chain of blocks and state database**. Therefore, the validators of each shard only need to **store 1/N of the global state,** where N is the number of shards.

The consistency between states from different shards are guaranteed by the property of eventual atomicity of cross-shard transactions, which guarantees that double spending between shards can not happen.

**Network Sharding**

Harmony's validator **network is also divided into shards** where each shard involves a separate set of validators connected closely with each other and running consensus between themselves. Most of the time, validators communicate with other validators within the same shard to reach consensus or synchronize blocks.

In cases of **cross-shard transactions and beacon chain synchronization**, validators from **different shards send messages** across shards through the globally connected network.

**Transaction Sharding**

**Transactions in Harmony blockchain are sent to and processed by a specific shard instead of all shards.** This way, shards can process transactions in parallel which greatly improves the overall transaction processing capacity of the blockchain. Users need to specify a field named **shard\_id** in the signed transaction which indicates which shard this transaction belongs to.

For cross-shard transactions, another field named t**o\_shard\_id** is needed to indicate the destination shard while the shard\_id

# **Epochs**

An epoch in Harmony blockchain is a **pre-determined period of time** when the validator committees of shards stay unchanged.

**In Harmony mainnet, one epoch is 32768 blocks which translates to around 18.2 hours.** In Harmony testnet, one epoch is 8192 blocks which is around 4.6 hours.

# **Crosslinks**

Crosslink is an important piece of data which is **sent from shard chains and stored in beacon chain.**

A crosslink contains data for **block signatures and the block identifier data** such as block hash, block number, view id and epoch etc.

When a **new block is confirmed in a shard chain**, the corresponding crosslink will be created and sent to the beacon chain. Upon the receipt of the crosslink, **the beacon chain verifies its signature** and checks that it’s from the canonical chain of the shard. Successfully verified crosslinks will be **added in the new block of the beacon chain** to permanently endorse the block of the shard chain as canonical.