Lighthouse

Performance & Security for Ethereum 2.0

Mehdi Zerouali

@ethzed

sigma prime

AGENDA

- Introduction
- Lighthouse Client Architecture
- Performance & Security Update
- Demo
- Next Steps

Introduction

- Sigma Prime (SigP) – Information security consultancy, focused on Blockchain tech, working mostly on Ethereum
  - Security researchers, academics and software engineers working towards a secure and decentralised future

- Some of our information security clients:

ethereum foundation

status

CHEEZE WIZARDS

Chainlink

AdEx

GODS UNCHAINED

- Maintainers of Lighthouse, a Rust implementation of Ethereum 2.0:
  - **sigp/lighthouse**
  - Officially started in July 2018
  - R
  - Free & Open-Source
  - Security focussed

Introduction

Decentralisation

Allow standard consumer laptop to participate
Support participation of a large # of validators

Liveness

Network should remain live in a WWIII scenario

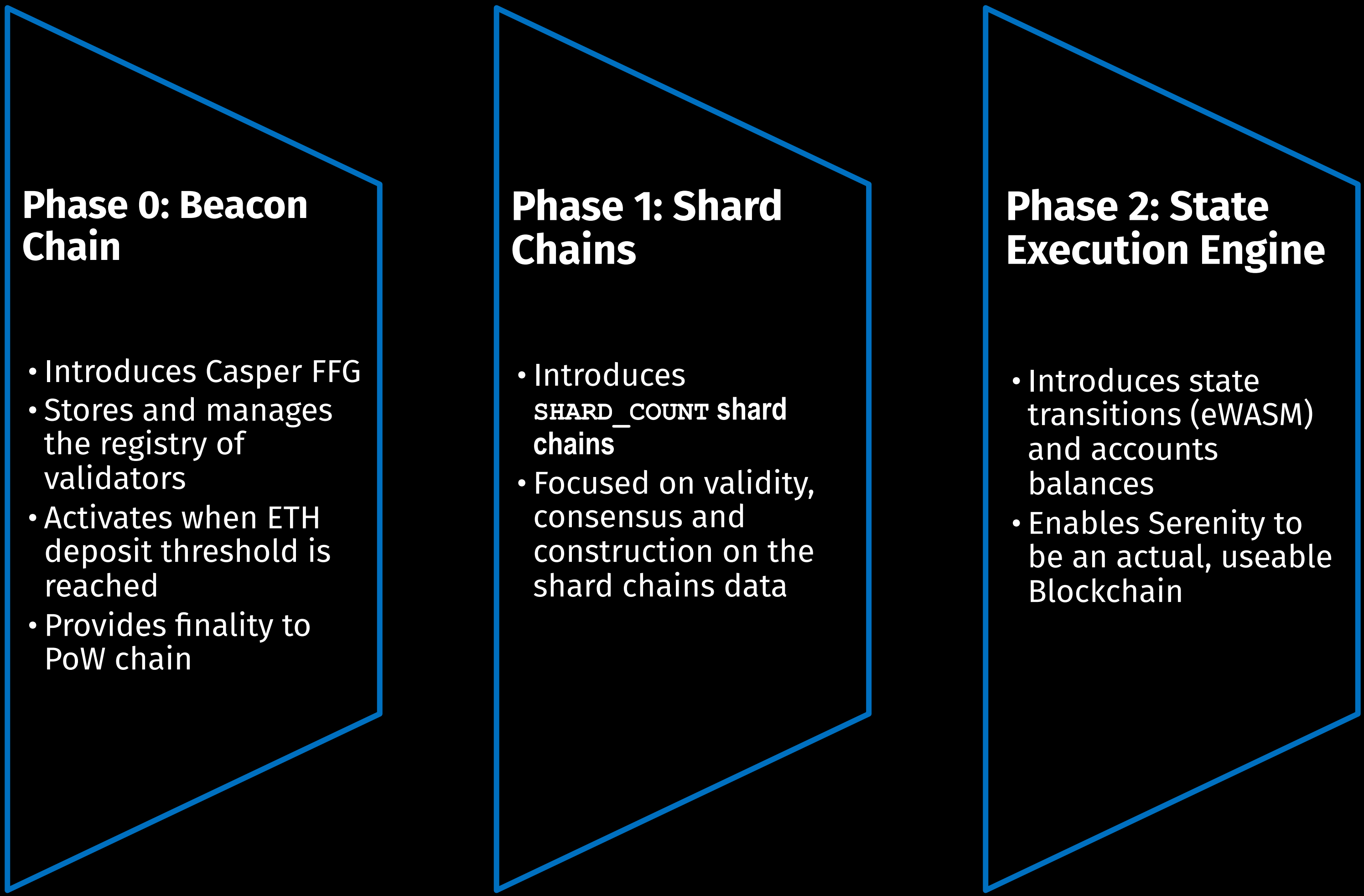
Simplicity

Minimize complexity even at the cost of efficiency

Security

Use quantum secure cryptographic primitives where possible
Allow easy swapping of cryptographic components

Introduction



Phase 0: Beacon Chain

- Introduces Casper FFG
- Stores and manages the registry of validators
- Activates when ETH deposit threshold is reached
- Provides finality to PoW chain

Phase 1: Shard Chains

- Introduces **`SHARD_COUNT` shard chains**
- Focused on validity, consensus and construction on the shard chains data

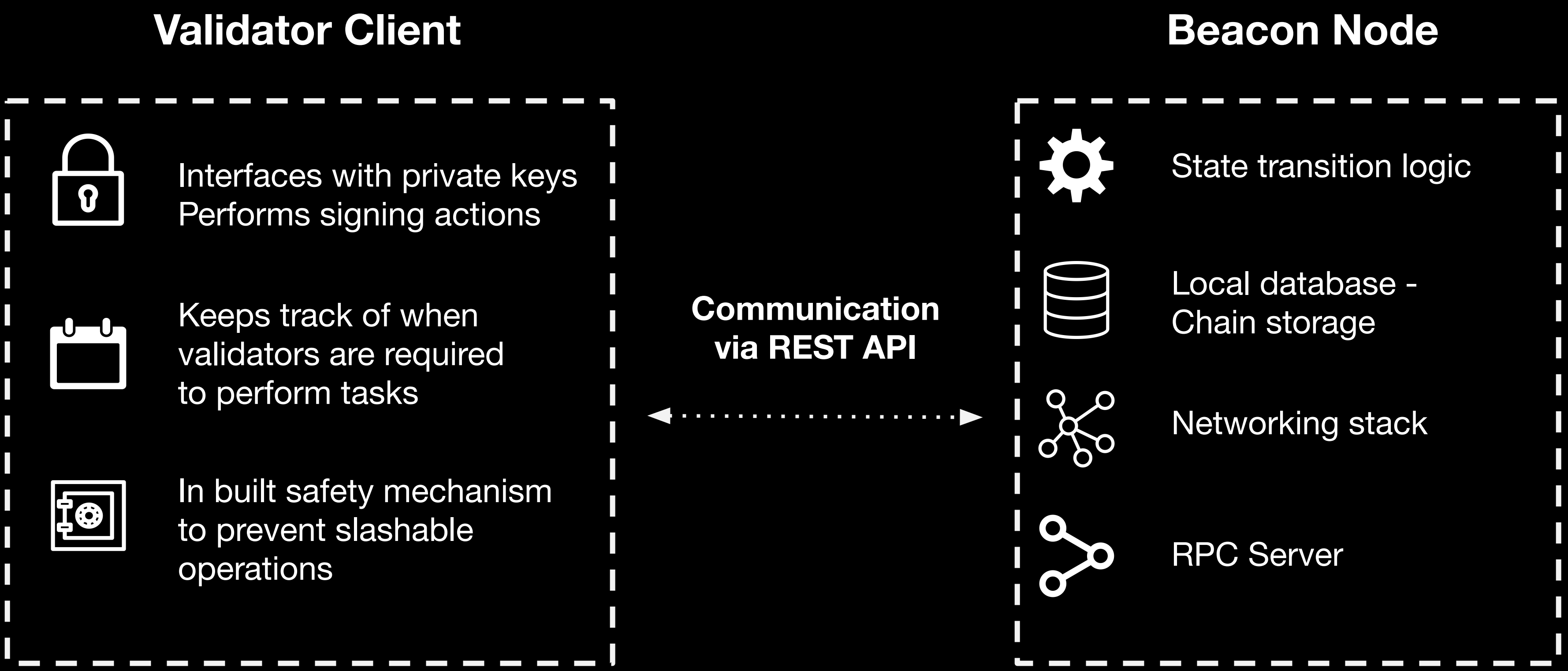
Phase 2: State Execution Engine

- Introduces state transitions (eWASM) and accounts balances
- Enables Serenity to be an actual, useable Blockchain

Lighthouse
Client Architecture

sigma prime

2 Separate Binaries



Validator Client

Interfaces with private keys
Performs signing actions

Keeps track of when
validators are required
to perform tasks

In built safety mechanism
to prevent slashable
operations

Communication via REST API

Beacon Node

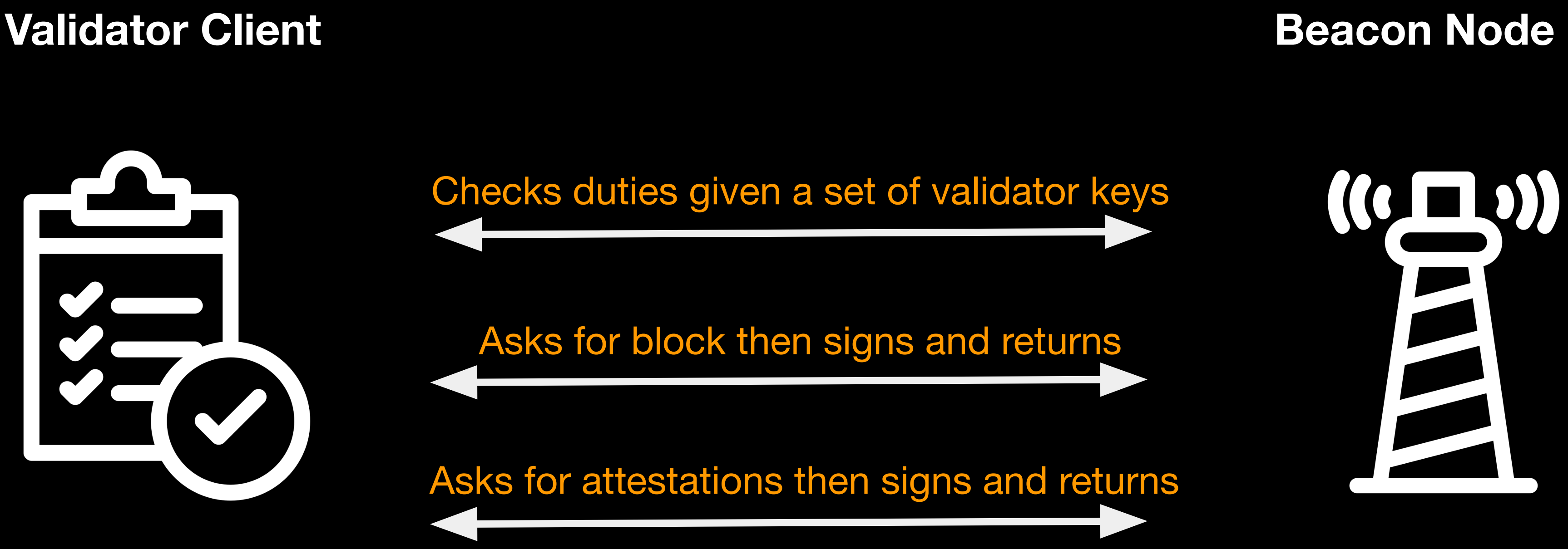
State transition logic

Local database -
Chain storage

Networking stack

RPC Server

VC <-> BN communication



Validator Client

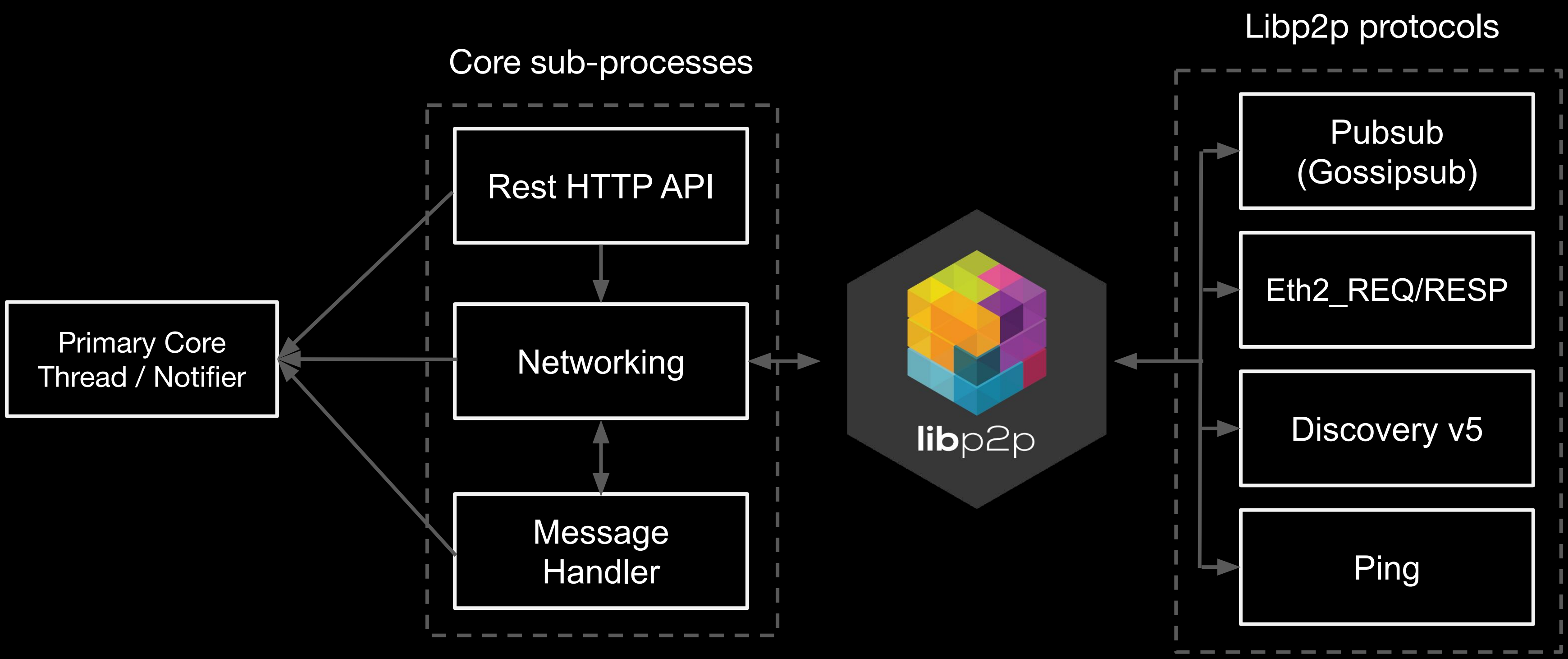
Beacon Node

Checks duties given a set of validator keys

Asks for block then signs and returns

Asks for attestations then signs and returns

Beacon Node - Networking Focus



Performance & Security Update

sigma prime

Major contributor to networking spec

- Standardisation for interoperability
- Outline the technologies needed:
  - SecIO for testnet and Noise for mainnet
  - Discv5 for peer discovery
  - Gossipsub for Pub/Sub
- Improves and simplifies RPC
- Compression and encodings
- Forward-looking and Future-proof

Identification of spec bugs

- Block header signature issue -> **Fixed in 0.5.1**
- Unsorted attester slashing indices -> **Fixed in 0.6.3**
- Confusion with merkle root of a single element -> **Fixed in 0.7.0**
- Off-by-one error in shuffling -> **Fixed in 0.8.1**
- Incorrect start shard for compact committees root -> **Fixed in 0.8.2**

Highly competitive performance

Rust is generally pretty fast, plus:

State transition

- Computational complexity reductions
- Caching optimizations
- Highly parallelizable

Fork choice

- First and only implementation of LMD GHOST “reduced-tree” optimization from IC3 2019 (Cornell)
- 5x speed improvement on previous impl

BLS cryptography

- Maintaining an optimized fork of the Apache Milagro library
- Working with cryptographers on the BLS standardization effort
- Prototyping new hash functions

Ongoing performance analysis

- Metering via Prometheus
- Benchmarking and reporting back to EF research team

Ongoing optimization works

Lots more planned:

BLS

- Bulk-signature optimization: 33% reduction in signature verification time
- Ongoing research

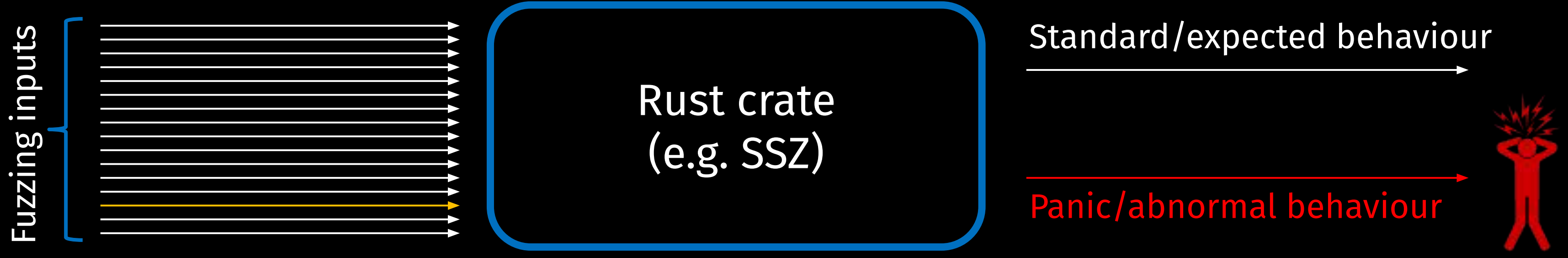
Storage (on-disk database)

- Geth-style hot/cold database for fast head operations and efficient historical storage

Fast-sync

- Sync forwards from a recent finalized checkpoint, optionally downloading historical data

Security hardening via fuzzing



```
!8;`kZN(d94LGCS*GN}=j)k:,pWwM?+#znJ'7q@%ua'6bL@|{'{!~'~-`@-@&\{~'{:'`
```

Security hardening via fuzzing

- Networking stack: *discv5, gossipsub, ENR*
  - Memory allocation bug found within a dependency
  - Bounds checking bug within discv5
- Serialization (SSZ):
  - Bug in SSZ decoding for bitfields
- State transition functions:
  - Overflow bug in processing transfers
  - Also caught in the spec by @protolanbda! Updated in v0.8

Advanced metrics monitoring

- Keeps track of:
  - Total validator balances
  - Blocks processed
  - Fork choice head count
  - Epochs since finalisation
  - Epochs since justification
  - Database size
  - …

Demo time!

Next Steps

Road ahead

- Interoperability
- Moar Optimisations
- Phase 1 & 2 Prototyping

CONSENSYS

- Security hardening & external audit

Questions?

Mehdi Zerouali

@ethzed
@sigp_io

sigma prime