



December 2018

Dmitry Meshkov

Ergo platform overview

Outline



Ergo vision

Decentralization

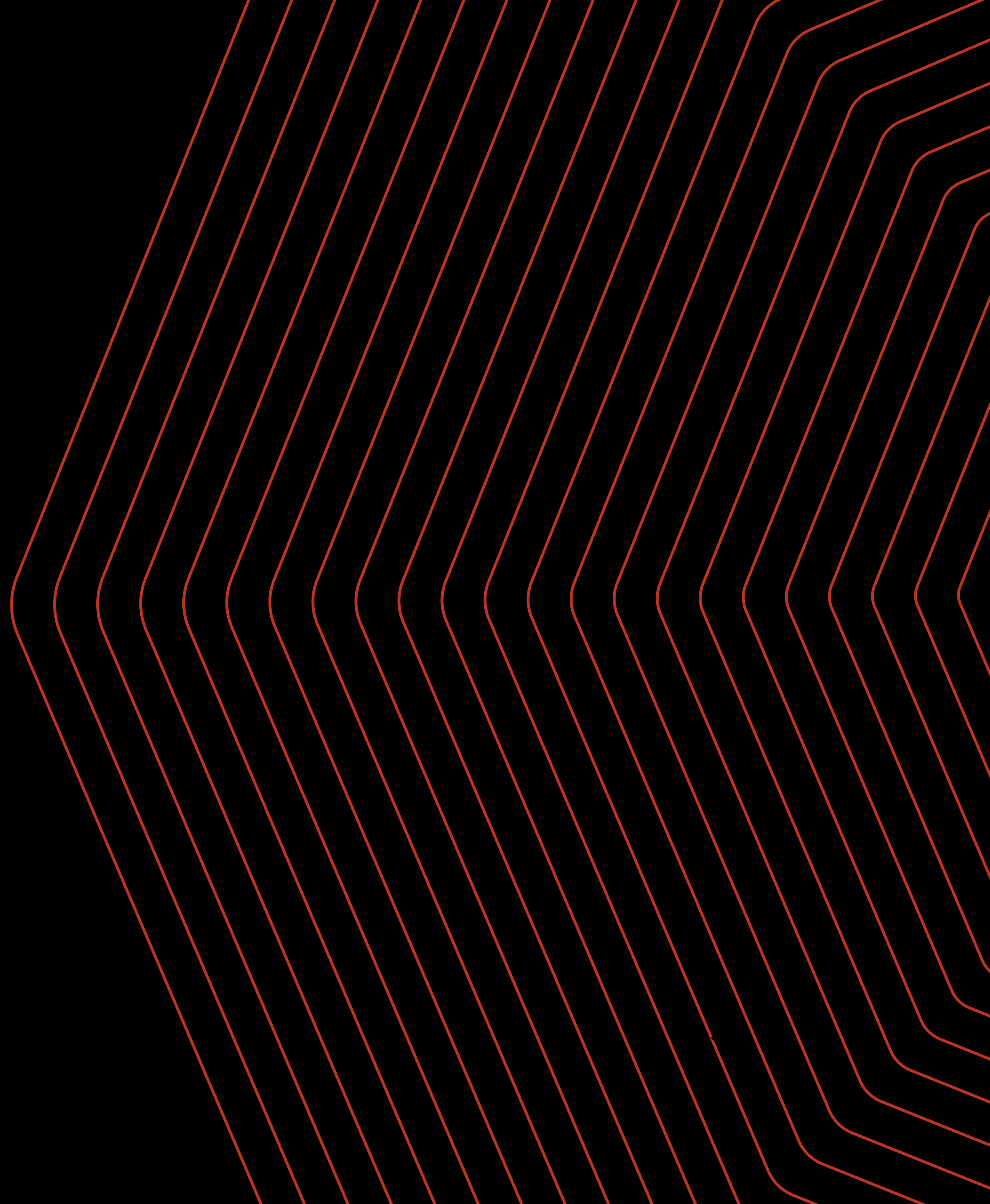
Survivability

Applicability

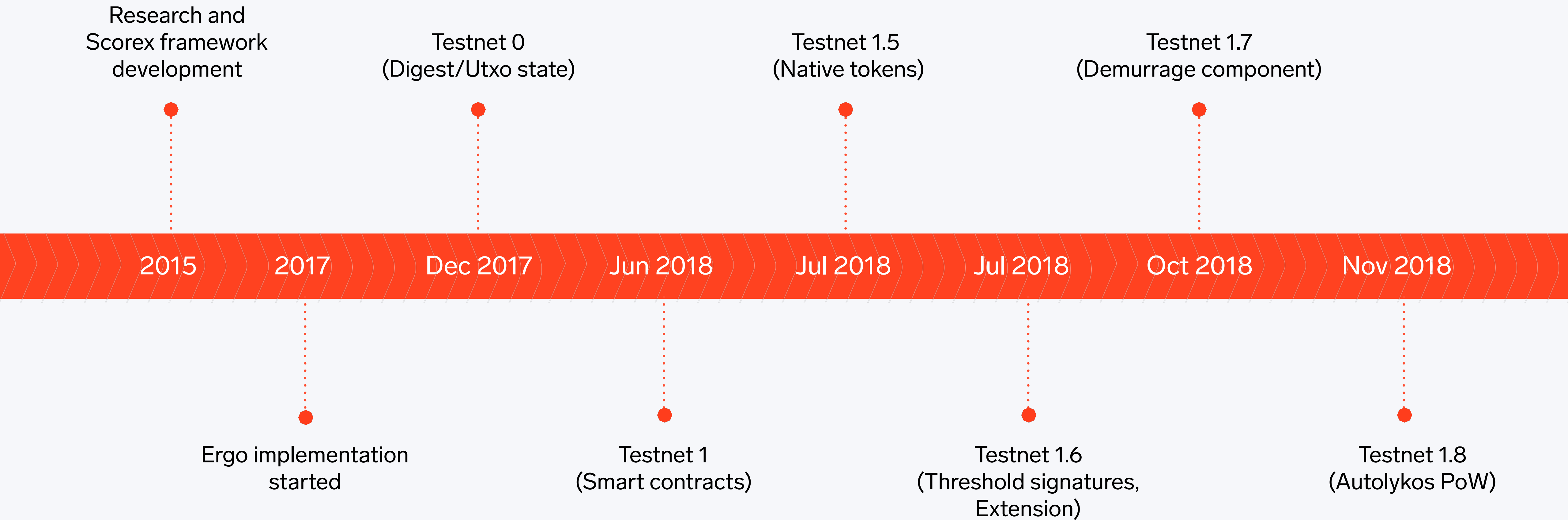
Roadmap

Part 1

Ergo vision



History



Vision

Why to start a new cryptocurrency?

- Huge hype of cryptocurrencies, but technology stuck
- Blockchain 2.0, 3.0, ..., while actually we are still at 1.0
- New protocols are trying to achieve high throughput, complicated smart contracts, ...
- .. while sacrificing decentralization, promising that it will be achieved somewhere in the future

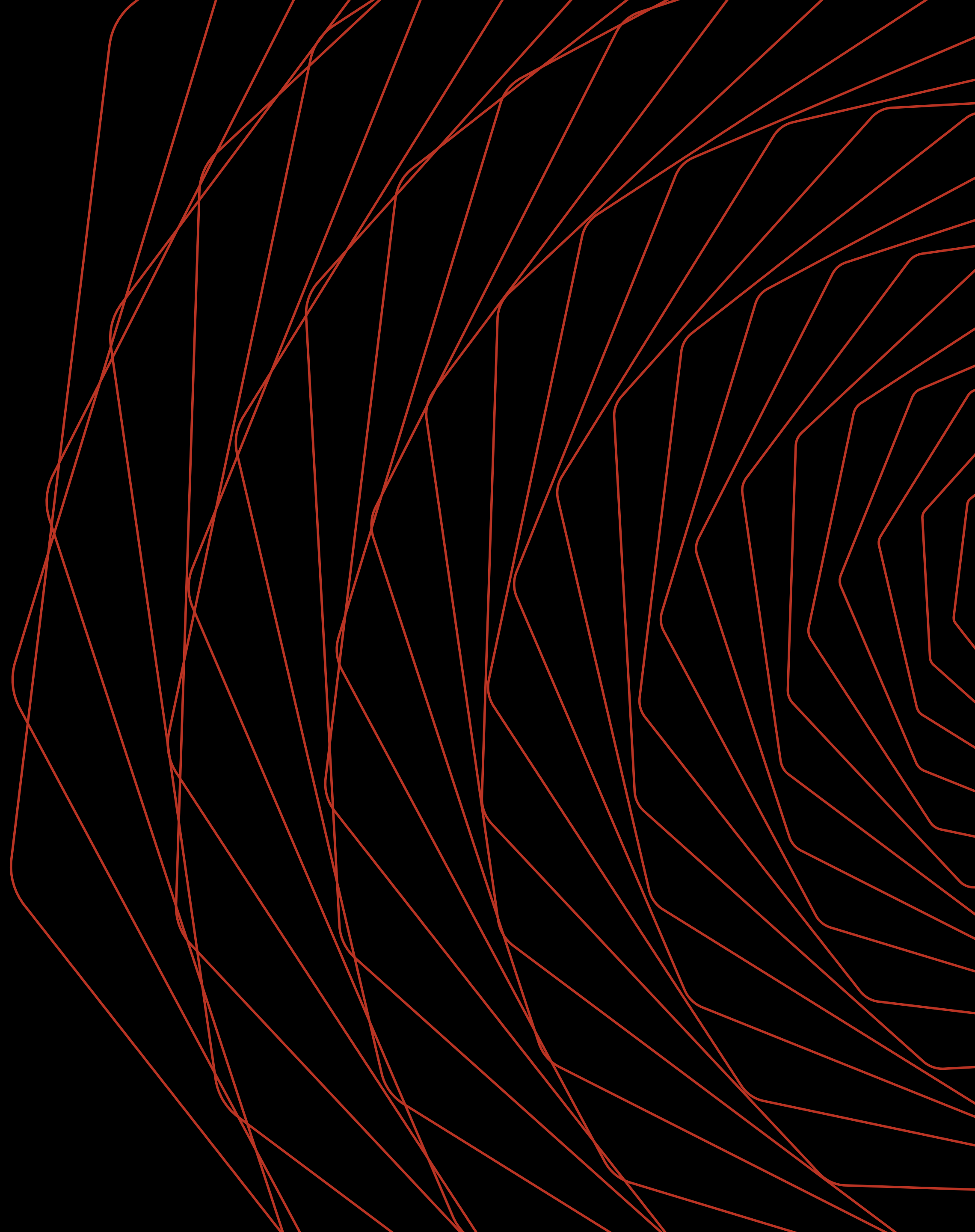
Vision

Ergo idea

- Blockchain 1.1 – a major update to blockchain technology without breaking changes
- Truly decentralized system
- Long-term survivability
- Friendly for clients and applications

Part 2

Decentralization



Decentralization

- The main value of cryptocurrencies
- Have multiple layers: miners, users, developers, ...
- Ergo should be as decentralized, as possible



Autolykos consensus protocol

Consensus: **Why Proof-of-Work?**

- 1 Widely studied and tested
- 2 Have high security guarantees
- 3 Allows new members to join the network
- 4 Light validation allows to use the blockchain without third parties

Consensus: Known Proof-of-Work Drawbacks

1

ASICs – centralize the network
around ASICs manufacturers

2

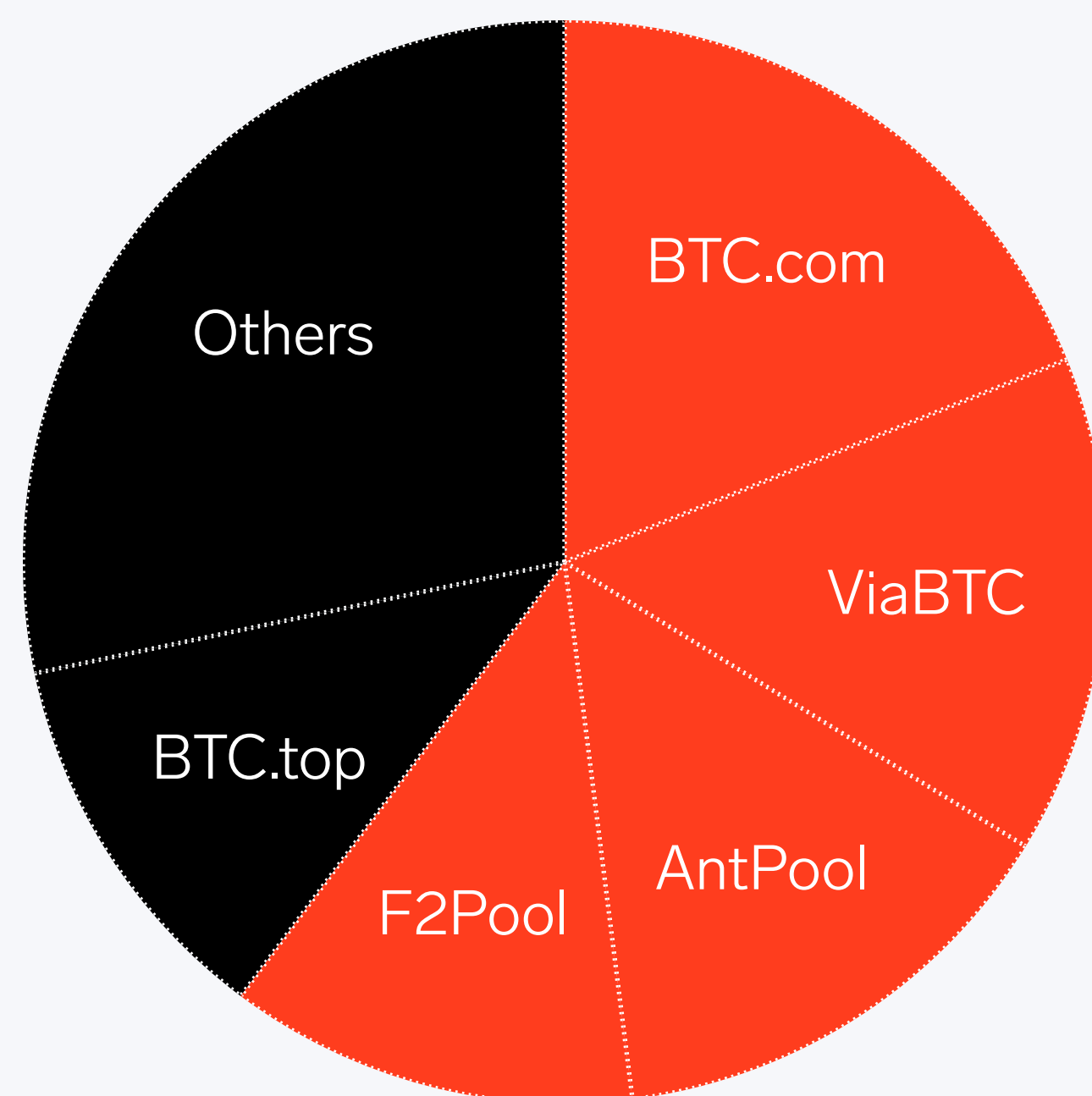
Mining pools - centralize the
network around pool operators

Consensus: ASIC-Resistance

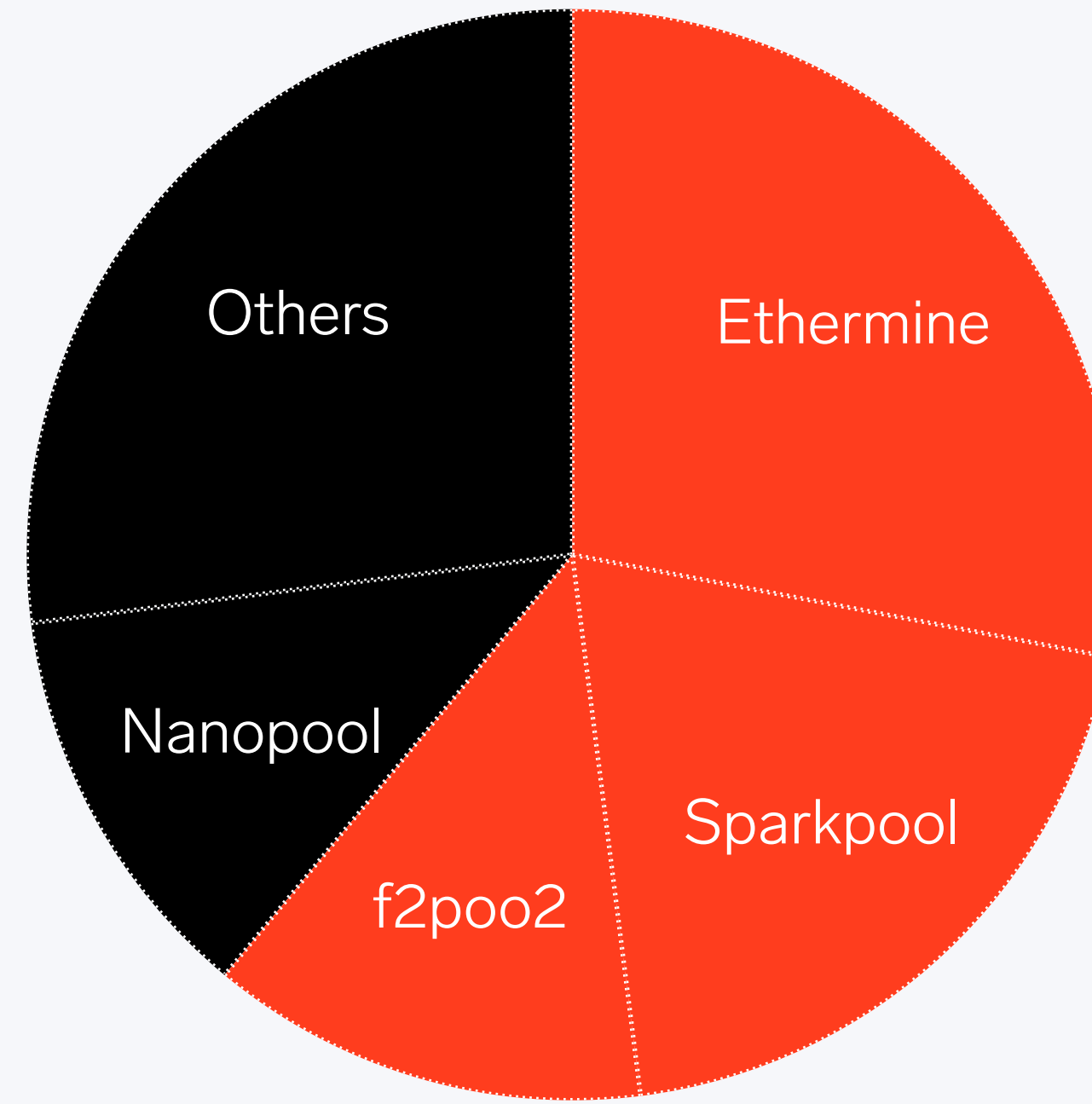
- Widely studied area
- Memory-hard computations to reduce ASICs supremacy over commodity hardware
- Autolykos is based on the one list k-sum problem, that is similar to the known Equihash PoW
- Miner should keep the whole list of elements (2Gb) in RAM or recalculate the same elements 104 -... times

Consensus: Mining pools

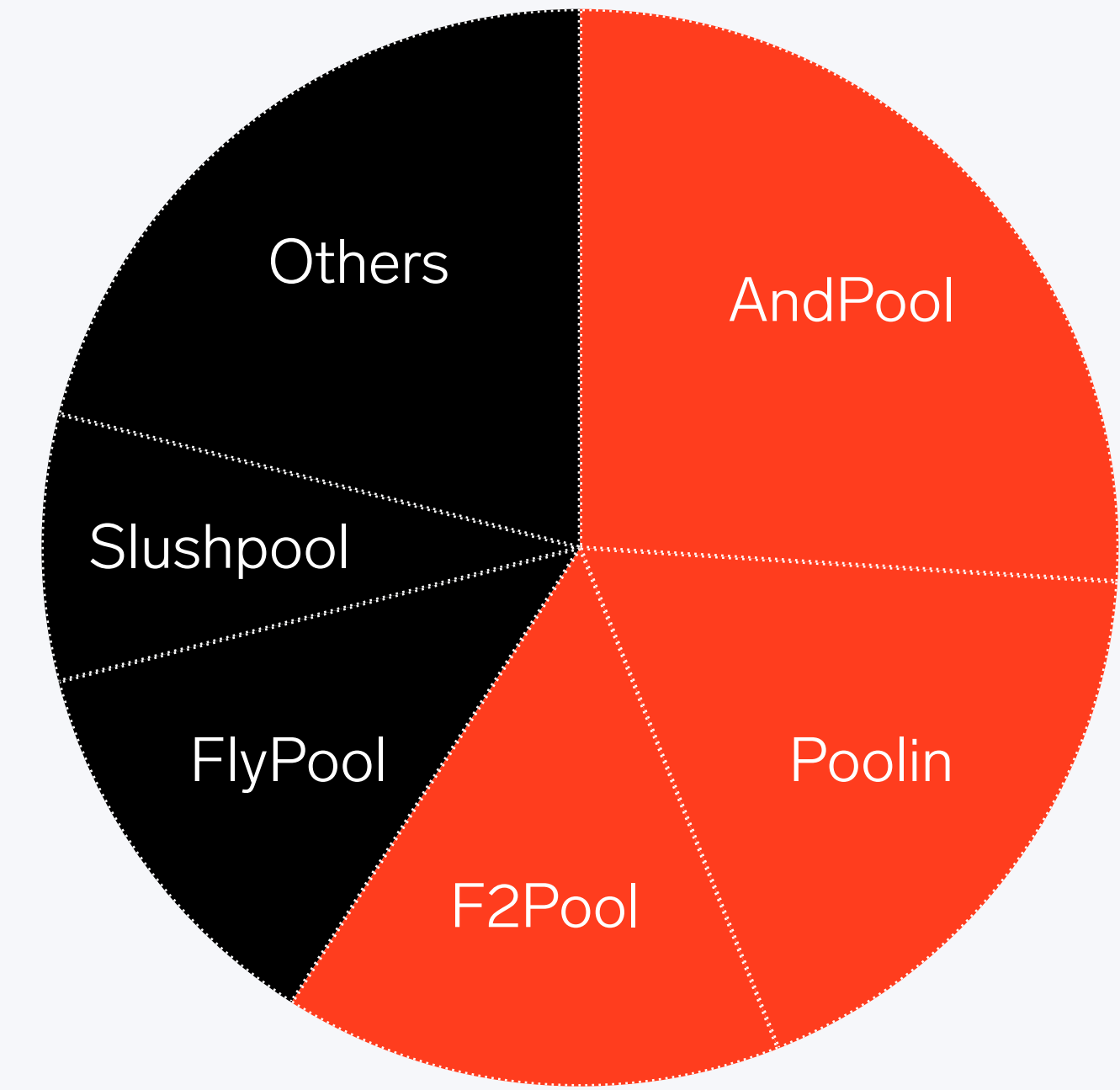
Regardless of the PoW algorithm, 2-4 pools control the network



BTC



ETH



ZEC

Hashrate distributions for 06.11.2018 (for 24 hours), taken from blockchain.com, etherchain.org, explorer.zcha.in

Consensus: Mining pools

- Non-Outsourceable puzzles: efficient puzzle solving requires access to private key
- Just few proposals before Autolykos
- Ergo is going to have the first PoW puzzle, that is both memory-hard and non-outsourceable

Consensus: Autolykos

Algorithm 1 Block mining

```
1: Input: latest block header  $hdr$ , key pair  $pk = g^{sk}$ 
2: Generate randomly a new key pair  $w = g^x$ 
3:  $m := \text{Blake2b256}(hdr.bytesWithoutPow)$ 
4: while true do
5:    $nonce \leftarrow \text{rand}$ 
6:    $J := \text{genIndexes}(m || nonce)$ 
7:    $d := \sum_{j \in J} H(j || M || pk || m || w) \cdot x - sk \mod q$ 
8:   if  $d \leq b$  then
9:     return  $(m, pk, w, nonce, d)$ 
10:  end if
11: end while
```

- q is the group order
- H is a hash function, returning values $[0, q)$
- genIndexes is a hash function, returning integer sequence
- Target interval parameter b , that is recalculated via difficulty adjustment rules
- Miner have 2 secrets: sk, x
- He is finding such a nonce, that sum of elements d does not exceeds b
- Solution contains 2 public keys, nonce, d

Consensus: Autolykos

Algorithm 2 Solution verification

- 1: **Input:** $m, pk, w, nonce, d$
- 2: require $d \in \{-b, \dots, 0, \dots, b \bmod q\}$
- 3: require $pk, w \in \mathbb{G}$ and $pk, w \neq e$
- 4: $J := genIndexes(m || nonce)$
- 5: $f := \sum_{j \in J} H(j || M || pk || m || w)$
- 6: require $w^f = g^d pk$

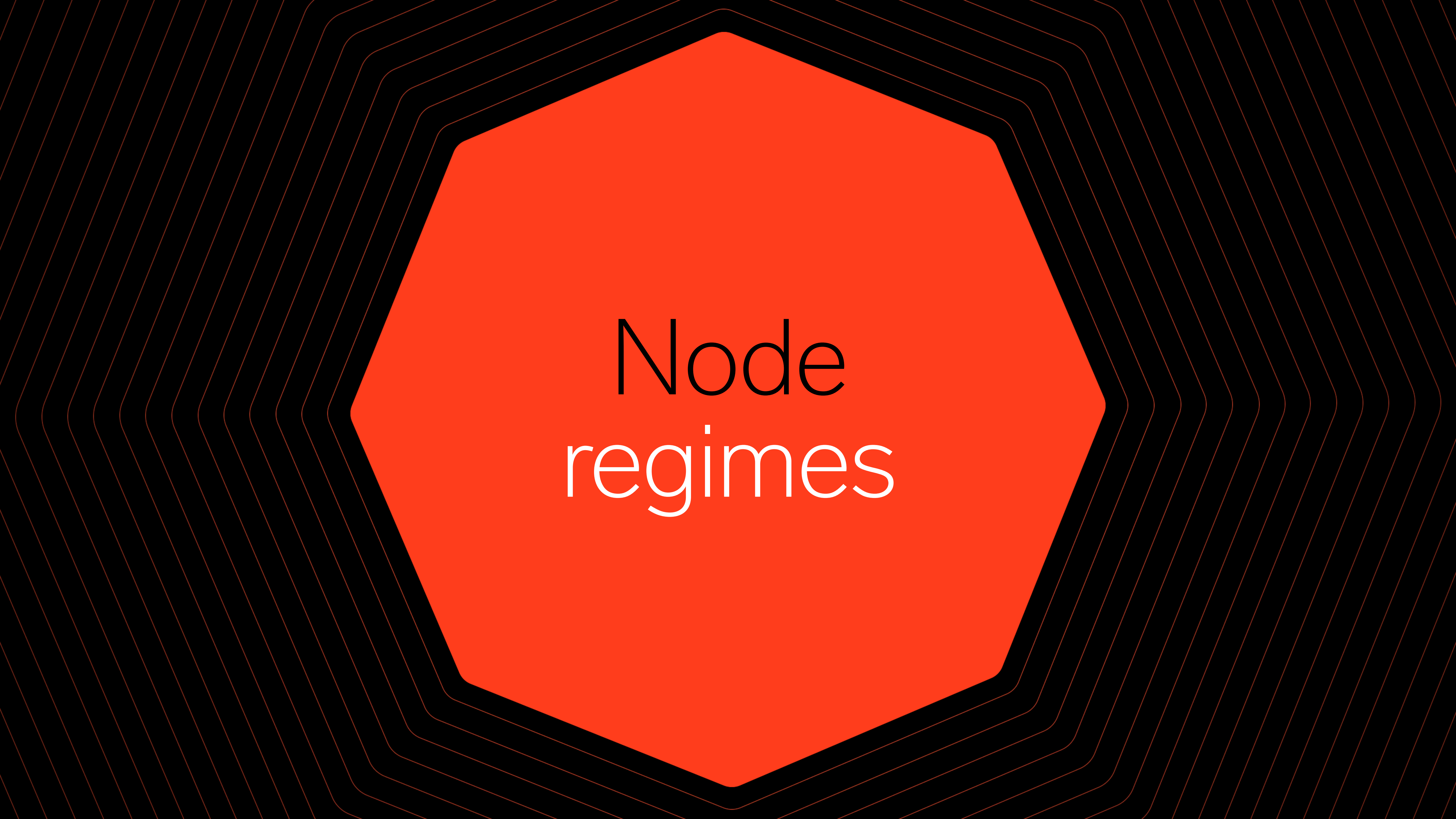
- Solution verification is done over public keys
- If miner can find a solution over public keys - he may compute dlog for pk
- Moreover, solution reveals one linear relation between secrets
- If miner is able to obtain two solutions, he may calculate secret

Autolykos: Efficiency

- Solution size is 75-107 (depends on d)
- Header verification requires verifier to calculate 1 genIndexes hash, 32 hashes H and perform two exponentiations in the group
- Scala implementation allows to verify block header in 2 ms (Intel Core i5-7200U, 2.5GHz)

Autolykos: Small miners

- Non-outsourcability means that small miners will rarely collect rewards
- While the same amount on average
- Example: if network hashrate is $2 \cdot 10^{14}$ H/s and one GPU hashrate $2 \cdot 10^7$ H/s (stats from Ethereum), miner should have 4 GPUs to generate 1 block per year
- Mining on 1 GPU is a kind of a lottery
- In USD equivalent you should spend approximately 1 block reward to create 1 block per year



Node regimes

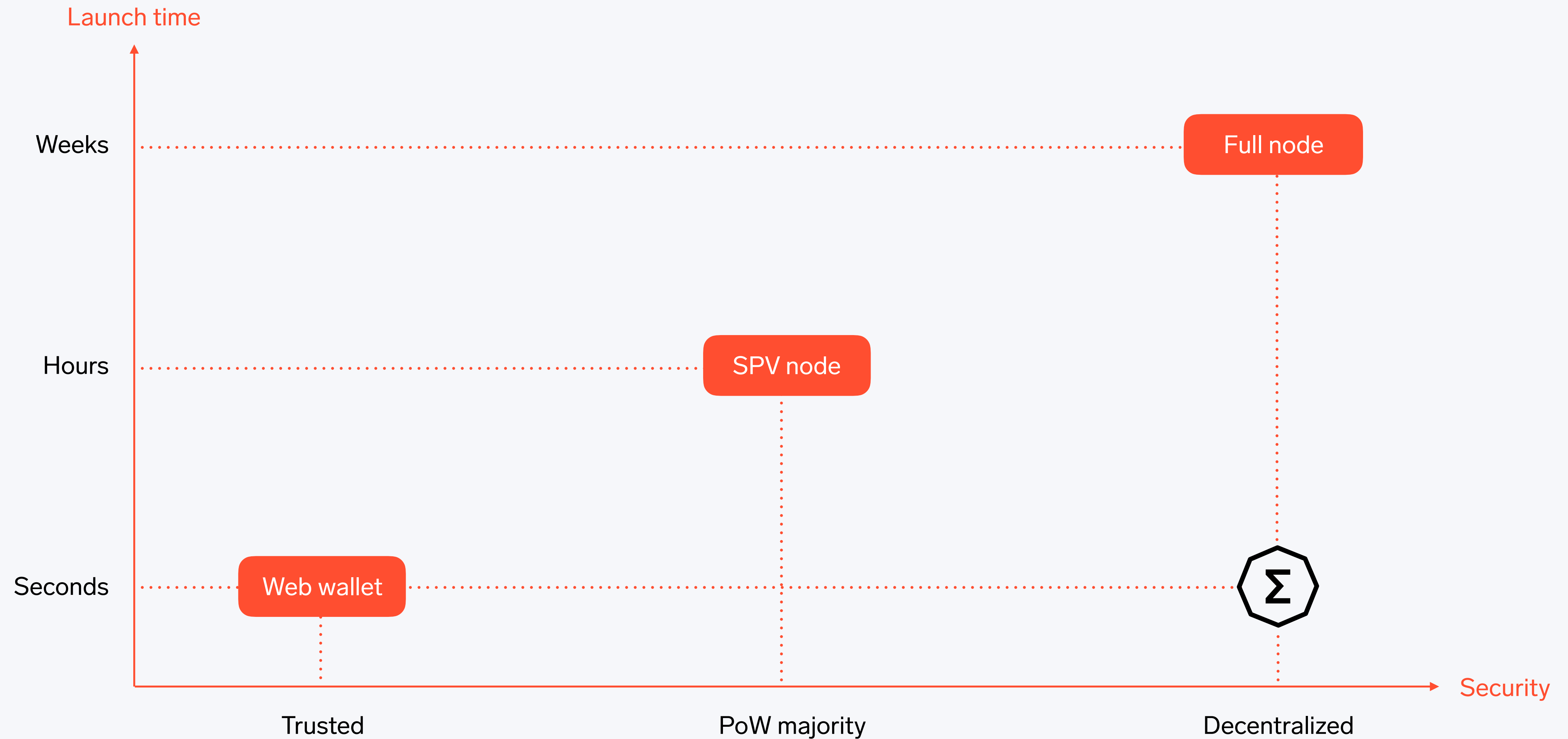
Node regimes: Now

- You must set up a node or trust someone
- Node synchronization is slow, unreliable and resource intensive
- Regular users resort to trusted solutions
- If service provider is hacked (or become malicious), users may lose their funds
- And may not even notice this, because they use trusted block explorer
- Better alternatives (e.g. SPV nodes in Bitcoin) exist, but only allows to validate some subset of rules

Node regimes: Ergo

- Ergo block header supports Non-Interactive Proofs of Proof-of-Work, that allows to synchronize the network, by downloading $< 1\text{Mb}$ of data
- Ergo state is authenticated, which enables verification of transactions without any trust and without keeping the entire state
- Flexible configuration for facilitated node regimes
- Client may only keep a subset of block headers and validate transactions using authenticated dictionaries
- Miner can keep the entire state and a fixed number of full blocks

Node regimes: Ergo



Node regimes: Ergo



It is possible to use Ergo from
a smartphone without any
trust



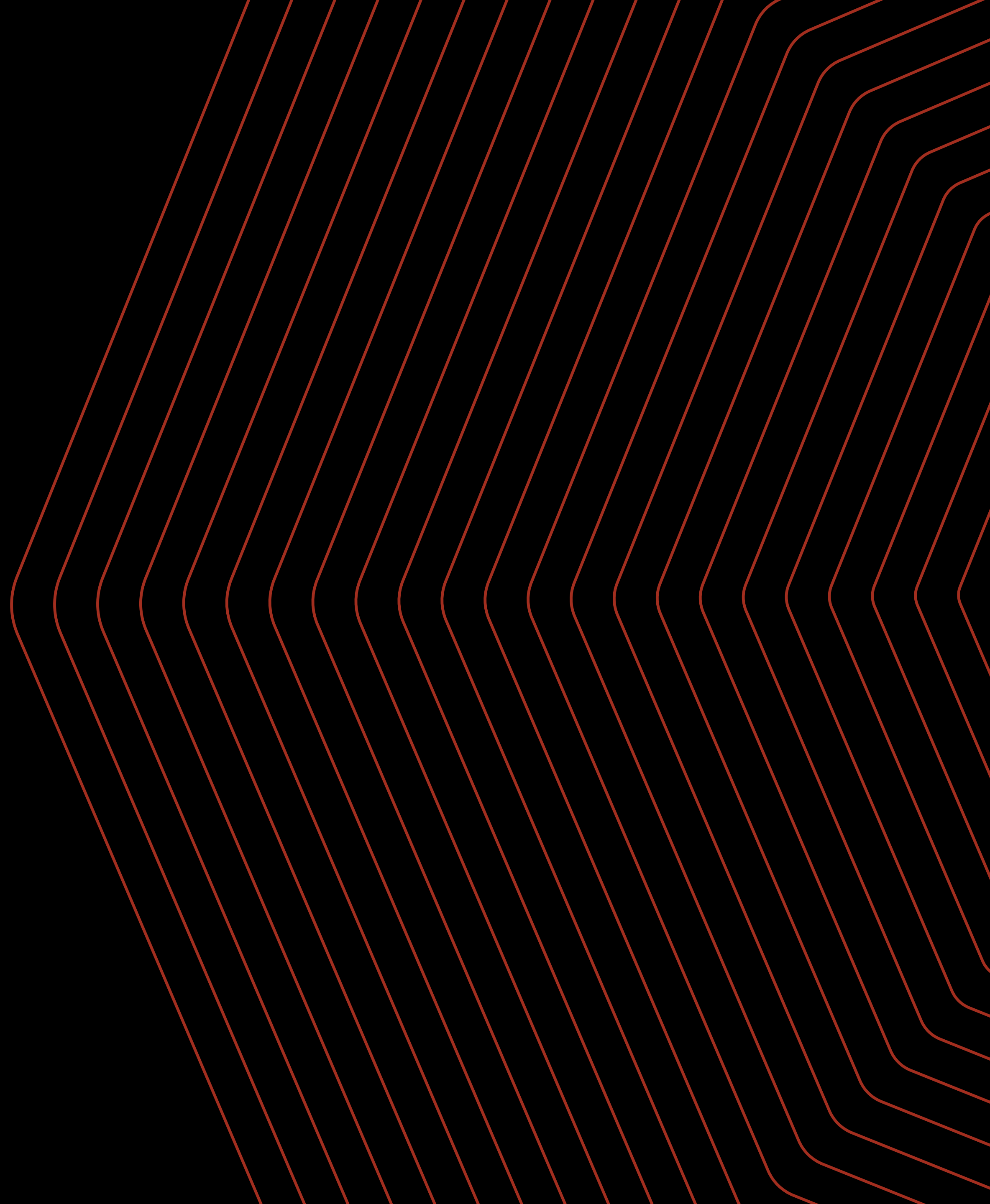
It is possible to join the network
and start mining within a few
hours



No material performance
degradation over time

Part 3

Survivability



Survivability: Introduction

- A key aspect of decentralization is the lack of dependence on developers
- Imagine you're going to buy a cryptocurrency for your child – what will you choose?
- No hardforks, chain splits, internal wars

Survivability: Fundamental approach

- Long-term survivability requires secure protocol
- Research, review, analysis and then the implementation

Well-known studies or own research:

- Chepurnoy, A., Kharin, V. and Meshkov, D., 2018. Self-reproducing Coins as Universal Turing Machine
- Chepurnoy, A., Kharin, V. and Meshkov, D., 2018. A Systematic Approach To Cryptocurrency Fees
- Meshkov, D., Chepurnoy, A. and Jansen, M., 2017. Revisiting Difficulty Control for Blockchain Systems
- Reyzin, L., Meshkov, D., Chepurnoy, A. and Ivanov, S., 2017. Improving authenticated dynamic dictionaries, with applications to cryptocurrencies
- ...



Demurrage
component

Demurrage

- Facilitated node regimes allow to survive along the blockchain growth
- Digest mode enables the verification of blocks without keeping the whole state
- But miners should create correctness proofs for light nodes, therefore they must still refrain the entire state
- How to limit the growth of the size of the state?

Demurrage: State size

- Demurrage component: miners are paid for keeping state
- If box was kept untouched in the state for 4 years, miner could collect $K \cdot \text{boxSize}$ coins from it
- If there are not enough coins in the box at this point - it is removed from the state.
- To incentivise miners to clean this dust, every box should contain at least $\text{MinValuePerByte} \cdot \text{boxSize}$ coins
- Upper-bound of the state size become predictable
- Parameters may be changed by miner votes

Demurrage: Circulating supply

- In existing cryptocurrencies, after the initial emission number of coins decreases due to lost keys, incorrect contracts, etc.
- This loss rate may exceed emission rate much earlier
- Eventually number of coins in circulation will reach 0
- By collecting storage fee from outdated boxes, miners return coins to circulation

Demurrage: Miner reward

- Mining without fixed reward is unstable
- Transaction fees are unpredictable and usually quite low
- Demurrage provides an additional miners reward
- Allows to stop emission quite soon (8 years)

Demurrage

- Makes Ergo similar to demurrage currency
- Similar to inflation, demurrage incentivizes people to use their money (at least once per 4 years)
- Real-life experiments demonstrated significant increase in the velocity of money in circulation
- However, number of real-life experiments is low
- May be turned off ($K=0$) via miners voting



Development

Development

- Environment is not static, therefore the network should also be changeable
- But how to make these changes?
- A decentralized cryptocurrency should avoid a dedicated "core" team
- But development still requires funds
- The network should achieve long-term survivability without promised further changes

Development: Voting protocol

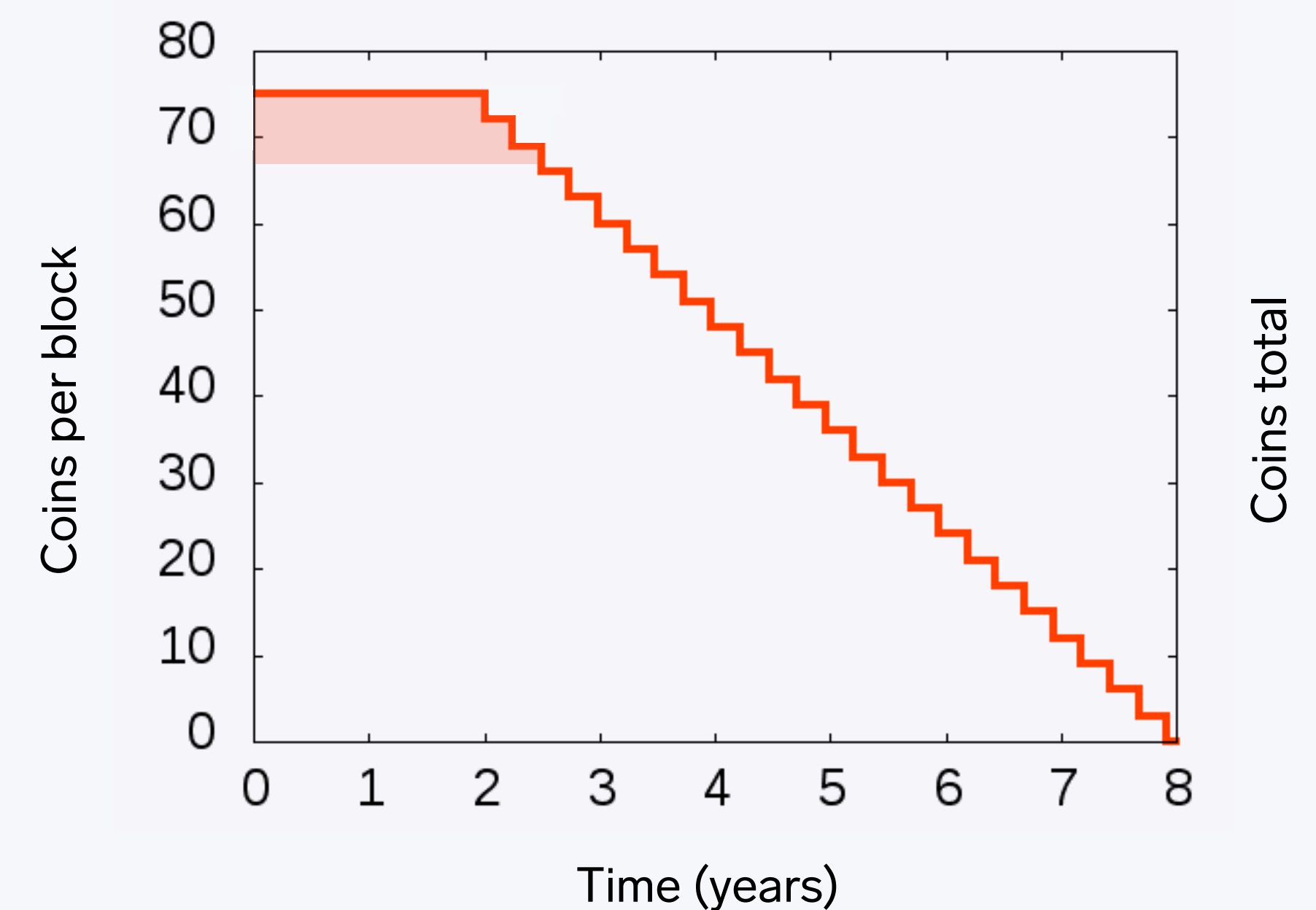
- Ergo allows to change a lot of parameters via miners voting:
K, MinValuePerByte, block size, contract costs and more...
- When a new epoch (1024 blocks) starts, miner proposes changes for up to 2 parameters
- If the majority of miners vote "yes" for proposed change at the end of the epoch, it is confirmed
- Some parameters are changed with a fixed step, the rest are limited to 1% per epoch

Development: Voting protocol

- voting on fundamental changes that require a soft-fork lasts for 32 epochs and requires 90 percent of the miners to vote
- Only one fundamental change can be voted on simultaneously
- After a successful vote a 128-epoch activation period starts
- If user sees a block with the next version, he skips validation of new features
- But this part is still under construction

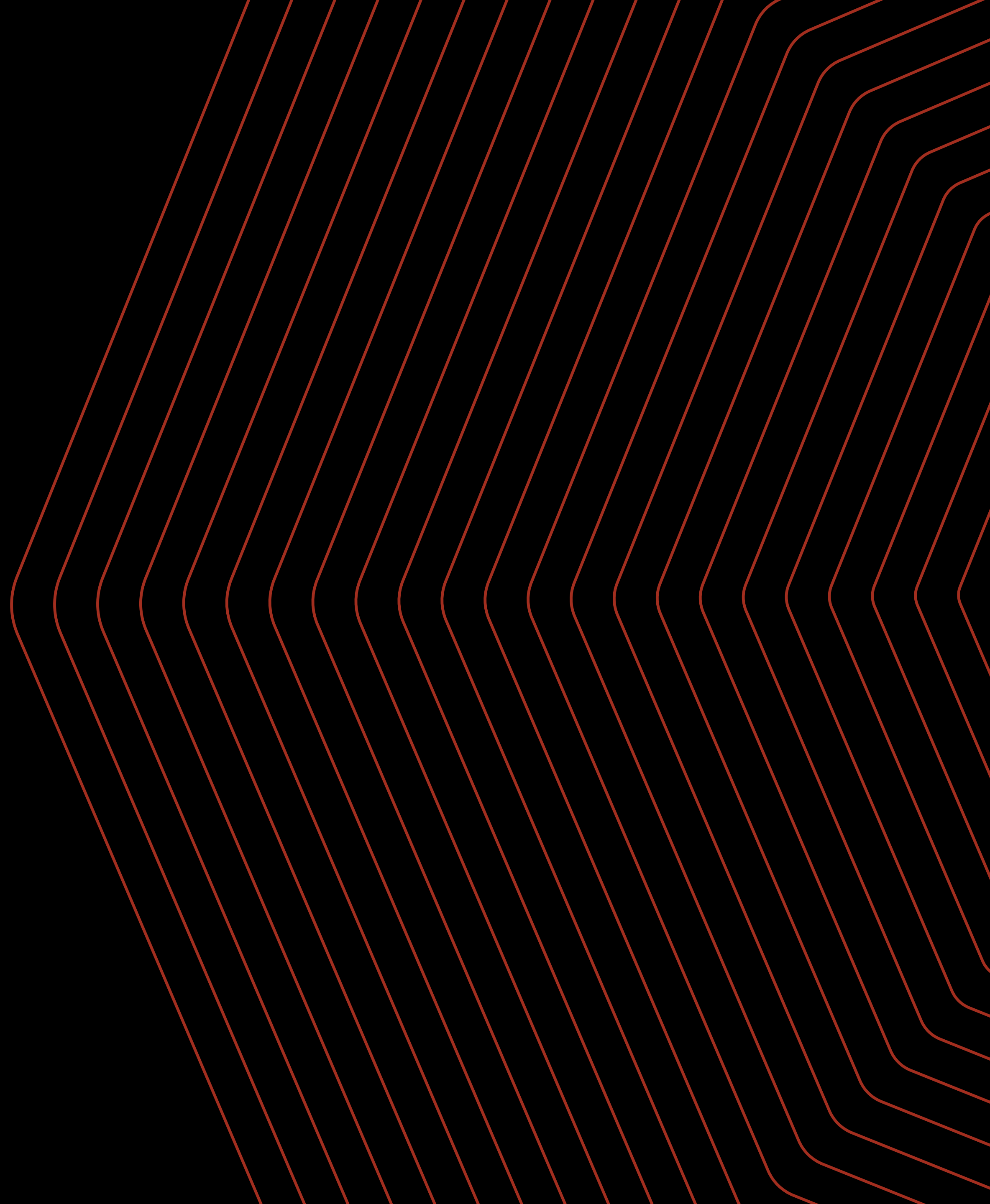
Development: Foundation

- Part of the emission goes to a treasury to fund the development
- For the first 2 years, block reward is 75 Erg, 7.5 Erg (10%) of them goes to foundation
- First year foundation reward will be used to cover EFYT token
- EFYT to Erg swap will be performed on Waves DEX by buyback in Erg/EFYT pair with limit 1
- After the first year the community will decide, where to spend these funds via voting
- But this part is still under construction



Part 4

Applicability



Applicability

- To survive, a blockchain must have a user base
- Ergo: user-friendly and protocol-friendly
- Voting - flexible on-chain configuration
- Light nodes - decentralized wallets, DApps, layer 2 solutions
- Smart contracts - secure and flexible



Smart contracts

Smart contracts: Bitcoin vs Ethereum

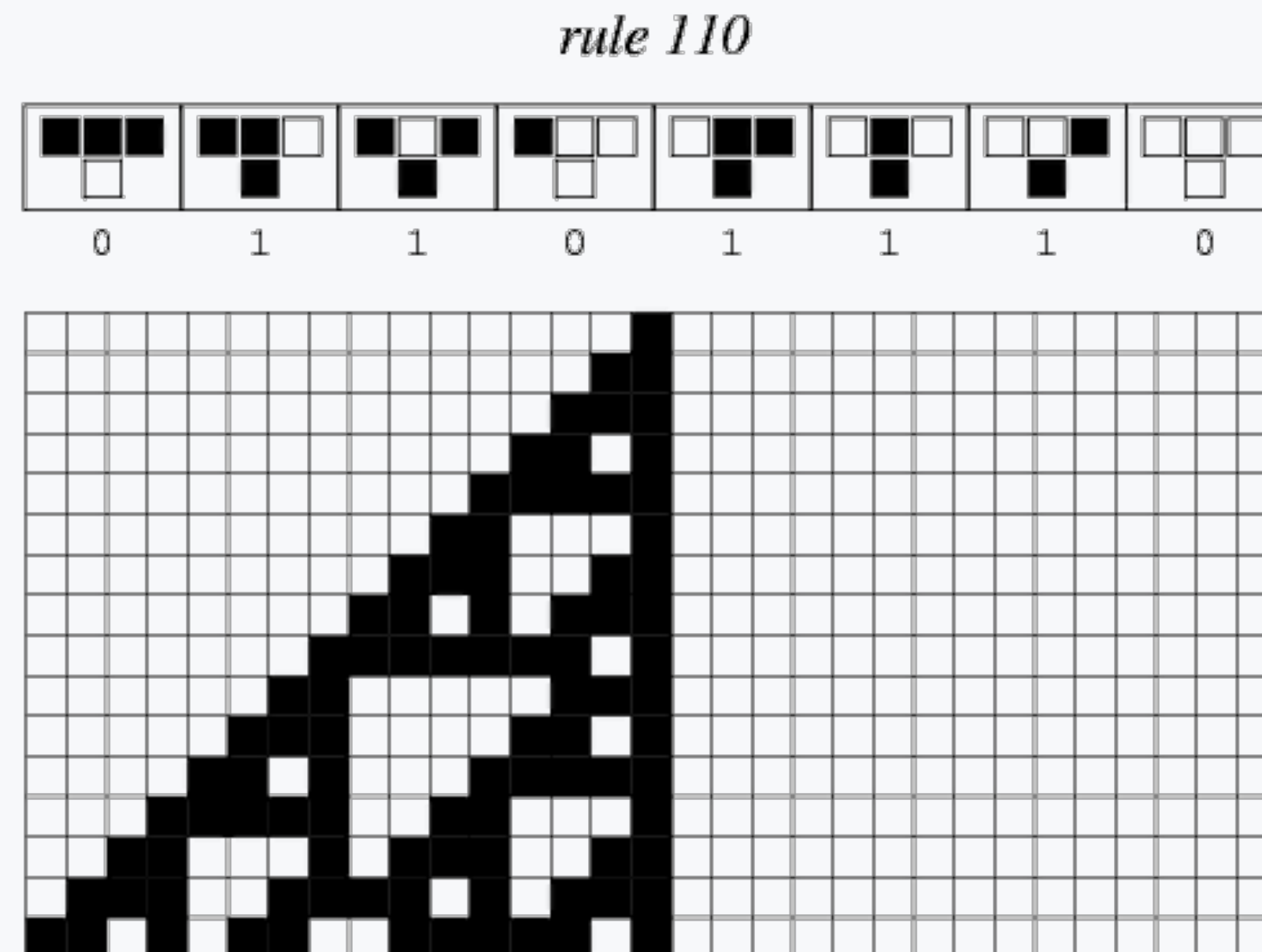
- UTXO model (output == box)
- Bitcoin-like script, that protects box from spending
- Bitcoin already allows to implement a lot of contracts: <https://en.bitcoin.it/wiki/Contract>
- Assumed to be not Turing complete
- But known Turing-complete blockchains are subject to DoS attacks
- While still have "gas" limit and don't allow infinite loops

Ergo script: **Idea**

- Strict upper-bound for computation time
- Only operations, that allow to estimate script complexity before execution
- Constant-time access to environment (few last headers)
- No jump operator and infinite loops
- Operations on predefined collections
- Supposed to be not Turing-complete...

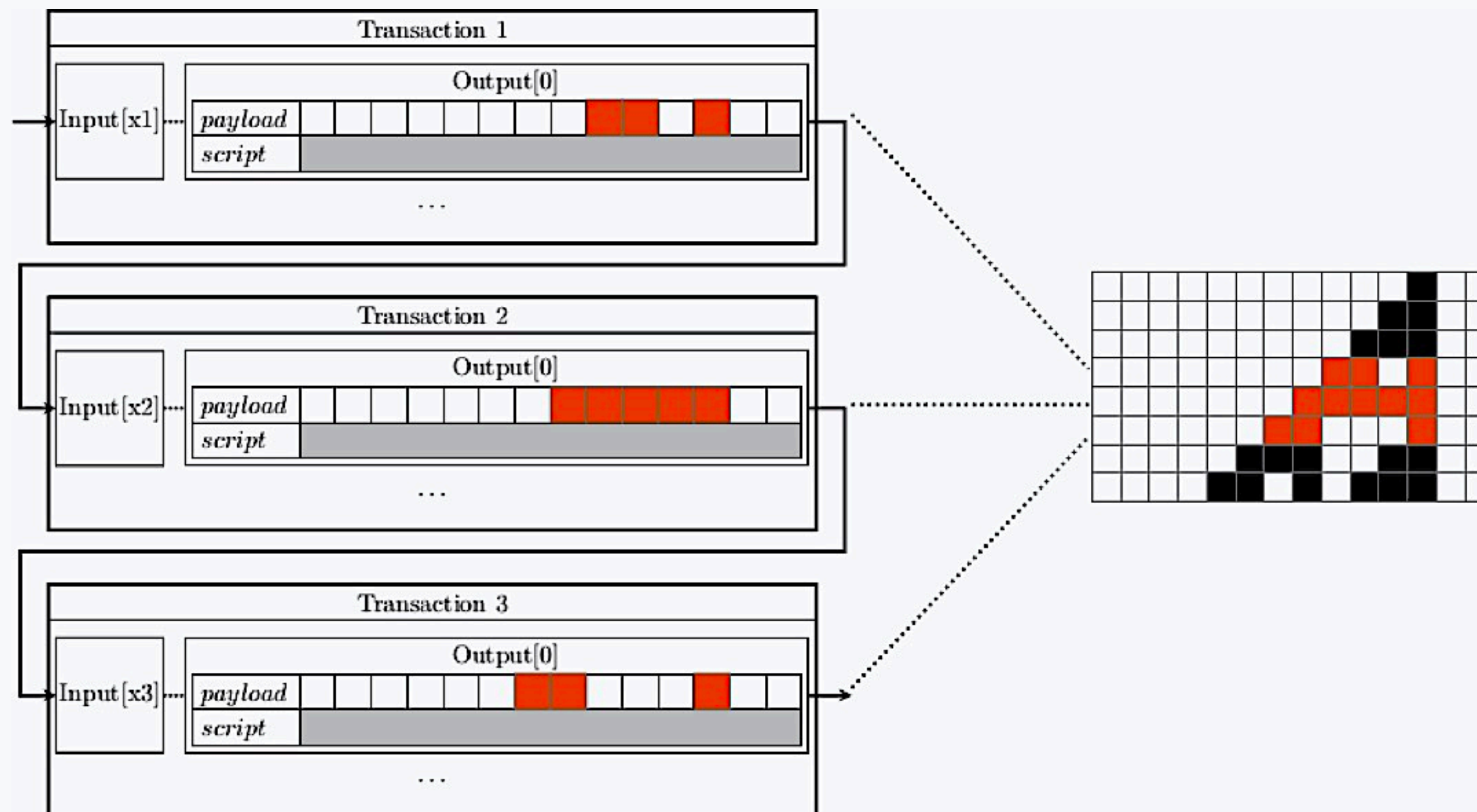
Ergo script: Chaining

- ... but actually is
- Turing completeness proof – implementation of known Turing complete system
- Rule 110 was implemented in Ergo script (see <http://arxiv.org/pdf/1806.10116v1>)



Ergo script: Chaining

Even if you don't have infinite loop inside a block, you have it between blocks



Ergo script: Chaining

- Rule 110 Ergo script implementation
- Becomes possible because of access to outputs (no such direct access in Bitcoin)

```
let indices: Array[Int] = Array(0, 1, 2, 3, 4, 5)
let inLayer: Array[Byte] = SELF.R3[Array[Byte]].value
fun procCell(i: Int): Byte = {
  let l = inLayer((if (i == 0) 5 else (i - 1)))
  let c = inLayer(i)
  let r = inLayer((i + 1) % 6)
  intToByte((l * c * r + c * r + c + r) % 2)
}
(OUTPUTS(0).R3[Array[Byte]].value == indices.map(procCell)) &&
(OUTPUTS(0).propositionBytes == SELF.propositionBytes)
```

Ergo script: Chaining

If you need some computation:

- Estimate work done before execution
- Put it to one or multiple transactions

Number of blocks is infinite



Possibly infinite loop

Ergo script: Operations

- Arithmetics: (+, -, *, /, %)
- Boolean operations (||, &&, xor)
- If-then-else
- Arrays: `append`, `slice`, `map`, `fold`, `exists`, `forall`, etc.
- Lambdas: `OUTPUTS.exists(fun (out: Box) = out.value >= minToRaise)`
- Deserialize → P2SH, MAST
- Crypto: blake2b, sha256, AVL+ tree, composable Σ -protocols

Ergo script: Emission

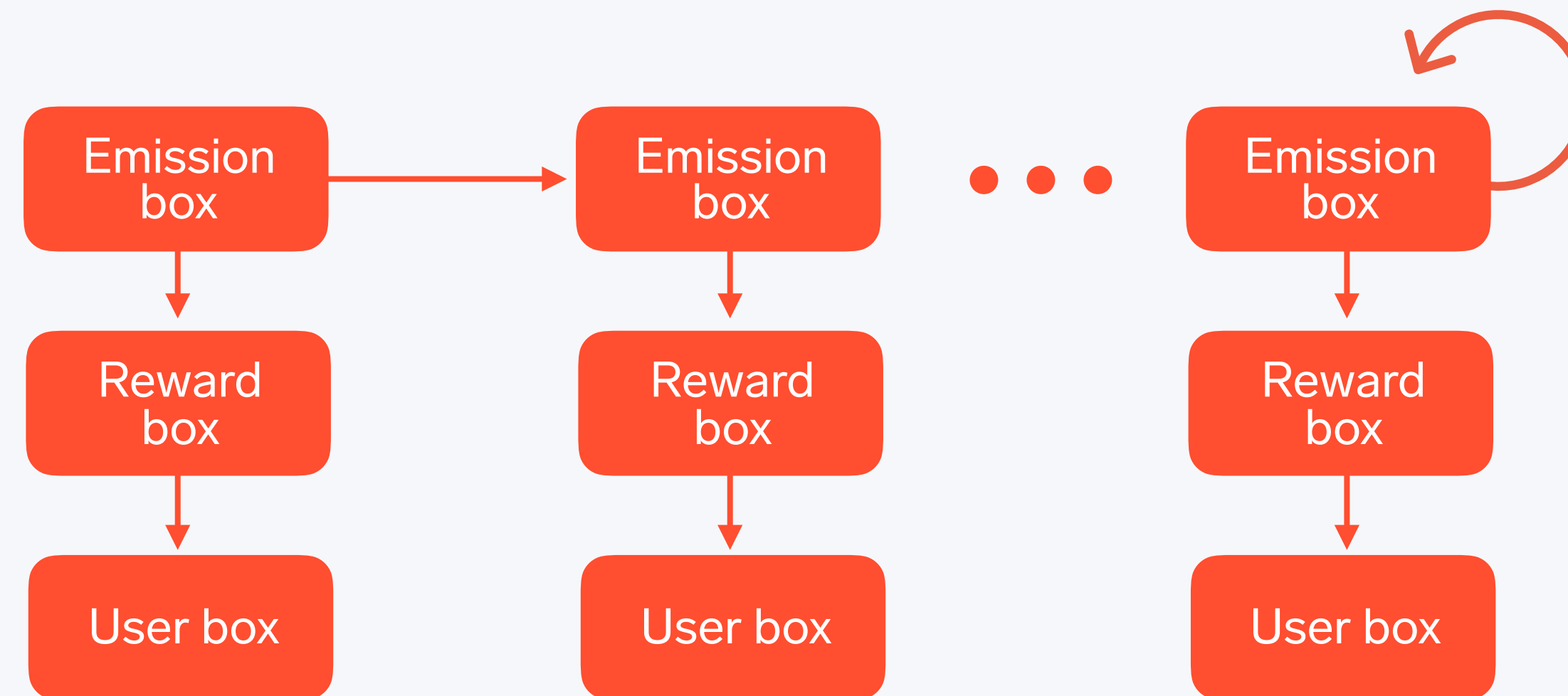
Consensus rules may be moved to scripting layer

Emission rules (testnet 1.7):

```
let epoch = 1 + ((HEIGHT - fixedRatePeriod) / epochLength)
let out = OUTPUTS(0)
let coinsToIssue = if(HEIGHT < fixedRatePeriod) fixedRate
else fixedRate - (oneEpochReduction * epoch)
let correctCoinsConsumed = coinsToIssue == (SELF.value - out.value)
let sameScriptRule = SELF.propositionBytes == out.propositionBytes
let heightIncreased = HEIGHT > SELF.R3[Long].value
let heightCorrect = out.R3[Long].value == HEIGHT
let lastCoins = SELF.value <= oneEpochReduction
(correctCoinsConsumed && heightCorrect && heightIncreased && sameScriptRule) ||
(heightIncreased && lastCoins)
```

Ergo script: Emission

- In testnet 1.8 there is also rule for miner output.
- It should be spent in such a transaction, that has exactly one output, which creation height is current height, and proposition is: $\text{Height} \geq \text{SELF.creationHeight} + 720 \ \&\& \ \text{minerPk}$

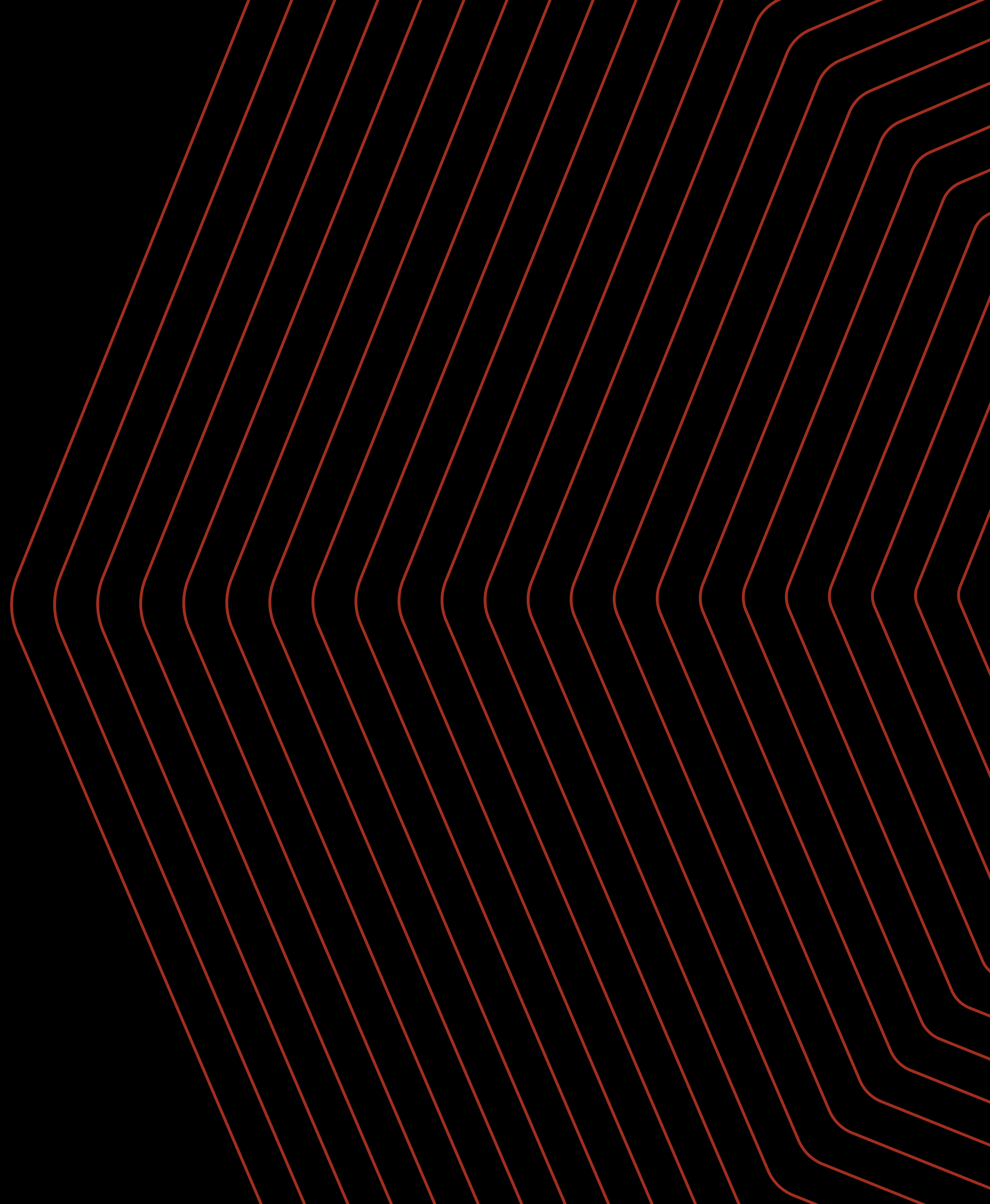


Ergo script: Examples

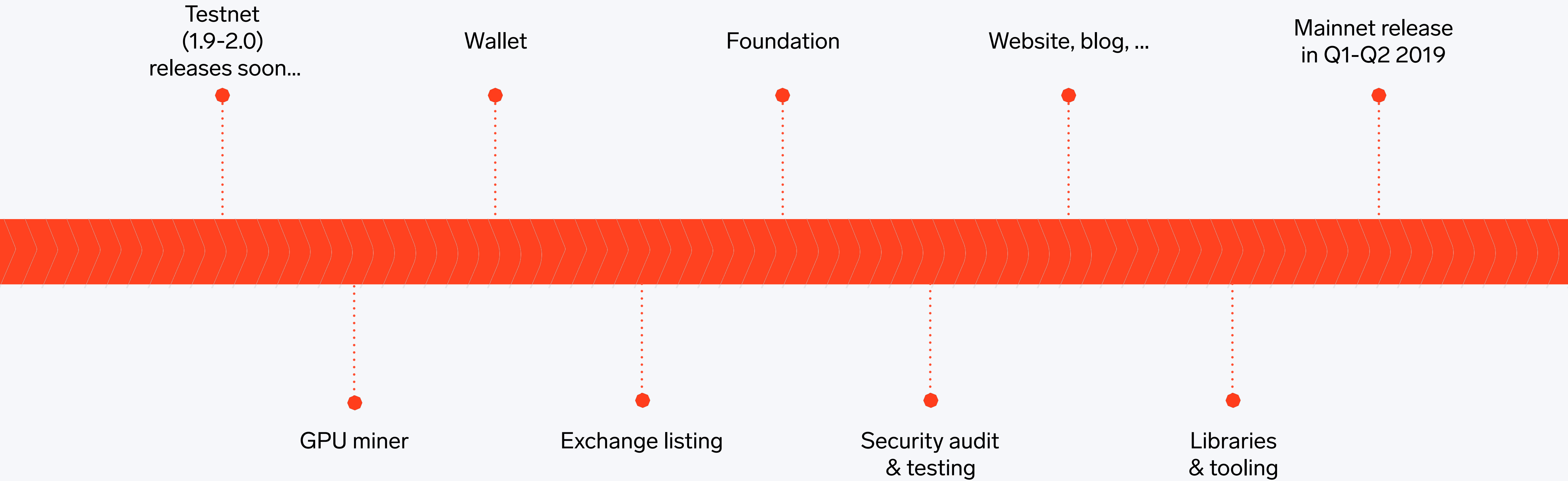
- More at <https://git.io/fpDhE>
- Atomic swaps, DEX, crowdfunding, etc..

Part 5

Roadmap



Roadmap



Thank You.
The End.



ergoplatform.org