KONTOR ECONOMIC MODEL

October 07, 2025

Contents

1 Protocol Specification

1.1 Introduction

This document outlines the formal algebraic model for the economics of the Kontor Data Persistence Protocol. It defines the core sets, state variables, parameters, and mathematical relationships that govern the behavior of participants and the overall system dynamics.

1.2 Participants

Kontor is a metaprotocol on Bitcoin, the economics of which are implemented in a core set of smart contracts that specify the KOR native token and the file persistence protocol. The participants in the KOR ecosystem are thus:

- **Users:** Accounts that pay to have their data stored on the network.
- **Storage Nodes:** Accounts that commit to store user data, stake KOR as a security deposit, and respond to storage challenges. They earn KOR rewards for their services.

1.3 Protocol Objects and State Variables

All state variables are considered at a specific block t, unless otherwise noted.

1.3.1 KOR States

KOR is held by users, storage nodes and other Kontor addresses that do not interact with the file storage protocol. KOR exists in one of the following states:

- **Liquid KOR** (b_n): Liquid KOR can be transferred, used to pay fees (e.g., voluntary leave fees), or it may be deposited into the stake. This is also referred to as "unstaked" KOR.
- **Staked KOR** (k_n): Staked KOR is locked as a security deposit to cover storage agreements. It cannot be transferred and is subject to slashing. The protocol uses a pooled stake model where each node maintains a single stake balance k_n that covers all their file agreements. There is no per-agreement stake.
- **Unvested KOR**: KOR that has been allocated but not yet released according to predetermined unlock schedules. Unvested KOR cannot be transferred, staked, or used to pay fees.

1.3.2 Storage Nodes

The set of all storage nodes at block t is denoted by $N(t)$. For an individual node $n \in N(t)$, the protocol tracks the following state:

- b_n: The spendable KOR balance of node n.
- k_n: The total amount of KOR staked by node n.
- F_n: The set of files currently being stored by node n.

1.3.3 File Agreements

A file agreement represents a contract to store a specific file. The set of all file agreements at block t is denoted by $F(t)$. Files are added to $F(t)$ once they reach $n_{\min}$ storage nodes. For an individual agreement $f \in F(t)$, the protocol tracks the following state:

- s_f^{bytes}: The size of file f in bytes.
- s_f: The number of sectors in file f, defined as $s_f \overset{\text{def}}{=} \left\lceil \frac{s_f^{\text{bytes}}}{s_{\text{sector}}^{\text{bytes}}} \right\rceil$.
- rank_f: The file's immutable creation order (1 for first file, 2 for second, etc.).
- ω_f: The file's perpetual emission weight, fixed at creation time.
- N_f: The set of storage nodes currently storing file f.

The protocol tracks the following global state relating to file agreements:

- $S_{\text{bytes}}(t)$: Total bytes stored across all file agreements, defined as $S_{\text{bytes}}(t) \overset{\text{def}}{=} \sum_{f \in F(t)} s_f^{\text{bytes}}$

- $S_{\text{sectors}}(t)$: Total sectors stored across all file agreements, defined as $S_{\text{sectors}}(t) \overset{\text{def}}{=} \sum_{f \in F(t)} s_f$

1.3.4 Sponsorship Agreements

The set of all sponsorship agreements at block t is denoted by $M(t)$. Each sponsorship agreement is a tuple:

$$m = (f, n_{\text{entrant}}, n_{\text{sponsor}}, \gamma_{\text{rate}}, \gamma_{\text{duration}}, t_{\text{start}}) \tag{1}$$

where:

- f: The file being sponsored.
- n_{entrant}: The sponsored node.
- n_{sponsor}: The sponsoring node.
- γ_{rate}: The fractional commission rate.
- γ_{duration}: The duration in blocks. The sponsorship agreement is active from block t_{start} through block $t_{\text{start}} + \gamma_{\text{duration}} - 1$ (inclusive), expiring when $t \geq t_{\text{start}} + \gamma_{\text{duration}}$.
- t_{start}: The start block of the agreement.

1.3.5 Challenges

A challenge is a deterministically generated protocol event that specifies a storage node (n) and a specific file (f) for which the node must generate a proof within a given window. This proof is a compact, non-interactive proof of retrievability (a recursive SNARK) that demonstrates knowledge of the file's data at the time of the challenge.

- C_{target}: The target number of challenges per file per year.
- s_{chal}: The number of sectors (Merkle leaves) sampled from a file during a single challenge.
- μ: The fraction of a file's sectors that a node has deleted or lost (the per-sector data loss probability).
- W_{proof}: A period of Bitcoin blocks during which the challenged node must submit a proof before the challenge expires.

For each challenge, the s_{chal} sectors are sampled pseudo-randomly using the Bitcoin block hash as a seed. This sampling is effectively independent and identically distributed (IID). In the edge case where a file has fewer sectors than the challenge requirement ($s_f < s_{\text{chal}}$), all s_f sectors are challenged.

1.3.6 Storage Proofs

A storage proof is a proof that a storage node is storing one or more files. A proof may be valid or invalid. Nodes can aggregate multiple challenges into a single proof transaction to minimize Bitcoin transaction fees.

1.4 Global State

- $\Omega(t)$: The network's total emission weight, defined as $\Omega(t) \overset{\text{def}}{=} \sum_{f \in F(t)} \omega_f$
- $\overline{R}(t)$: Weighted moving average of global emission-weighted file replication.
- $\alpha(t)$: Emission rate multiplier, which adjusts based on $\overline{R}(t)$

2 Protocol Mechanics

2.1 Protocol Functions

2.1.1 File and Network Primitives

- **File Emission Weight** (ω_f): Each file receives a perpetual emission weight based on its size and creation rank. This weight is fixed for the file's lifetime and determines its relative share of network emissions. Because the file's rank appears in the denominator, earlier files (with lower ranks) receive a higher emission weight.

$$\omega_f \overset{\text{def}}{=} \frac{\ln\left(s_f^{\text{bytes}}\right)}{\ln\left(1 + \text{rank}_f\right)} \tag{2}$$

The minimum file size $s_{\min}$ enforced by the protocol ensures $\omega_f > 0$ for all valid files.

- **Total Emission Weight** ($\Omega(t)$): The network's total emission weight is the sum of all individual file weights, with a genesis bootstrap value:

$$\Omega(t) \overset{\text{def}}{=} \begin{cases} 1.0 & \text{if } F(t) = \emptyset \\ \sum_{f \in F(t)} \omega_f & \text{otherwise} \end{cases} \tag{3}$$

  Genesis Weight: At network initialization ($t = 0$), when no files exist, the protocol initializes $\Omega(0) = 1.0$ to prevent division-by-zero errors in formulas dependent on emission weights. This genesis weight is a pragmatic implementation detail that ensures mathematical stability during network bootstrap, allowing stake calculations and emission distributions to function from the first block. The genesis weight represents a minimal baseline emission capacity that is immediately superseded once the first file is added to the network.

- **Per-Node Base Stake** (k_f): The base stake quantity is calculated at file creation time based on the network's total file count at that moment, ensuring that stake requirements grow sub-linearly with network size. This value is immutable for the lifetime of the file.

$$k_f \overset{\text{def}}{=} \left(\frac{\omega_f}{\Omega(t_{\text{create}})}\right) \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t_{\text{create}})|}{F_{\text{scale}}}\right) \tag{4}$$

  where c_{stake} is the base capital parameter (in KOR), F_{scale} is a scaling factor that controls how quickly stake requirements grow with network size (higher values slow the growth), and t_{create} is the block at which file f was created.

2.1.2 Monetary Policy and Emissions

- **Emission-Weighted Average Replication** ($\overline{R}(t)$): The protocol tracks network health through a weighted moving average of emission-weighted replication. The instantaneous weighted replication is calculated:

$$\overline{R}_{\text{inst}}(t) \overset{\text{def}}{=} \frac{\sum_{f \in F(t)} \left(\omega_f \cdot |N_f|\right)}{\Omega(t)} \tag{5}$$

  This is then smoothed using a weighted moving average over a trailing W_R-block window:

$$\overline{R}(t) \overset{\text{def}}{=} \frac{\sum_{i=0}^{W_R - 1} \left[(W_R - i) \cdot \overline{R}_{\text{inst}}(t - i)\right]}{W_R \cdot \frac{W_R + 1}{2}} \tag{6}$$

  where more recent observations receive linearly higher weights.

- **Emission Rate Multiplier** ($\alpha(t)$): The protocol uses a responsive control system to maintain network health based on the health buffer relative to the minimum replication threshold $n_{\min}$.

  Let the health "buffer" be $m(t) \overset{\text{def}}{=} \overline{R}(t) - n_{\min}$. The emission rate multiplier $\alpha(t)$ is defined as in 7:

$$\alpha(t) \overset{\text{def}}{=} \begin{cases} \min\left(\alpha_{\max}, 1 + \frac{\kappa_\alpha}{m(t)}\right) & \text{if } m(t) > 0 \\ \alpha_{\max} & \text{if } m(t) \leq 0 \end{cases} \tag{7}$$

  This function has two regimes:

    - When the network is healthy ($m(t) > 0$), the emission rate multiplier is 1 plus an inverse function of the buffer. As the buffer shrinks, the multiplier increases rapidly. The result is capped at $\alpha_{\max}$.
    - When the network is in a critical state ($m(t) \leq 0$), the emission rate multiplier is immediately set to $\alpha_{\max}$ to create the strongest possible incentive for repair.

- **Total KOR Emissions** ($\varepsilon(t)$): Total emissions are calculated based on the dynamic inflation rate and the current total supply. The per-block emissions are derived from the annualized rate adjusted by the dynamic multiplier from the previous block.

$$\varepsilon(t) \overset{\text{def}}{=} \mathrm{KOR}_{\text{total}}(t-1) \cdot \frac{\alpha(t-1) \cdot \mu_0}{B} \tag{8}$$

  This ensures emissions scale with the total supply, maintaining the target inflation rate. The use of $\alpha(t-1)$ ensures the emission rate is known at the start of block t.

- **File-Specific KOR Emissions** ($\varepsilon_{f(t)}$): Each file's emissions are its proportional share of the total emissions for the block, based on its emission weight.

$$\varepsilon_{f(t)} \overset{\text{def}}{=} \varepsilon(t) \cdot \left(\frac{\omega_f}{\Omega(t)}\right) \tag{9}$$

2.1.3 Rewards and Fees

- **User Storage Fee** (v_f): To create a storage agreement, the user pays a one-time fee that is a fraction of the per-node base stake requirement at the time of creation. This entire fee is burned.

$$v_f \overset{\text{def}}{=} \chi_{\text{fee}} \cdot k_f \tag{10}$$

- **Leave Fee** ($\varphi_{\text{leave}}(f,t)$): A node exiting an agreement for file f pays a fee that scales with file vulnerability, increasing quadratically as replication approaches the minimum.

$$\varphi_{\text{leave}}(f,t) \overset{\text{def}}{=} k_f \cdot \left(\frac{n_{\min}}{|N_f|}\right)^2 \tag{11}$$

  where $|N_f|$ is the current replication count before the node's departure. Since leaving is only permitted when $|N_f| > n_{\min}$, the fee is always less than the base stake requirement k_f. The fee is paid from the spendable balance b_n for voluntary departures, or from the stake balance k_n for involuntary departures during stake insufficiency handling. All paid leave fees are burned.

- **Dynamic Stake Factor** (λ_{stake}): The stake amplification factor provides a safety margin against slashing and deters Sybil attacks by imposing a capital premium on nodes with few files. The logarithmic scaling ensures that nodes with very few files face higher capital requirements per file, while large nodes approach the baseline multiplier of 1.

$$\lambda_{\text{stake}}(n) \overset{\text{def}}{=} 1 + \frac{\lambda_{\text{slash}}}{\ln(2 + |F_n|)} \tag{12}$$

  The offset of 2 in the logarithm ensures that even single-file nodes ($|F_n| = 1$) have finite stake requirements.

- **Total Required Stake** (k_{req}): The total stake a node must hold is the sum of its per-file stake requirements, amplified by the dynamic stake factor.

$$k_{\text{req}}(n,t) \overset{\text{def}}{=} \left(\sum_{f \in F_n} k_f\right) \cdot \lambda_{\text{stake}}(n) \tag{13}$$

- **Node Storage Reward** (r_{storage}): The emissions generated by a file are distributed among the nodes storing it, with adjustments for sponsorship agreements. The base per-node reward for holding file f is $r_{\text{n}|\text{f}}(t) \overset{\text{def}}{=} \frac{\varepsilon_{f(t)}}{|N_f|}$.

  The total reward for a node n storing file f is:

$$r_{\text{storage}}(n, f, t) \overset{\text{def}}{=} r_{\text{n}|\text{f}}(t) \cdot \left(1 - \gamma_{\text{paid}}(n, f, t) + \gamma_{\text{earned}}(n, f, t)\right) \tag{14}$$

  where:
  - $\gamma_{\text{paid}}(n, f, t) \overset{\text{def}}{=} \begin{cases} \gamma_{\text{rate}} & \text{if } \exists m \in M(t) \text{ s.t. } m = (f, n, n_{\text{sponsor}}, \gamma_{\text{rate}}, \gamma_{\text{duration}}, t_{\text{start}}) \\ 0 & \text{otherwise} \end{cases}$

    This is the commission rate node n pays as an entrant for file f in a sponsorship agreement m.
  - $\gamma_{\text{earned}}(n, f, t) \overset{\text{def}}{=} \sum_{m \in M(n,f,t)} \gamma_{\text{rate}}$

    This is the sum of all commission rates earned by node n for sponsoring other nodes for file f. $M(n, f, t)$ is the set of agreements where n is the sponsor for file f.

- **Expected Slashing Revenue** ($E[r_{\text{slash}}]$): When a node storing a file fails a challenge, their slashed stake is partially redistributed to remaining nodes. Each node has probability $P_{\text{chal}}\left(|N_f|, t\right) = \frac{p_f}{|N_f|}$ of being challenged and probability p_{fail} of failing if challenged. When a node is slashed, the redistributed amount $(1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}$ is split equally among the $\left(|N_f| - 1\right)$ remaining nodes.

  The expected revenue for node n is the sum over all other nodes $j \in N_f, j \neq n$. Since each other node has the same probability of being challenged and failing, and there are $\left(|N_f| - 1\right)$ such nodes:

$$\begin{aligned} E[r_{\text{slash}}(n, f, t)] &= \sum_{j \in N_f, j \neq n} P_{\text{chal}}\left(|N_f|, t\right) \cdot p_{\text{fail}} \cdot \frac{(1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}}{|N_f| - 1} \\ &= \left(|N_f| - 1\right) \cdot \left(\frac{p_f}{|N_f|}\right) \cdot p_{\text{fail}} \cdot \frac{(1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}}{|N_f| - 1} \end{aligned} \tag{15}$$

  This simplifies to:

$$E[r_{\text{slash}}(n, f, t)] \overset{\text{def}}{=} \frac{p_f \cdot p_{\text{fail}} \cdot (1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}}{|N_f|} \tag{16}$$

2.2 Challenge Frequency

The protocol targets a constant challenge-rate per file per year, ensuring predictable security regardless of network scale.

- **Files Challenged per Block** ($\theta(t)$): The number of files challenged network-wide per block scales linearly with the total number of files.

$$\theta(t) \overset{\text{def}}{=} \frac{C_{\text{target}} \cdot |F(t)|}{B} \tag{17}$$

- **Per-File Challenge Probability** (p_f): From this, the probability that any specific file f is selected for a challenge in a given block is constant.

$$p_f \overset{\text{def}}{=} \frac{\theta(t)}{|F(t)|} = \frac{C_{\text{target}}}{B} \tag{18}$$

- **Per-Node Challenge Probability** (P_{chal}): For a file f, one of its N_f storing nodes is chosen randomly. The probability a specific node is challenged for a specific file is:

$$P_{\text{chal}}\left(|N_f|, t\right) \overset{\text{def}}{=} \frac{p_f}{|N_f|} \tag{19}$$

2.3 State Transitions

- **Storage Agreement Creation**:
  - User pays fee v_f which is burned.
  - The file is added to $F(t)$ once at least n_{min} nodes have joined to store it.
  - Once in $F(t)$, the file remains permanently and receives emissions every block.
- **Storage Node Joining**:
  - For a node to join an agreement, the node must have sufficient stake: $k_n \geq k_{\text{req}}(n, F_n \cup \{f\})$, where $k_{\text{req}}(n, F_n \cup \{f\})$ is the required stake calculated as if node n were already storing file f.
    - A node can join an existing agreement for file f in one of two ways:
      - **Unsponsored Join:** A node that can acquire the file data through off-chain means (e.g., from the original user) can join directly. It must have sufficient stake: $k_n \geq k_{\text{req}}(n, F_n \cup \{f\})$. The sets F_n and N_f are updated. The node will earn the full base reward.
      - **Sponsored Join:** A node requiring the file data from an existing storer (an "Entrant") can enter into a sponsorship agreement with one of the existing nodes in N_f (a "Sponsor"). This creates a new sponsorship contract $m = (f, \text{entrant}, \text{sponsor}, \gamma_{\text{rate}}, \gamma_{\text{duration}})$ which is added to the global set M. The γ_{rate} and γ_{duration} are negotiated off-chain. The Entrant's rewards are reduced by γ_{rate} and the Sponsor's are increased by the same amount for the duration of the contract.
- **Voluntary Storage Node Leaving**:
  - A storage node may leave an agreement voluntarily only when $|N_f| > n_{\text{min}}$ and $b_n \geq \varphi_{\text{leave}}(f, t)$. The node pays a leave fee from its spendable balance: b_n decreases by $\varphi_{\text{leave}}(f, t)$. The leave fee is burned.
  - The membership sets F_n and N_f are updated. Any sponsorship contracts where the node was an entrant or sponsor are voided.
- **Stake Withdrawal**:
  - Withdrawals are programmatically blocked if they would result in the node's stake k_n falling below its required stake $k_{\text{req}}(n)$.
- **Stake Insufficiency Handling:** If a node's stake falls below its requirement (e.g., after being slashed), this automated process restores sufficiency by removing it from agreements in a pseudo-random order. This operation takes precedence over withdrawal.
  - **Pass 1 (Graceful Exit):** The node is removed from agreements where its departure is non-critical ($|N_f| > n_{\text{min}}$). This continues until sufficiency is met. For each involuntary exit from file f, the protocol deducts a penalty from the node's staked balance k_n:
    - Of this penalty $k_f \cdot \lambda_{\text{slash}}$, an amount $\beta_{\text{slash}} \cdot k_f \cdot \lambda_{\text{slash}}$ is burned.
    - The remainder $(1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}$ is distributed equally among the other storers in N_f.
  - **Pass 2 (Total Forfeiture):** If Pass 1 is insufficient, the node's entire remaining stake k_n is burned, and the node is removed from all remaining agreements. This ensures the node pays the maximum possible penalty and cannot game the insufficiency mechanism.
- **Challenge Generation**:
  - With each block, a file f is selected for challenge with probability p_f. One node ($n_f \in N_f$) is randomly selected with probability $\frac{1}{|N_f|}$.
- **Proof Submission**:
  - If n_f is challenged, the node may generate a proof and pay $c_{\text{proof}}^{\text{BTC}}(t)$ in Bitcoin for its broadcast.

- **Proof Validation**: If proof is valid, there is no penalty. Nodes can aggregate multiple challenges into a single proof transaction within the W_{proof} window to minimize fees.
  - If the proof is not submitted within W_{proof} blocks, the node fails the challenge and faces slashing.
  - If the proof is invalid, the node fails the challenge and faces slashing.
- **Slashing**:
  - When a node is slashed, its stake k_n is reduced by $k_f \cdot \lambda_{\text{slash}}$.
    - Of this amount, $\beta_{\text{slash}} \cdot k_f \cdot \lambda_{\text{slash}}$ is burned.
    - The remainder $(1 - \beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}$ is distributed equally among the other storers in N_f.
    - The slashed node n is immediately removed from N_f and f is removed from F_n.
    - Any sponsorship contracts where node n was either an entrant or a sponsor are immediately voided and removed from M.
  - If $k_n < k_{\text{req}}(n)$ after slashing, the Stake Insufficiency Handling algorithm is automatically triggered.
- **Rewards Emissions**:
  - For each file agreement f and each node $n \in N_f$, the node's balance b_n increases by $r_{\text{storage}}(n, f, t)$.
- **Update Dynamic Protocol Variables**:
  - Remove any expired sponsorship contracts from $M(t)$ (i.e., all agreements where $t \geq t_{\text{start}} + \gamma_{\text{duration}}$).
  - Calculate instantaneous replication $\overline{R}_{\text{inst}}(t)$ based on the current state after all changes in block t
  - Update weighted moving average: $\overline{R}(t)$ using the past W_R blocks of $\overline{R}_{\text{inst}}$ values
  - Update $\alpha(t)$ based on the health buffer $m(t) = \overline{R}(t) - n_{\min}$ (this value will be used for emissions in block $t + 1$)

3 System Invariants

1. **Conservation of KOR:** The total KOR in the system changes according to dynamic emissions and burns. At any block t:

$$\sum_{n \in N} (b_n + k_n) + \text{KOR}_{\text{other}}(t) = \text{KOR}_{\text{total}}(t) \tag{20}$$

   where:
   - $\sum_{n \in N} (b_n + k_n)$ represents all KOR held by storage nodes (balances + stakes)
   - $\text{KOR}_{\text{other}}(t)$ represents KOR held by users, protocol treasury, exchanges, and other entities outside the storage node ecosystem
   - $\text{KOR}_{\text{total}}(t)$ represents the net circulating supply at block t, which evolves according to:

$$\text{KOR}_{\text{total}}(t) = \text{KOR}_{\text{initial}} + \sum_{i=1}^{t} (\varepsilon(i) - \Phi_{\text{burned}}(i)) \tag{21}$$

   Model Scope: This economic model focuses primarily on the storage node ecosystem dynamics. While $\text{KOR}_{\text{other}}(t)$ exists and can be significant, the protocol mechanics described herein assume that KOR flows between storage nodes and other holders do not fundamentally alter the incentive structures within the storage network.

   Let:
   - $F_{\text{created}}(t)$ be the set of files created in block t
   - $S(t)$ be the set of slashing events (failed challenges) in block t, where each event $s \in S(t)$ involves a node storing file f_s

- $L(t)$ be the set of leave events in block t, where each event $l \in L(t)$ involves a node leaving file f_l

The burns in block t are:

$$\Phi_{\text{burned}}(t) = \sum_{f \in F_{\text{created}}(t)} v_f + \sum_{s \in S(t)} \beta_{\text{slash}} \cdot k_{f_s} \cdot \lambda_{\text{slash}} + \sum_{l \in L(t)} \varphi_{\text{leave}}(f_l, t) \tag{22}$$

The cumulative burns through block t are:

$$\Phi_{\text{burned}}^{\text{total}}(t) = \sum_{i=0}^{t} \Phi_{\text{burned}}(i) \tag{23}$$

where $\Phi_{\text{burned}}(0) = 0$ at genesis.

2. **Non-negative Balances:** For all nodes $n \in N$:

$$b_n \geq 0 \tag{24}$$

If any operation would result in $b_n < 0$, the operation fails.

3. **Bidirectional Consistency:** For all nodes $n \in N$ and files $f \in F$:

$$f \in F_n \Leftrightarrow n \in N_f \tag{25}$$

4. **Positive Parameters:** All protocol parameters must be positive where specified:

$$\mu_0 > 0 \tag{26}$$

$$c_{\text{stake}} > 0 \tag{27}$$

$$\alpha_{\max} \geq 1 \tag{28}$$

$$0 \leq \rho \tag{29}$$

$$0 \leq p_{\text{fail}} < 1 \tag{30}$$

$$n_{\min} \geq 1 \tag{31}$$

$$\lambda_{\text{slash}} > 0 \tag{32}$$

5. **Stake Sufficiency:** All nodes must maintain a total stake greater than or equal to their total required stake.

$$\forall n \in N : k_n \geq k_{\text{req}}(n) \tag{33}$$

The protocol programmatically prevents withdrawals that would violate this invariant. If a node's stake becomes insufficient (e.g., after the node is slashed), the automated removal algorithm is triggered to restore sufficiency.

6. **Staked KOR Bound:** The total staked KOR across all nodes is always less than or equal to the total KOR supply:

$$\sum_{n \in N} k_n \leq \text{KOR}_{\text{total}}(t) \tag{34}$$

This invariant ensures that stake requirements cannot exceed the available KOR supply.

4 Economic Analysis

4.1 Storage Node Economics

4.1.1 Modeling Parameters

These parameters are used for economic analysis and node decision-making, and may vary based on market conditions or individual node strategies.

- p_{fail}: Probability of an honest storage node failing a challenge (due to operational issues).
- ρ: Opportunity cost rate (discount rate) in USD terms per block (per-block fractional interest rate). Annualized rate: $r_{\text{annual}} \approx (1+\rho)^B - 1$. Conversely $\rho \approx (1 + r_{\text{annual}})^{\frac{1}{B}} - 1$. For example, a 20% annual rate corresponds to $\rho = (1.20)^{\frac{1}{52560}} - 1 \approx 0.0000034$ per block.
- h: Number of future blocks considered by nodes for profit evaluation (NPV calculation). Typical values range from 2,016 blocks (two weeks) to 26,280 blocks (six months), depending on the node operator's planning horizon and risk tolerance.
- π_{min}: Minimum net expected profit per block in USD that a node requires to stay in an agreement.
- $\xi_{\text{KOR/USD}}$: Exchange rate from KOR to USD.
- $\xi_{\text{BTC/USD}}$: Exchange rate from BTC to USD.
- $\varphi_{\text{BTC}}^{\text{rate}}(t)$: Bitcoin transaction fee rate in BTC per vByte at block t. This is a market-determined value that fluctuates based on Bitcoin network congestion.

4.1.2 Economic Functions

- **Net Present Value calculation for time-varying cashflows:**

$$\text{NPV}(\{c_i\}, \rho, h) \overset{\text{def}}{=} \sum_{i=1}^{h} \left(\frac{c_i}{(1+\rho)^i} \right) \tag{35}$$

  where $\{c_i\}$ represents the sequence of expected cashflows in USD for blocks $i = 1, 2, ..., h$.

4.1.3 Cost Functions

- **Expected Costs for node n for file f at time t (all in USD):**

  This model focuses on the primary on-chain economic costs. It does not formally include secondary operational costs such as hardware amortization, bandwidth for data repair and propagation, or manual labor, which must also be considered by node operators.

  - Expected proving cost: With aggregated proofs, a node pays once per block if challenged on any files: Let $P_{\text{any}}(n,t)$ be the probability that node n is challenged on at least one file. Let $c_{\text{proof}}^{\text{BTC}}(t) \overset{\text{def}}{=} s_{\text{proof}} \cdot \varphi_{\text{BTC}}^{\text{rate}}(t)$ be the cost of submitting an aggregated proof in BTC.

$$E\left[c_{\text{prove}}^{\text{USD}}(n,t)\right] \overset{\text{def}}{=} P_{\text{any}}(n,t) \cdot c_{\text{proof}}^{\text{BTC}}(t) \cdot \xi_{\text{BTC/USD}} \tag{36}$$

    where $P_{\text{any}}(n,t) \overset{\text{def}}{=} 1 - \Pi_{f \in F_n} \left(1 - \left(\frac{p_f}{|N_f|}\right)\right)$. Note that this assumes independent challenge selection across files, which is pseudo-random. Assuming equal replication $|N_f| = n$ for all f, this simplifies to $1 - \left(1 - \frac{p_f}{n}\right)^{|F_n|}$.

  - Expected slashing cost:

$$E\left[c_{\text{slash}}^{\text{USD}}(f,t)\right] \overset{\text{def}}{=} P_{\text{chal}}(|N_f|, t) \cdot p_{\text{fail}} \cdot k_f \cdot \lambda_{\text{slash}} \cdot \xi_{\text{KOR/USD}} \tag{37}$$

    Note: Slashing occurs when a node is challenged but chooses not to respond with a proof, or when the proof is invalid, or when the proof is not included on-chain within the W_{proof} block window.

  - Physical storage cost:

$$c^{\text{USD}}_{\text{storage}}(f,t) \overset{\text{def}}{=} s^{\text{bytes}}_f \cdot c^{\text{USD}}_{\text{byte-block}} \tag{38}$$

where $c^{\text{USD}}_{\text{byte-block}}$ is the marginal cost of storing one byte for one block. While typically small, this cost scales linearly with file size, in contrast to the logarithmic scaling of rewards.

4.1.4 Profit Functions

- **Net Expected Profit for node n at time t (USD):** The total profit for a node is the sum of profits from all files minus the aggregated proving cost and the opportunity cost on the total required stake:

$$\begin{aligned} E[\pi(n,t)] \overset{\text{def}}{=} \sum_{f\in F_n} \Big[\big(E\big[r_{\text{storage}}(n,f,t)\big] + E[r_{\text{slash}}(n,f,t)]\big) \cdot \xi_{\text{KOR/USD}} \\ -E\big[c^{\text{USD}}_{\text{slash}}(f,t)\big] - c^{\text{USD}}_{\text{storage}}(f,t)\Big] - E\big[c^{\text{USD}}_{\text{prove}}(n,t)\big] - k_{\text{req}}(n) \cdot \xi_{\text{KOR/USD}} \cdot \rho \end{aligned} \tag{39}$$

  The per-file profit (excluding the shared proving cost and total stake opportunity cost) is:

$$\begin{aligned} E[\pi(n,f,t)] \overset{\text{def}}{=} \big(E\big[r_{\text{storage}}(n,f,t)\big] + E[r_{\text{slash}}(n,f,t)]\big) \cdot \xi_{\text{KOR/USD}} \\ -E\big[c^{\text{USD}}_{\text{slash}}(f,t)\big] - c^{\text{USD}}_{\text{storage}}(f,t) \end{aligned} \tag{40}$$

  This represents the direct profit from a single file agreement before accounting for costs shared across the node's entire portfolio (aggregated proving and total stake opportunity cost).

4.2 Node Decision Framework

Storage nodes make decisions to maximize their expected utility (measured in USD profit).

- **Decision to Join File f:** A node n not currently storing f considers joining if the marginal impact on its total profit is positive. Let $E[\pi(n, t \mid F_n)]$ denote the node's total expected profit given its current file set F_n. The node joins if:

$$\text{NPV}(\{E[\pi(n,t+i \mid F_n \cup \{f\})] - E[\pi(n,t+i \mid F_n)]\}, \rho, h) > 0 \tag{41}$$

  This marginal profit calculation captures:
  - The additional storage rewards from file f
  - The change in total stake opportunity cost due to increased $k_{\text{req}}(n)$ and potential change in $\lambda_{\text{stake}}(n)$
  - The change in expected proving costs due to increased $P_{\text{any}}(n,t)$
  - The physical storage cost for file f

  For this decision, the node must forecast future values of:
  - Exchange rates ($\xi_{\text{KOR/USD}}$, $\xi_{\text{BTC/USD}}$)
  - Network replication levels ($|N_f|$)
  - Bitcoin fee rates ($\varphi^{\text{rate}}_{\text{BTC}}$)
  - Network size and emission parameters

  The analysis assumes nodes make rational forecasts based on current trends and market conditions. The participant count is assumed to be $|N_f| + 1$ (including the joining node).

- **Decision to Sponsor Node for File f:** An existing node n in N_f decides whether to sponsor an Entrant by providing the file data in exchange for a commission rate γ_{rate} for a duration γ_{duration}. The Sponsor's one-time cost is the bandwidth to transfer the data. The benefit is the NPV of the commission stream. The Sponsor will agree if:

$$\text{NPV}\left(\left\{\gamma_{\text{rate}} \cdot \left(\frac{\varepsilon_{f(t+i)}}{|N_f|+1}\right)\right\}, \rho, \gamma_{\text{duration}}\right) > c^{\text{USD}}_{\text{transfer}} \tag{42}$$

This creates a competitive market for sponsorship. A cartel of existing nodes attempting to block entry (gatekeep) will be broken by the incentive for any single member to defect and capture the entire sponsorship contract for themselves.

- **Decision to Leave File f:** A node n currently storing f evaluates whether its total profit would improve by leaving. The node compares:
  - The immediate cost: leave fee $\varphi_{\text{leave}}(f,t) \cdot \xi_{\text{KOR/USD}}$
  - The benefit: NPV of the improvement in total profit from leaving

  The node leaves if:

$$\text{NPV}(\{E[\pi(n,t+i \mid F_n \setminus \{f\})] - E[\pi(n,t+i \mid F_n)]\}, \rho, h) > -\varphi_{\text{leave}}(f,t) \cdot \xi_{\text{KOR/USD}} \tag{43}$$

  Note that the left side is typically negative (leaving reduces profit), so this condition checks if the profit reduction is less than the leave fee. The node may also be subject to a minimum profitability constraint $\pi_{\min}$, leaving if its total profit falls below this threshold and the above condition is met. Equivalently, with an explicit per-block profit floor $\pi_{\min}$ (USD/block), a node leaves when $E[\pi(n,t)] < \pi_{\min}$ (or $E[\pi(n,t)] < 0$ if $\pi_{\min} = 0$) and the NPV condition above holds.

  When $|N_f| \le n_{\min}$, leaving is not permitted regardless of profitability.

- **Decision to Respond to Challenge:** When node n is challenged for file f, it chooses to generate a proof if the cost of proving is less than the total loss from being slashed.

  Analysis: The true decision must account for the loss of all future revenue streams. Being slashed removes the node from the agreement permanently, forfeiting future rewards. Using a finite analysis horizon h and assuming parameters are roughly constant over that horizon, a simplified estimate of this loss is given by 44:

$$\begin{aligned} \text{NPV}_{\text{future}} &\approx \sum_{i=1}^{h} \left(\frac{\varepsilon_{f(t+i)} \cdot \xi_{\text{KOR/USD}}}{|N_f| \cdot (1+\rho)^i} \right) \\ &\approx \left(\frac{\varepsilon_{f(t)} \cdot \xi_{\text{KOR/USD}}}{|} N_f| \right) \cdot \frac{1-(1+\rho)^{-h}}{\rho} \end{aligned} \tag{44}$$

  In reality, $|N_f|$, $\varepsilon_{f(t)}$, and $\xi_{\text{KOR/USD}}$ will vary over time. The node must forecast these changes when making its decision.

  All terms in the inequalities below are expressed in USD (NPV); KOR flows are converted at $\xi_{\text{KOR/USD}}$ and BTC costs at $\xi_{\text{BTC/USD}}$.

  The decision criterion is then given by 45:

$$c_{\text{proof}}^{\text{BTC}}(t) \cdot \xi_{\text{BTC/USD}} < k_f \cdot \lambda_{\text{slash}} \cdot \xi_{\text{KOR/USD}} + \text{NPV}_{\text{future}} \tag{45}$$

4.3 Macroeconomic Stability

The protocol's long-term economic stability is governed by a system of interconnected feedback loops. These loops regulate node participation and network health by dynamically adjusting economic incentives. The core of this system is a counter-cyclical monetary policy designed to maintain network replication, which operates in tension with a potentially pro-cyclical feedback loop driven by external market prices.

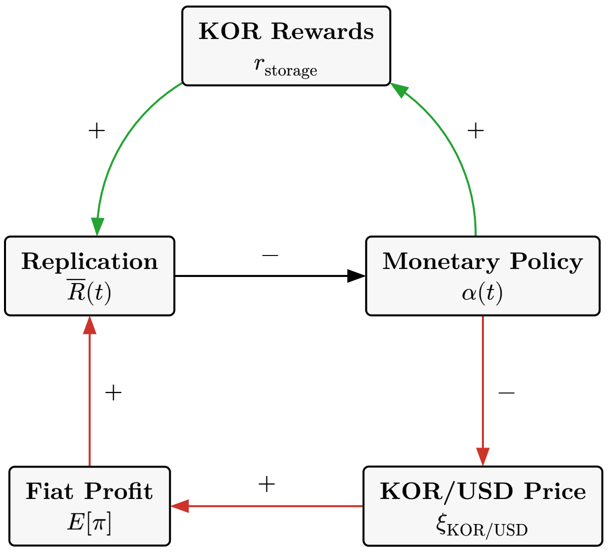

FIGURE 1. Unified macroeconomic feedback system. Signs show $\frac{\partial y}{\partial x}$ for edge $x \to y$. The **green loop** is stabilizing, but saturates when $\alpha(t)$ reaches $\alpha_{\max}$, limiting corrective strength. The **red loop** is destabilizing through the price channel.

The primary mechanism is a **stabilizing negative feedback loop** designed to maintain network health, measured by the emission-weighted replication metric $\overline{R}(t)$.

1. A decline in replication $\overline{R}(t)$ signifies weakening network health.
2. The protocol's monetary policy responds by increasing the emission rate multiplier $\alpha(t)$.
3. Higher emissions increase the KOR-denominated rewards (r_{storage}) paid to storage nodes.
4. Increased rewards improve node profitability, incentivizing existing nodes to stay and new nodes to join, which in turn increases replication.

This constitutes a negative feedback loop: an initial drop in replication triggers a response that counteracts the drop, pushing the system back towards its target equilibrium.

However, the system is also subject to a **destabilizing positive feedback loop** that operates through the external fiat price of KOR ($\xi_{\text{KOR/USD}}$). This loop can amplify external market shocks:

1. A fall in the KOR/USD exchange rate directly reduces the fiat-denominated profitability ($E[\pi]$) for storage nodes, even if KOR rewards remain constant.
2. Lower profitability may cause nodes to exit the network, reducing replication $\overline{R}(t)$.
3. As in the primary loop, the protocol responds to lower replication by increasing the emission rate multiplier $\alpha(t)$.
4. The resulting increase in KOR supply can exert further downward pressure on the KOR/USD exchange rate, amplifying the initial shock.

This pro-cyclical dynamic, where the product of the partial derivatives is positive, presents a systemic risk, particularly during periods of market stress.

The protocol's overall stability depends on the interplay between these two loops.

4.4 Capital Cost Dominance

The protocol's security model fundamentally relies on capital costs (staking and proving) dominating physical storage costs. This economic asymmetry prevents various attacks where nodes might attempt to collect rewards without actually storing data.

4.4.1 Stake Requirements

The per-node base stake for a file is determined by the emission weight and network scale. This formula ensures that:

- Stake requirements scale proportionally with the file's share of network emissions
- The logarithmic scaling in network size provides reasonable growth while preventing excessive barriers
- Earlier files (lower rank) require higher stakes, reflecting their greater emission value

With typical parameters, the capital cost of staking significantly exceeds storage costs. For example, at $|F(t)| = 10^9$ files, the annual opportunity cost of staking (~\$0.05) exceeds the physical storage cost (~\$0.01 for 100MB) by 5x, ensuring that honest storage remains more profitable than attempting to fake it.

4.4.2 Proving Costs and Aggregation

The protocol's proof aggregation mechanism creates economies of scale that benefit legitimate operators while maintaining security. Nodes have a W_{proof} window (approximately 2 weeks) to submit proofs after being challenged, allowing them to aggregate multiple challenges into a single Bitcoin transaction.

With a per-file challenge probability of $p_f = \frac{C_{\text{target}}}{B} = \frac{12}{52560} \approx 0.000228$ per block, a node storing $|F_n|$ files expects approximately $\left(\sum_{f \in F_n} \frac{p_f}{|N_f|}\right) \times 2016$ challenges over the 2-week window. Under equal replication $|N_f| = r$ for all f, this becomes $|F_n| \times \left(\frac{p_f}{r}\right) \times 2016 \approx |F_n| \times \left(\frac{0.46}{r}\right)$. By aggregating these into a single 10,000-byte proof transaction, nodes pay Bitcoin fees only once regardless of the number of challenges.

This aggregation opportunity:

- Makes proving costs negligible even for small operators
- Allows nodes to optimize fee payment timing based on Bitcoin network conditions
- Creates no disadvantage for honest nodes while maintaining security guarantees
- Reduces the risk of network congestion and high BTC proving fees

4.5 Failure Detection

The protocol's challenge mechanism provides strong probabilistic guarantees for detecting data loss, regardless of file size. By sampling a fixed number of sectors ($s_{\text{chal}} = 100$) from each challenged file, the protocol ensures uniform security properties across all stored data.

4.5.1 Detection Probability Analysis

The key insight is that detection probability depends only on the **fraction** of missing data, not the absolute file size. For a file missing fraction μ of its sectors:

- Exact (hypergeometric, without replacement):

$$P(\text{detection} \mid \text{challenged}) = 1 - \Pi_{i=0}^{s_{\text{chal}}-1} \left(\frac{(1-\mu) \cdot s_f - i}{s_f - i} \right) \tag{46}$$

- Binomial approximation (valid when s_f is large and $s_{\text{chal}} \ll s_f$):

$$P(\text{detection} \mid \text{challenged}) \approx 1 - (1-\mu)^{s_{\text{chal}}} \tag{47}$$

For example, if a node deletes half of a file's data ($\mu = 0.5$) and the protocol challenges just two sectors ($s_{\text{chal}} = 2$), the approximate probability of detection is $1 - (1 - 0.5)^2 = 75\%$.

Each file expects $C_{\text{target}} = 12$ challenges per year, regardless of network size. This constant challenge rate ensures predictable security properties as the network scales.

Complete Loss Detection: For a completely missing file (100% data loss), the annual detection probability follows a Poisson process, as shown in 48:

$$P_{\text{annual detection}} = 1 - e^{-C_{\text{target}}} = 1 - e^{-12} \approx 99.9994\% \tag{48}$$

Partial Loss Detection: For partial data loss, we combine the per-challenge detection probability with the expected number of annual challenges, as in 49. With 10% data loss:

$$P_{\text{annual detection}} = 1 - e^{-C_{\text{target}} \cdot P(\text{detection} \mid \text{challenged})} \tag{49}$$

With $\mu = 0.1$ (10% data loss) and $s_{\text{chal}} = 100$, we first calculate the per-challenge detection probability using 47:

$$P(\text{detection} \mid \text{challenged}) = 1 - (1 - 0.1)^{100} = 1 - 0.9^{100} \approx 0.99997 \tag{50}$$

Therefore, using Eq. 49, the annual detection probability is **also**:

$$P_{\text{annual detection}} = 1 - e^{-12 \times 0.99997} \approx 1 - e^{-11.9996} \approx 99.9994\% \tag{51}$$

With a high number of challenged sectors, the detection probability for even 10% data loss is so high that the annual detection rate is difficult to distinguish from that of complete data loss. The protocol assumes files are erasure-coded such that all data can be recovered from this level of degradation.

Beyond this threshold, data loss is detected with near-certain probability. With $s_{\text{chal}} = 100$, storing only 90% of the data (at the erasure coding threshold) results in 99.997% detection probability per challenge, implying an expected time before detection of approximately 10.0 months as given by 52:

$$E[\text{blocks to detection}] = \frac{1}{P_{\text{chal}}(|N_f|, t) \cdot [1 - (1 - \mu)^{s_{\text{chal}}}]} \tag{52}$$

For typical values with $|N_f| = 10$, $p_f = 0.000228$, and $\mu = 0.1$:

$$E[\text{blocks to detection}] \approx \frac{1}{0.0000228 \cdot 0.99997} \approx 43,860 \text{ blocks} \approx \left(43, \frac{860}{52}, 560\right) \times 12 \approx 10.0 \text{ months} \tag{53}$$

5 Security and Risk Analysis

5.1 Attack Vectors

This section analyzes potential attack vectors against the Kontor protocol, their economic implications, and mitigation strategies. The attacks are grouped by the targeted component of the protocol: storage provision, market mechanics, or network operations.

5.1.1 Attacks on Storage Provision

These attacks involve storage nodes failing to meet their obligations, either to increase profit or to save costs.

5.1.1.1 Selective Storage Attack

A malicious node attempts to maximize profit by storing only a fraction of the file data, gambling that the randomly selected challenge sectors will be among those it has retained. This attack encompasses several strategies, from storing 0% (zero-storage) to storing 90% (marginal savings).

Risk-Reward Analysis: The attacker must weigh the marginal storage cost savings against the risk of detection and total loss. For a node storing fraction $(1 - \mu)$ of file f:

Savings per block: $\mu \cdot c^{\text{USD}}_{\text{storage}}(f, t)$ (only the marginal storage cost)

Risk per block: $P_{\text{chal}}(|N_f|, t) \cdot [1 - (1 - \mu)^{s_{\text{chal}}}] \cdot \left[k_f \cdot \lambda_{\text{slash}} \cdot \xi_{\text{KOR/USD}} + \text{NPV}_{\text{future}}\right]$

where:

- $P_{\text{chal}}(|N_f|, t) = \frac{p_f}{|N_f|}$ is the probability of being challenged
- $[1 - (1 - \mu)^{s_{\text{chal}}}]$ is the probability of detection if challenged
- The loss includes both the slashed stake and all future rewards (NPV), as defined in Eq. 44.

Files are assumed to be erasure-coded to tolerate up to 10% data loss, meaning the network can recover from this level of degradation. However, nodes that allow data to degrade to this threshold face near-certain detection. With $s_{\text{chal}} = 100$, storing only 90% of the data (at the erasure coding threshold) results in 99.997% detection probability per challenge. The expected time to detection (as given in 52) is approximately 10.0 months for typical values with $|N_f| = 10$, $p_f = 0.000228$, and $\mu = 0.1$.

The attacker loses all future rewards upon detection, making the NPV of the attack negative even with high discount rates.

5.1.1.2 Collusion Attack

Multiple nodes (k of n total) storing a file coordinate to have one member fail a challenge, aiming to profit from the redistributed stake.

Analysis: Let $n = |N_f|$ and let the colluding subset have size k. One colluder n_s intentionally fails a challenge.

- Slashed amount: $S = k_f \cdot \lambda_{\text{slash}}$ is deducted from n_s.
- Burn: $S_{\text{burned}} = \beta_{\text{slash}} \cdot S$ is burned.
- Redistribution: $S_{\text{redist}} = (1 - \beta_{\text{slash}}) \cdot S$ is split equally among the $(n - 1)$ remaining nodes.

The colluding group's net change is: Δ
$\text{KOR}_{\text{colluders}} = -S + \left(\frac{k-1}{n-1}\right) \cdot S_{\text{redist}} = -S + \left(\frac{k-1}{n-1}\right) \cdot (1 - \beta_{\text{slash}}) \cdot S$

Profitability would require $\left(\frac{k-1}{n-1}\right) \cdot (1 - \beta_{\text{slash}}) > 1$, which is impossible since each factor is ≤ 1. Hence, this collusion is strictly unprofitable.

5.1.1.3 Disk-Sharing Attack

An attacker creates multiple Sybil node identities but stores only a single physical copy of the data, aiming to collect multiple rewards for a single storage cost while undermining the protocol's data replication guarantees. The attacker saves on physical storage costs for all but one identity, but must still cover the full capital costs (staking and proving) for every identity.

Analysis: Consider an attacker creating n_{sybil} identities to store the same file f. The attacker's costs are:

- Physical storage: $c_{\text{storage}}^{\text{USD}}(f, t)$ (only one copy needed)
- Stake opportunity cost: $n_{\text{sybil}} \cdot k_f \cdot \xi_{\text{KOR/USD}} \cdot \rho$ (full stake per identity)
- Expected proving costs: $n_{\text{sybil}} \cdot P_{\text{chal}}\big(|N_f|, t\big) \cdot c_{\text{proof}}^{\text{BTC}}(t) \cdot \xi_{\text{BTC/USD}}$ (each identity may be challenged)

The attacker's revenue is $n_{\text{sybil}} \cdot r_{\text{storage}}(n, f, t) \cdot \xi_{\text{KOR/USD}}$. A simple profitability proxy per block (USD) is:

$$\pi_{\text{sybil}} \approx n_{\text{sybil}} \cdot \left(\frac{\varepsilon_{f(t)}}{|N_f| + n_{\text{sybil}}} \right) \cdot \xi_{\text{KOR/USD}} - n_{\text{sybil}} \cdot k_f \cdot \rho \cdot \xi_{\text{KOR/USD}} - c_{\text{storage}}^{\text{USD}}(f, t) \quad (54)$$

At the margin (adding one more identity), the change is approximately:

$$\Delta\pi \approx \frac{\varepsilon_{f(t)} \cdot \xi_{\text{KOR/USD}} \cdot |N_f|}{\big(|N_f| + n_{\text{sybil}}\big)^2} - k_f \cdot \rho \cdot \xi_{\text{KOR/USD}} \quad (55)$$

As $|N_f|$ or n_{sybil} grow, the first term shrinks as $\frac{1}{(|N_f| + n_{\text{sybil}})^2}$ while the second term is constant, so beyond a small scale the attack is unprofitable even when saving on c_{storage}.

Mitigation: Capital costs dominate storage costs, making the attack unprofitable.

5.1.2 Attacks on Market Mechanics

These attacks exploit the protocol's economic rules to manipulate outcomes or gain an unfair advantage.

5.1.2.1 Sybil Attack (Risk Compartmentalization)

An attacker creates multiple node identities to limit their downside risk from correlated failures. While a single identity is more robust to an isolated, random failure, consolidating a large portfolio creates a single point of failure for systemic operational problems. A correlated failure event could cause the node to fail challenges for many files at once, leading to complete stake forfeiture. By splitting files across many Sybil identities, an operator can compartmentalize this operational risk.

Analysis: The protocol defends against this by making portfolio splitting more capital-intensive. The dynamic stake factor, $\lambda_{\text{stake}} = 1 + \frac{\lambda_{\text{slash}}}{\ln(2+|F_n|)}$, imposes a capital premium on nodes with fewer files. The Sybil strategy is only rational if the benefit from risk reduction exceeds this significant increase in capital cost.

Example with $\lambda_{\text{slash}} = 30.0$ and an operator with a 100,000-file portfolio, where the average file has a base stake requirement of $k_f = 1.0$ KOR:

Node Profile	**Files $\lvert F_n \rvert$**	**Stake Factor (λ_{stake})**	**Required Stake (k_{req})**
Sybil Node	1	28.30x	28.30 KOR
Small Node	10	13.56x	135.6 KOR
Large Consolidated Node	100,000	3.60x	360,000 KOR

To run the portfolio as a single consolidated entity, the operator must post 360,000 KOR in stake. To run the same portfolio as 100,000 individual Sybil nodes, the operator would need to post 2,830,000 KOR. This is a ~7.9x capital increase, rendering large-scale compartmentalization attacks economically irrational. The mechanism correctly aligns capital efficiency with network health.

Mitigation: Dynamic stake factor $\lambda_{\text{stake}} = 1 + \frac{\lambda_{\text{slash}}}{\ln(2+|F_n|)}$ penalizes fragmentation.

5.1.2.2 Sybil-Based Reward Amplification (Self-Sponsorship / Duplicate Staking)

An attacker, already having a presence for a file f (either directly or via another identity n_1), introduces a new Sybil identity (n_2) to also store file f. The aim is to collect an additional share of the file's rewards. This can be framed as n_1 sponsoring its own Sybil n_2, or simply adding n_2 as a duplicate storer. n_2 typically only stores file f, thus incuring a high stake factor λ_S.

Analysis of Economic Implications:

Let:

- $\varepsilon_{f(t)}$: Total KOR emissions for file f at time t.
- k_f: Per-node base stake requirement for file f at time t.
- ρ: Operator's per-block opportunity cost rate for capital.
- $\xi_{\text{KOR/USD}}$: Exchange rate from KOR to USD.
- $|N_f|$: The number of nodes storing file f **before** the Sybil node n_2 joins.
- $\lambda_S = 1 + \frac{\lambda_{\text{slash}}}{\ln(3)}$: Stake factor for the Sybil node n_2 (assuming it only stores file f, so its portfolio size $|F_{n_2}| = 1$).

Operator's Rewards Before Sybil Join: The operator, through their existing presence (e.g., node n_1), earns: $r_{\text{orig}} = \frac{\varepsilon_{f(t)}}{|N_f|}$

Operator's Rewards After Sybil Join: After Sybil node n_2 joins, there are now $|N_f| + 1$ nodes storing file f.

- The original presence n_1 now earns: $r_1^{\text{new}} = \frac{\varepsilon_{f(t)}}{|N_f| +1}$
- Sybil node n_2 earns: $r_2 = \frac{\varepsilon_{f(t)}}{|N_f| +1}$

The operator's total reward from file f (across n_1 and n_2) is: $r_{\text{total}} = r_1^{\text{new}} + r_2 = 2 \cdot \frac{\varepsilon_{f(t)}}{|N_f| +1}$

Marginal Gain in KOR Rewards for the Operator: The net change in KOR rewards for the operator is: $\Delta r = r_{\text{total}} - r_{\text{orig}}$ $\Delta r = \left(2 \cdot \frac{\varepsilon_{f(t)}}{|N_f| +1}\right) - \left(\frac{\varepsilon_{f(t)}}{|N_f|}\right)$ $\Delta r = \varepsilon_{f(t)} \cdot \left[\frac{2(|N_f|)-(|N_f| +1)}{(|N_f|)(|N_f| +1)}\right]$ $\Delta r = \varepsilon_{f(t)} \cdot \left(\frac{2(|N_f|)-(|N_f| +1)}{(|N_f|)(|N_f| +1)}\right)$

This marginal gain Δr is positive only if $|N_f| > 1$. If the operator's node n_1 was the sole storer ($|N_f| = 1$), then $\Delta r = 0$, meaning no additional KOR rewards are gained from this strategy, only costs incurred.

Marginal Cost for the Operator (in KOR terms for direct comparison): The additional KOR-denominated opportunity cost for the Sybil node n_2's stake is: $C_{\text{sybil,KOR}} = k_f \cdot \lambda_S \cdot \rho$

Profitability Condition and $\boldsymbol{\lambda_{\text{slash}}}$ Threshold: The attack is unprofitable if the value of the marginal KOR gain in rewards does not exceed the marginal KOR opportunity cost: $\Delta r \leq C_{\text{sybil,KOR}}$ $\varepsilon_{f(t)} \cdot \left(\frac{2(|N_f|)-(|N_f| +1)}{(|N_f|)(|N_f| +1)}\right) \leq k_f \cdot \left(1 + \frac{\lambda_{\text{slash}}}{\ln(3)}\right) \cdot \rho$

Let $r_k = \frac{\varepsilon_{f(t)}}{k_f}$ be the file's KOR emission-to-stake ratio. Let $G_N = \frac{|N_f| -1}{(|N_f|)(|N_f| +1)}$ be the network geometry factor affecting marginal gain. This factor is maximized at $\frac{1}{6}$ when $|N_f| = 2$ and tends to $\frac{1}{|N_f|}$ for large $|N_f|$. The condition for unprofitability becomes: $r_k \cdot G_N \leq \left(1 + \frac{\lambda_{\text{slash}}}{\ln(3)}\right) \cdot \rho$

For the attack to be unprofitable, λ_{slash} must satisfy:

$$\lambda_{\text{slash}} \geq \ln(3) \cdot \left(\left(\frac{r_k \cdot G_N}{\rho}\right) - 1\right) \tag{56}$$

This inequality shows the minimum λ_{slash} required to deter such attacks, given the file's emission/stake characteristics (r_k), the number of existing storers ($|N_f|$ influencing G_N), and the operator's discount rate (ρ). If the term inside the square brackets is negative (i.e., $\left(r_k \frac{G_N}{\rho}\right) < 1$), then any non-negative λ_{slash} suffices, as the attack would be unprofitable even without the Sybil penalty.

Numerical Example (parameters from Section 6.2): Consider the profitability condition: $\varepsilon_{f(t)} \cdot G_N > k_f \cdot \lambda_S \cdot \rho$. Using values from Section 6.2 and scenario assumptions: $c_{\text{stake}} = 2{,}000{,}000$ KOR, $F_{\text{scale}} = 50{,}000$, $|F(t)| = 100{,}000$, $\rho = 0.0000034$, $|N_f| = 10$. And $\varepsilon(t) \approx 1{,}903$ KOR/block (total network emissions, assuming total supply near initial supply). If $\lambda_{\text{slash}} = 30$, then $\lambda_S = 1 + \frac{30}{\ln(3)} \approx 1 + \frac{30}{1.0986} \approx 1 + 27.30 = 28.30$.

The marginal gain part (left side of inequality for profit): $\varepsilon_{f(t)} \cdot G_N = \left(\varepsilon(t) \cdot \frac{\omega_f}{\Omega(t)}\right) \cdot \left(\frac{|N_f| -1}{(|N_f|)(|N_f| +1)}\right)$ The cost part (right side of inequality for profit): $k_f \cdot \lambda_S \cdot \rho = \left(\left(\frac{\omega_f}{\Omega(t)}\right) \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t)|}{F_{\text{scale}}}\right)\right) \cdot \lambda_S \cdot \rho$

Comparing $\varepsilon(t) \cdot G_N$ with $c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t)|}{F_{\text{scale}}}\right) \cdot \lambda_S \cdot \rho$ (after canceling $\frac{\omega_f}{\Omega(t)}$ from both sides of the profitability condition $\varepsilon_{f(t)} G_N > k_f \lambda_S \rho$): Left side: $1903 \cdot \frac{10-1}{10 \cdot (10+1)} = 1903 \cdot \frac{9}{110} \approx 155.7$ KOR/block Right side: $2{,}000{,}000 \cdot \ln\left(1 + \frac{100000}{50000}\right) \cdot 28.30 \cdot 0.0000034$

$$= 2{,}000{,}000 \cdot \ln(3) \cdot 28.30 \cdot 0.0000034 \tag{57}$$

$= 2{,}000{,}000 \cdot 1.0986 \cdot 28.30 \cdot 0.0000034 \approx 210.4$ KOR/block

Since $155.7 < 210.4$, the attack is unprofitable with $\lambda_{\text{slash}} = 30$ when ρ corresponds to a 20% annual rate. This demonstrates that $\lambda_{\text{slash}} = 30$ provides sufficient security margin under realistic opportunity cost assumptions.

Mitigation: The dynamic stake factor $\lambda_{\text{stake}}(n) = 1 + \frac{\lambda_{\text{slash}}}{\ln(2+|F_n|)}$ imposes a severe capital inefficiency on Sybil nodes holding few files (e.g., only the target file f). As demonstrated by the profitability condition and the numerical example, a sufficiently high λ_{slash} makes these attacks economically irrational, as the marginal reward gain is dwarfed by the high opportunity cost of the Sybil's amplified stake. The derived threshold for λ_{slash} provides a clear guideline for parameterizing the system against such attacks.

5.1.2.3 Node Identities and Inter-Node Competition

The protocol is designed to be indifferent to node identities. Economic incentives create natural competition among nodes with no inherent synergies from collusion. This fundamental property neutralizes several identity-based attacks:

5.1.2.3.1 Crowding-Out Attack

An attacker creates multiple Sybil nodes to join a file agreement, attempting to dilute honest nodes' rewards and force them to exit.

Analysis: Consider an attacker controlling k Sybil nodes joining a file with n existing honest nodes. The per-node reward changes from $\frac{\varepsilon_{f(t)}}{n}$ to $\frac{\varepsilon_{f(t)}}{n+k}$. The attacker's total reward is $k \cdot \frac{\varepsilon_{f(t)}}{n+k}$.

To maximize their reward, the attacker would want to maximize $\frac{k}{n+k}$, which approaches 1 as $k to \infty$. However, each Sybil node requires:

- Stake: $k_f \cdot \lambda_{\text{stake}}$ where $\lambda_{\text{stake}} \approx 28.30$ for single-file nodes
- Proving costs when challenged
- Operational overhead

The attacker's profit per Sybil node is:

$$\pi_{\text{sybil}} = \frac{\varepsilon_{f(t)}}{n+k} \cdot \xi_{\text{KOR/USD}} - k_f \cdot \lambda_{\text{stake}} \cdot \xi_{\text{KOR/USD}} \cdot \rho - E\left[c_{\text{prove}}^{\text{USD}}\right] \tag{58}$$

As k increases, the first term approaches zero while the costs remain constant. The attack becomes increasingly unprofitable with scale. Moreover, honest nodes can simply wait—as the attacker bleeds capital, they will eventually exit, restoring profitability.

5.1.2.3.2 Coordinated Departure Attack

Multiple colluding nodes storing a file coordinate to leave simultaneously, attempting to drive replication below $n_{\min}$ and trigger data loss.

Analysis: Consider k colluding nodes among n total nodes storing file f. For the attack to succeed, they need $n - k < n_{\min}$, meaning $k > n - n_{\min}$.

The leave fee increases quadratically as replication approaches $n_{\min}$:

$$\varphi_{\text{leave}}(f,t) = k_f \cdot \left(\frac{n_{\min}}{|N_f|}\right)^2 \tag{59}$$

As nodes leave sequentially:

- First leaver pays: $k_f \cdot \left(\frac{n_{\min}}{n}\right)^2$
- Second leaver pays: $k_f \cdot \left(\frac{n_{\min}}{n-1}\right)^2$
- k-th leaver pays: $k_f \cdot \left(\frac{n_{\min}}{n-k+1}\right)^2$

The total cost grows rapidly. For example, with $n = 5$, $n_{\min} = 3$, and $k_f = 100$ KOR:

- First leaver: 36 KOR
- Second leaver: 56.25 KOR
- Third leaver: 100 KOR
- Total cost: 192.25 KOR

Additionally, the rewards for remaining nodes increase hyperbolically, creating strong incentives for new nodes to join before the attack completes.

Mitigation: Economic incentives strongly discourage both attacks through capital inefficiency and market dynamics.

5.1.2.4 Wash-Trading Attack

An attacker stores their own data through nodes they control to farm rewards or manipulate network metrics.

Analysis: The finite-horizon annuity model creates specific economics for wash-trading. To store a file, an attacker must pay a one-time, upfront fee υ_f (determined at creation time), which is entirely burned. The attack is profitable only if the Net Present Value (NPV) of the future reward stream over horizon h exceeds this upfront, non-recoverable cost. Assuming the attacker gets all rewards and that future emissions remain at the creation level over this horizon, the profitability condition is:

$$\text{NPV}_{\text{rewards}} \approx \sum_{i=1}^{h} \left(\frac{\varepsilon_{f(t_{\text{create}}+i)}}{(1+\rho)^i} \right) > \upsilon_f \tag{60}$$

Substituting the definition of the user fee ($\upsilon_f = \chi_{\text{fee}} \cdot k_f$) and emissions ($\varepsilon_{f(t)} = \varepsilon(t) \cdot \left(\frac{\omega_f}{\Omega(t)}\right)$) and assuming ε_f is approximately constant over h gives:

$$\left(\varepsilon(t_{\text{create}}) \cdot \frac{1-(1+\rho)^{-h}}{\rho} \right) \cdot \left(\frac{\omega_f}{\Omega(t_{\text{create}})} \right) > \chi_{\text{fee}} \cdot \left(\frac{\omega_f}{\Omega(t_{\text{create}})} \right) \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t_{\text{create}})|}{F_{\text{scale}}} \right) \tag{61}$$

The terms $\frac{\omega_f}{\Omega(t_{\text{create}})}$ cancel, yielding the finite-horizon condition:

$$\left(\varepsilon(t_{\text{create}}) \cdot \frac{1-(1+\rho)^{-h}}{\rho} \right) > \chi_{\text{fee}} \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t_{\text{create}})|}{F_{\text{scale}}} \right) \tag{62}$$

Since $\varepsilon(t) = \text{KOR}_{\text{total}}(t-1) \cdot \frac{\alpha(t-1)\cdot\mu_0}{B}$, this becomes a condition on the total KOR supply and other parameters. Since nodes also pay ongoing proving costs, the actual threshold for profitability is even higher.

Mitigation: Setting χ_{fee} appropriately based on the security condition ensures wash-trading remains unprofitable.

5.1.2.5 File Size Manipulation Attacks

An attacker creates files at extreme size boundaries to exploit the logarithmically-weighted replication metric $\overline{R}(t)$ and manipulate the global emission rate multiplier $\alpha(t)$.

5.1.2.5.1 Small File Attack (Spam)

An attacker creates many tiny files to influence the health metric. The cost to create a file and its influence on the replication metric are both proportional to the logarithm of its size. This means the cost-to-influence ratio is constant, regardless of file size. An attacker cannot gain leverage by creating many small files instead of one large one. However, spamming can still exhaust the file challenge budget, increase node overhead, and dilute the challenge probability for legitimate files in the capped regime.

Mitigation: Enforcing a minimum file size s_{min} and charging a fee υ_f make this attack economically unattractive.

5.1.2.5.2 Large File Attack (Replication Phantom)

An attacker creates a single massive file with minimal replication to drag down the weighted average. This artificially depresses $\overline{R}$, which increases the emission rate multiplier $\alpha(t)$ and boosts the attacker's revenue from all other files they are storing.

Analysis:

- The cost to create a file is $\upsilon_f = \chi_{\text{fee}} \cdot k_f$, where $k_f = \left(\frac{\omega_f}{\Omega(t_{\text{create}})}\right) \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{|F(t_{\text{create}})|}{F_{\text{scale}}}\right)$ and $\omega_f = \frac{\ln\left(s_f^{\text{bytes}}\right)}{\ln(1+\text{rank}_f)}$

- The file's influence on $\overline{R}$ is proportional to ω_f

Both cost and influence scale identically with ln(size). This means there's no special advantage to using a large file versus multiple smaller files with the same total emission weight. The attacker gains no leverage from file size manipulation.

Mitigation: The logarithmic weighting for both cost and influence eliminates size-based leverage. The maximum file size $s_{\max}$ provides an additional safeguard.

5.1.2.6 Data Gatekeeping Attack

A cartel of existing nodes for a file f collude to refuse to share the file data with newcomers ("Entrants"). This prevents new nodes from joining the agreement, which stops reward dilution and creates a permanent, super-profitable monopoly for the cartel.

Analysis: Without access to file data, new nodes cannot generate valid proofs when challenged, effectively blocking entry. The cartel could maintain exactly $n_{\min}$ nodes to maximize per-node rewards while preventing departure.

Mitigation: The protocol implements an optional, market-driven "Sponsored Access" mechanism. An Entrant who cannot obtain file data through other means can solicit sponsorship from any existing node in N_f. A sponsorship is a temporary, on-chain contract where a Sponsor provides the data in exchange for a commission rate (γ_{rate}) on the Entrant's rewards for a fixed duration (γ_{duration}).

This creates a competitive market that breaks cartels through defection incentives. In equilibrium with a finite sponsorship duration D, sponsors will bid γ_{rate} down to where their expected profit equals their bandwidth cost:

$$\gamma_{\text{eq}}(D) \approx \frac{c_{\text{transfer}}^{\text{USD}} \cdot \rho \cdot \left(|N_f| + 1\right)}{\varepsilon_{f(t)} \cdot \xi_{\text{KOR/USD}} \cdot (1 - (1+\rho)^{-D})} \tag{63}$$

Numerical Example:
- Transfer cost: $c_{\text{transfer}}^{\text{USD}} = \$0.10$ (bandwidth for 10MB file)
- File emissions: $\varepsilon_{f(t)} = 0.002$ KOR/block
- KOR price: $\xi_{\text{KOR/USD}} = \$0.20$
- Discount rate: $\rho = 0.0001$ per block
- Current storers: $|N_f| = 3$
- Sponsorship duration: $D = 10,000$ blocks (approx. 2.8 "weeks")

$$\gamma_{\text{eq}}(10,000) \approx \frac{0.10 \cdot 0.0001 \cdot 4}{0.002 \cdot 0.20 \cdot (1 - (1+0.0001)^{-10,000})} \approx 0.158 = 15.8\% \tag{64}$$

A rational node within the cartel has strong incentive to defect and capture this sponsorship value. This competitive pressure ensures there is always a permissionless path for new nodes to join any agreement. Additionally, any party with backup access to the file data can break a monopoly by joining as a new node and sponsoring others. This mechanism fails if the attacker controls all nodes in a given agreement and there are no altruistic nodes replicating the file.

5.1.3 Attacks on Network Integrity

These attacks target the underlying operations of the protocol, such as the challenge process or resource management.

5.1.3.1 Challenge Censorship Attack

A Bitcoin miner with an economic interest in Kontor (e.g., competing storage nodes) selectively excludes proof transactions from blocks to cause honest nodes to be slashed.

Analysis: The attack requires the miner to:
1. Identify Kontor proof transactions in the mempool
2. Forgo the transaction fees from these proofs

3. Maintain censorship for the entire W_{proof} window (≈2 weeks)

With Bitcoin's decentralized mining, the attacker needs substantial hashrate to sustain censorship. Even with 10% of global hashrate, the probability of censoring a proof for the full window is $(0.1)^{\{2016\}} \approx 0$, effectively impossible. The attacker also loses fee revenue.

Mitigation: Bitcoin's censorship resistance and the economic loss from foregone fees make this attack impractical. The long proof window provides ample opportunity for honest miners to include proofs.

5.1.3.2 Challenge Front-Running Attack

An attacker floods the Bitcoin mempool with high-fee transactions to prevent a challenged node's proof from being included within the W_{proof} response window.

Analysis: To execute this attack, the attacker must:
1. Monitor for challenged nodes' proof attempts
2. Continuously outbid the proof transaction's fee
3. Maintain this for up to $W_{\text{proof}} = 2016$ blocks

The cost is substantial. To fill blocks and prevent proof inclusion, the attacker must pay for significant block space. Assuming:
- 400kB available for Kontor proofs per block
- Minimum fee premium of 10 sat/vByte above the proof's fee
- 2016 blocks maximum duration

Attack cost: $400,000 \times 10 \times \frac{2016}{10^8} = 80.64$ BTC (≈$8 million at $100k/BTC)

The challenged node, facing potential loss of stake plus all future rewards, has strong incentive to escalate fees. This creates a bidding war the attacker is likely to lose.

Mitigation: The prohibitive cost in Bitcoin fees, combined with the challenged node's strong incentive to outbid the attacker, makes this attack economically irrational.

6 Appendix

6.1 Notation Guide

The document uses systematic notation conventions for mathematical clarity:

Capitalization:
- Capital letters denote global totals, aggregates, and sets
- Lowercase letters denote local/specific variables and set elements

Variable Prefixes:
- n_*: Counts and quantities
- F_*: File-related sets (e.g., F_n is the set of files stored by node n)
- k_*: Stake-related variables (in KOR)
- s_*: Size-related variables (file/sector/proof sizes, sector counts)
- S_*: Network-wide storage totals
- φ_*: Fees (units specified by context or superscript)
- Φ_*: Burn-related variables (e.g., Φ_{burned})
- p_*, P_*: Probability values and functions
- c_*: Costs (currency denoted by superscript: c^{USD}, c^{BTC})
- r_*: Rewards per node (in KOR)
- ξ_*: Exchange rates (currency conversion factors)
- $E[\cdot]$: Expected values
- υ_*: User fees (in KOR)
- C_*: Challenges
- M_*: Sponsorship-related sets (e.g., $M(t)$ is the set of all sponsorship agreements)
- W_*: Window parameters (e.g., W_R, W_{proof})
- m_*: Health-related metrics (e.g., $m(t)$ is the health buffer)

Greek Letters:

- α: Emission rate multiplier
- β: Burn proportions
- ε: Emissions (in KOR)
- γ: Commission rates and variables
- κ: Control system parameters (e.g., κ_α)
- λ: Stake-related multipliers
- μ: Inflation rates and data loss fractions
- ω, Ω: Emission weights (per-file and total)
- π: Profit (in USD)
- ρ: Discount rates and opportunity costs
- θ: Files challenged per block

6.2 Protocol Parameters

Parameter	Example	Units	Definition	Too Low	Too High
B	52560	blocks/year	Bitcoin blocks per 365-day year	N/A	N/A
$s_{\text{sector}}^{\text{bytes}}$	1024	bytes	Fixed sector size	N/A	N/A
$\text{KOR}_{\text{initial}}$	10^9	KOR	Total KOR supply at genesis	N/A	N/A
μ_0	0.10	1/year	Baseline annual inflation rate	Insufficient incentives; stalled growth	Devaluation of KOR
κ_α	0.5	dimensionless	Emission rate sensitivity to average network replication	Slow response to network degradation	Volatile inflation rate
$\alpha_{\max}$	2.0	dimensionless	Maximum emission rate multiplier	Limits response to poor health	Permits high inflation
c_{stake}	2×10^6	KOR	Base stake normalization factor	Low stake requirements reduce security	High stake requirements slow adoption
F_{scale}	50,000	files	Network size normalization factor	Stake reqs. grow too quickly	Stake reqs. grow too slowly
$s_{\min}$	10 KB	bytes	Minimum file size	N/A	N/A
$s_{\max}$	100 MB	bytes	Maximum file size	N/A	N/A
χ_{fee}	0.003	dimensionless	Ratio of user fee to per-node base stake	Low-value data	Slow adoption
$n_{\min}$	3	nodes	Minimum number of storage nodes required per file	Insufficient redundancy	Excessive cost
λ_{slash}	30	dimensionless	Sybil-resistance stake req. factor	Insufficient deterrent for Sybil attacks	Excessive capital requirements
β_{slash}	0.5	fraction	Proportion of slashed stake burned	Monopolistic behavior	Reduced anti-collusion incentive
W_R	2016	blocks	Network replication averaging window	Too volatile	Too sluggish
C_{target}	12	challenges/year	Target challenges per file per year	Reduced security	Higher proving and computational costs
s_{chal}	100	sectors	Sectors sampled per challenge	Lower detection of partial loss	Increased proving overhead

Parameter	Example	Units	Definition	Too Low	Too High
s_{proof}	10,000	vBytes	Size of an aggregated proof	N/A	N/A
W_{proof}	2016	blocks	Proof verification window	Higher operational risk for storage nodes	Delayed punishment for dishonest nodes

6.3 Economic-Modeling Parameters

Parameter	Example	Units	Definition
ρ	0.0000034	1/block	Per-block discount rate (~20% annual)
$\xi_{\text{KOR/USD}}$	0.20	USD/KOR	Reference USD/KOR exchange rate
$\xi_{\text{BTC/USD}}$	100,000	USD/BTC	Reference BTC/USD exchange rate
$\varphi_{\text{BTC}}^{\text{rate}}$	50	sat/vByte	Bitcoin transaction fee rate
$c_{\text{proof}}^{\text{BTC}}(t)$	500,000	sat	Bitcoin cost of submitting an aggregated proof
p_{fail}	0.001	probability	Probability of an honest node failing a challenge
h	2016	blocks	Number of future blocks considered in NPV calculations
π_{min}	0	USD/block	Minimum net expected profit a node requires
$c_{\text{byte-block}}^{\text{USD}}$	10^{-12}	USD/(byte · block)	Marginal cost of storing one byte for one block
$c_{\text{transfer}}^{\text{USD}}$	0.10	USD	Bandwidth cost to transfer file data for sponsorship
$P_{\text{any}}(n,t)$	varies	probability	Probability node n is challenged on at least one file

6.4 Equations

Equation	Expression	Scaling	Units
File Emission Weight	$\omega_f \overset{\text{def}}{=} \frac{\ln\left(s_f^{\text{bytes}}\right)}{\ln(1+\text{rank}_f)}$	$\propto \frac{\ln\left(s_f^{\text{bytes}}\right)}{\ln(\text{rank}_f)}$	dimensionless
Total Emission Weight	$\Omega(t) \overset{\text{def}}{=} \sum_{f\in F(t)} \omega_f$	Sub-linear in $\lvert F(t)\rvert$	dimensionless
Per-Node Base Stake	$k_f \overset{\text{def}}{=} \left(\frac{\omega_f}{\Omega(t_{\text{create}})}\right) \cdot c_{\text{stake}} \cdot \ln\left(1 + \frac{\lvert F(t_{\text{create}})\rvert}{F_{\text{scale}}}\right)$	$\propto \left(\frac{\omega_f}{\Omega(t_{\text{create}})}\right) \cdot \ln(\lvert F(t_{\text{create}})\rvert)$	KOR
Emission Rate Multiplier (where $m(t) = \overline{R}(t) - n_{\text{min}}$)	$\alpha(t) \overset{\text{def}}{=} \begin{cases} \min\left(\alpha_{\text{max}}, 1+\frac{\kappa_\alpha}{m(t)}\right) & \text{if } m(t)>0 \\ \alpha_{\text{max}} & \text{if } m(t)\leq 0 \end{cases}$	Piecewise	dimensionless
Total KOR Emissions	$\varepsilon(t) \overset{\text{def}}{=} \text{KOR}_{\text{total}}(t-1) \cdot \frac{\alpha(t-1)\cdot\mu_0}{B}$	$\propto \text{KOR}_{\text{total}}(t)$	KOR/block
File-Specific Emissions	$\varepsilon_{f(t)} \overset{\text{def}}{=} \varepsilon(t) \cdot \left(\frac{\omega_f}{\Omega(t)}\right)$	$\propto \frac{\omega_f}{\Omega(t)}$	KOR/block
User Storage Fee	$\upsilon_f \overset{\text{def}}{=} \chi_{\text{fee}} \cdot k_f$	$\propto \frac{\omega_f}{\Omega(t_{\text{create}})}$	KOR
Leave Fee	$\varphi_{\text{leave}}(f,t) \overset{\text{def}}{=} k_f \cdot \left(\frac{n_{\text{min}}}{\lvert N_f\rvert}\right)^2$	$\propto \frac{k_f}{(\lvert N_f\rvert)^2}$	KOR
Dynamic Stake Factor	$\lambda_{\text{stake}}(n) \overset{\text{def}}{=} 1 + \frac{\lambda_{\text{slash}}}{\ln(2+\lvert F_n\rvert)}$	$\propto \frac{1}{\ln(\lvert F_n\rvert)}$	dimensionless
Total Required Stake	$k_{\text{req}}(n,t) \overset{\text{def}}{=} \left(\sum_{f\in F_n} k_f\right) \cdot \lambda_{\text{stake}}(n)$	Complex	KOR
Base Per-Node Reward	$r_{\text{n}\mid\text{f}}(t) \overset{\text{def}}{=} \frac{\varepsilon_{f(t)}}{\lvert N_f\rvert}$	$\propto \frac{\varepsilon_{f(t)}}{\lvert N_f\rvert}$	KOR/block
Node Storage Reward	$r_{\text{storage}}(n,f,t) \overset{\text{def}}{=} r_{\text{n}\mid\text{f}}(t) \cdot \left(1 - \gamma_{\text{paid}}(n,f,t) + \gamma_{\text{earned}}(n,f,t)\right)$	$\propto r_{\text{n}\mid\text{f}}(t)$	KOR/block
Commission Paid	$\gamma_{\text{paid}}(n,f,t) \overset{\text{def}}{=} \begin{cases} \gamma_{\text{rate}} & \text{if entrant} \\ 0 & \text{otherwise} \end{cases}$	0 or γ_{rate}	fraction
Commission Earned	$\gamma_{\text{earned}}(n,f,t) \overset{\text{def}}{=} \sum_{m\in M(n,f,t)} \gamma_{\text{rate}}$	Sum over sponsorships	fraction

Equation	**Expression**	**Scaling**	**Units**
Expected Slashing Revenue	$E[r_{\text{slash}}(n,f,t)] \overset{\text{def}}{=} \frac{p_f \cdot p_{\text{fail}} \cdot (1-\beta_{\text{slash}}) \cdot k_f \cdot \lambda_{\text{slash}}}{\lvert N_f \rvert}$	$\propto \frac{k_f \cdot \lambda_{\text{slash}}}{\lvert N_f \rvert}$	KOR/block
Files Challenged per Block	$\theta(t) \overset{\text{def}}{=} \frac{C_{\text{target}} \cdot \lvert F(t) \rvert}{B}$	$\propto \lvert F(t) \rvert$	files/block
Challenge Probability	$P_{\text{chal}}(\lvert N_f \rvert, t) \overset{\text{def}}{=} \frac{p_f}{\lvert N_f \rvert}$	$\propto \frac{1}{\lvert N_f \rvert}$	probability/block
Per-File Challenge Probability	$p_f \overset{\text{def}}{=} \frac{C_{\text{target}}}{B}$	Constant	probability/block
Expected Proving Cost	$E\left[c^{\text{USD}}_{\text{prove}}(n,t)\right] \overset{\text{def}}{=} P_{\text{any}}(n,t) \cdot s_{\text{proof}} \cdot \varphi^{\text{rate}}_{\text{BTC}}(t) \cdot \xi_{\text{BTC/USD}}$	$\propto 1 - \Pi_{f \in F_n}\left(1 - \frac{p_f}{\lvert N_f \rvert}\right)$	USD/block
Expected Slashing Cost	$E\left[c^{\text{USD}}_{\text{slash}}(f,t)\right] \overset{\text{def}}{=} P_{\text{chal}}(\lvert N_f \rvert, t) \cdot p_{\text{fail}} \cdot k_f \cdot \lambda_{\text{slash}} \cdot \xi_{\text{KOR/USD}}$	$\propto \frac{k_f \cdot \lambda_{\text{slash}}}{\lvert N_f \rvert}$	USD/block
Stake Opportunity Cost	$c^{\text{USD}}_{\text{opp}}(n,t) \overset{\text{def}}{=} k_{\text{req}}(n,t) \cdot \xi_{\text{KOR/USD}} \cdot \rho$	$\propto k_{\text{req}}(n,t)$	USD/block
Physical Storage Cost	$c^{\text{USD}}_{\text{storage}}(f,t) \overset{\text{def}}{=} s^{\text{bytes}}_f \cdot c^{\text{USD}}_{\text{byte-block}}$	$\propto s^{\text{bytes}}_f$	USD/block
Instantaneous Emission-Weighted Replication	$\overline{R}_{\text{inst}}(t) \overset{\text{def}}{=} \frac{\sum_{f \in F(t)} (\omega_f \cdot \lvert N_f \rvert)}{\Omega(t)}$	Weighted average	nodes
Total Bytes Stored	$S_{\text{bytes}}(t) \overset{\text{def}}{=} \sum_{f \in F(t)} s^{\text{bytes}}_f$	$\propto \lvert F(t) \rvert \cdot \overline{s}_{\text{bytes}}$	bytes
Total Sectors Stored	$S_{\text{sectors}}(t) \overset{\text{def}}{=} \sum_{f \in F(t)} s_f$	$\propto \lvert F(t) \rvert \cdot \overline{s}$	sectors