

Research article

DOI: <https://doi.org/10.18721/JCSTCS.19311>

UDC 004.9:004.94



MULTI-LAYER HASH ANCHORING METHOD WITH WATERFALL CONSENSUS FOR FACT-ORIENTED BLOCKCHAIN

D.D. Razuvaev 

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

 Razuvaev_DD@mail.ru

Abstract. The paper proposes a method based on the ideas of classical Proof-of-Work (PoW) for building a distributed registry (Multi-Layer Hash Anchoring, MLHA) based on redundancy of PoW block headers. Each block contains links to the last blocks of all layers, determined by the number of leading hash zeros, which allows you to highlight the sparse skeleton of the waterfall chain. The properties of the waterfall are investigated: uniqueness, determinism, and logarithmic growth of average length. The criterion for the safe deletion of blocks not included in the waterfall is proved, and the possibility of a trustless audit of deleted data is shown. A consensus mechanism based on a lexicographic comparison of waterfall vectors is proposed, consistent with the principle of maximum performance. A single security window has been introduced linking the network hashrate, the minimum mining complexity and the allowable probability of an attack; an upper bound on this probability has been obtained. The simulation results confirm the theoretical estimates. The method does not require a trusted installation, has post-quantum stability and can be used in lightweight decentralized systems.

Keywords: Proof-of-Work, blockchain, data compression, logarithmic growth, consensus, waterfall, MLHA

Citation: Razuvaev D.D. Multi-Layer Hash Anchoring method with waterfall consensus for fact-oriented blockchain. Computing, Telecommunications and Control, 2026, Vol. 19, No. 3, Pp. 130–147. DOI: 10.18721/JCSTCS.19311

Научная статья

DOI: <https://doi.org/10.18721/JCSTCS.19311>

УДК 004.9:004.94



МЕТОД МНОГОСЛОЙНОЙ ПРИВЯЗКИ ХЕШЕЙ С ВОДОПАДНЫМ КОНСЕНСУСОМ ДЛЯ ФАКТО-ОРИЕНТИРОВАННЫХ БЛОКЧЕЙНОВ

Д.Д. Разуваев Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация✉ Razuvaev_DD@mail.ru

Аннотация. В работе предложен базирующийся на идеях классического Proof-of-Work (PoW) метод построения распределенного реестра (Multi-Layer Hash Anchoring, MLHA), основанный на избыточности заголовков блоков PoW. Каждый блок содержит ссылки на последние блоки всех слоев, определяемых количеством ведущих нулей хеша, что позволяет выделить разреженный скелет цепочки – водопад. Исследованы свойства водопада: единственность, детерминированность и логарифмический рост средней длины. Доказан критерий безопасного удаления блоков, не входящих в водопад, и показана возможность бездоверительного аудита удаленных данных. Предложен механизм консенсуса на основе лексикографического сравнения водопадных векторов, согласованный с принципом наибольшей работы. Введено единое окно безопасности, связывающее хешрейт сети, минимальную сложность майнинга и допустимую вероятность атаки; получена верхняя граница этой вероятности. Результаты имитационного моделирования подтверждают теоретические оценки. Метод не требует доверенной установки, обладает постквантовой стойкостью и может найти применение в легковесных децентрализованных системах.

Ключевые слова: Proof-of-Work, блокчейн, сжатие данных, логарифмический рост, консенсус, водопад, MLHA

Для цитирования: Razuvaev D.D. Multi-Layer Hash Anchoring method with waterfall consensus for fact-oriented blockchain // Computing, Telecommunications and Control. 2026. T. 19, № 3. С. 130–147. DOI: 10.18721/JCSTCS.19311

Introduction

The object of the study is the Fact-Oriented Blockchain (FOB), a distributed ledger that stores atomic facts (indivisible, self-sufficient units of information) in a chronological chain of blocks secured under the Proof-of-Work (PoW) concept [1]. Blockchain technologies are used in medicine [2], agronomy [3], and numerous other sectors of critical importance to government and industry. Traditional PoW blockchains require full nodes to store the entire block history, a volume that grows monotonically [4]. This steadily reduces the reach of users with the technical means to maintain a node, thereby threatening decentralization.

The problem is compounded by critical limitations in existing approaches [5]. For instance, pruning, while widely adopted, results in the irretrievable loss of historical block bodies across the entire network. Although the header chain preserves cryptographic coherence, the complete erasure of the source data makes it impossible for nodes to physically prove or verify the authenticity of the history from the genesis block itself. At the same time, compact proof protocols such as Non-Interactive Proofs of Proof-of-Work (NiPoPoW) [6] only facilitate synchronization of light clients but do not reduce storage requirements on full nodes, nor address consensus-level problems, preservation of auditability for erased data, or node synchronization within the proposed paradigms. Thus, until now,

the problem of building a PoW blockchain remains open, in which each full node stores data growing sublinearly (ideally logarithmically) with the total number of blocks created, while retaining the capacity for a complete trustless audit of any historical fact using only hash functions.

The problem of uncontrolled data growth and auditability attracted the attention of many researchers, giving rise to a variety of methods. Each of them, however, has fundamental limitations [7–9] in the context of the task at hand.

Classical pruning and sharding methods are designed to reduce the load on individual nodes [10]. Pruning, however, destroys the local completeness of historical data, making it impossible to audit remote transactions offline without re-requesting them again from other network participants. Sharding distributes data across nodes but does not enable each validator to audit the full history, significantly complicates cross-shard interaction and reduces the security of individual segments (shards). Neither approach allows an individual node to maintain complete independence when verifying history without linear accumulation of data.

Compact proof protocols such as NiPoPoW and FlyClient use probabilistic block samples and Merkle structures to create super-light clients that can verify consensus with a logarithmic amount of transmitted data. However, these protocols address the problem exclusively for light clients: full nodes are still required to store the entire linear volume of blocks, and auditing a specific historical transaction requires a request to a full node, which does not eliminate the storage problem on the proof source itself.

Approaches based on zero-knowledge recursive proofs (Incremental Verifiable Computation, IVC) compress the proof of correctness of the entire history of state transitions to a constant size (for example, the Mina blockchain). However, this architecture does not address data availability for independent audit of historical facts. A mathematical proof guarantees only the validity of the current state of the network, but to confirm the existence of a particular old block or transaction, an archive node is still required storing the original transaction history. Without access to these historical block bodies, a node is unable to extract and audit facts locally. In addition, most compact zk-SNARK systems are based on elliptic curves vulnerable to quantum attacks [8, 11], which imposes long-term security risks compared with systems that rely solely on cryptographic hash functions.

Editable blockchains based on chameleon hash functions (for example, PADHA [1]) permit history modification given a special key. This requires trusted key management, which contradicts the principles of full decentralization, and does not resolve the problem of complete data deletion without losing verifiability.

Thus, all known approaches either sacrifice auditability for compression, require trusted parties, or fail to reduce the load on a full node. None of the existing solutions simultaneously offers the following properties:

- a full node stores data that grows sublinearly or logarithmically with the total number of blocks created;
- trustless cryptographic audit of any historical fact without external compact proofs;
- complete decentralization of consensus without trusted keys or installations;
- post-quantum persistence based solely on standard hash functions.

The identified gap [12] necessitates the development of a fundamentally new method of organizing the blockchain, one that exploits the structural redundancy inherent in PoW to construct a sparse, but cryptographically complete chain skeleton that can replace the full history with minimizing losses in PoW.

Taking into account the gap identified in the literature review – the lack of a method for organizing a PoW blockchain that simultaneously provides logarithmic growth of stored data on full nodes, trustless audit of any historical fact, complete decentralization of consensus, and post-quantum stability based on standard hash functions – the aim of this study is to develop and formally justify a new

paradigm for building a distributed registry, called the Multi-Layer Hash Anchoring (MLHA) method with waterfall (skeleton) consensus.

To achieve this aim, the following objectives are addressed:

1. To construct a formal MLHA model that includes:
 - 1.1. A definition of layers based on the number of leading zeros in a block's PoW hash, together with an analysis of their probability distribution depending on the base of the number system L ;
 - 1.2. A block structure with redundant references $\text{prevHash}[0...N]$ and rules for constructing new blocks that ensure all previous layers remain connected.
2. To introduce the concept of a skeletal waterfall (WF) as a minimal cryptographically connected chain skeleton and strictly prove its key properties:
 - 2.1. uniqueness;
 - 2.2. determinism;
 - 2.3. logarithmic growth of the average WF length from the total number of blocks created;
 - 2.4. monotonic non-decrease of the lexicographic vector of layers over time.
3. To develop an algorithm for secure block deletion based on the invariant of continuity of neighbors after deletion, and show that, provided the deletion criterion is satisfied, complete auditability is preserved: any deleted block can be unambiguously verified through a continuous chain of blocks to the nearest WF block on the right without involving external cryptographic evidence.
4. To design a consensus protocol based on lexicographic comparison of WFs and justify its resistance to attacks of reorganization and forgery of historical references, assuming an honest majority.
5. To perform a simulation to confirm the logarithmic growth of WF.

The solution of these problems forms the MLHA theory, which is expected to overcome the fundamental contradiction between data compression and trustless auditing in decentralized PoW systems.

Materials and methods

Formal MLHA model

The MLHA method is based on using the natural heterogeneity of PoW results to build a distributed registry with controlled growth. The basic cryptographic primitive is a standard hash function $H: \{0,1\}^* \rightarrow \{0,1\}^{512}$ (since Secure Hash Algorithm 512), modeled as a random oracle. The hash length in hexadecimal representation is 128 characters or 512 bits in binary. For applications that require a different balance between the compression ratio and the size of service information, an arbitrary base of the number system L and any cryptographically strong hash function can be used, where L is a power of two ($L = 2, 4, 8, 16, \dots$). In this paper, all statements are formulated for the general case, and, if necessary, they are specified, for example, to $L = 2$, which corresponds to the binary representation of the hash, or $L = 16$, which corresponds to the usual form of hash function results.

Definition 1: Block layer. Let $h = H(m)$ be the hash of the block represented in the number system with the base L . The block layer $\ell \in \{0, 1, \dots, N\}$ (where $N = 512/\log_2 L$ for SHA-512) is the number of leading zero digits in this representation:

$$\ell = \max \left\{ k \in [0, N] \mid \forall i \in [0 \dots k-1]: h[i] = 0 \right\}.$$

In other words, ℓ is the length of the maximum hash prefix consisting of zeros only.

The probability distribution of the layer value ℓ for a random hash obeys the geometric distribution law. For the base of the number system L , the probability that the block belongs to layer k is:

$$P_r(\ell = k) = \begin{cases} ((L-1)/L) \cdot (1/L)^k, & 0 \leq k < N \\ (1/L)^N, & k = N \end{cases}.$$

This distribution is a key for subsequent analysis of WF properties and consensus security.

Definition 2: MLHA block structure. Block B in the MLHA system is a tuple:

$$B = (h, prev, data, t, \eta),$$

where $h = H(prev||t||\eta||data)$ is the cryptographic hash of the block; $prev = (prev[0], prev[1], \dots, prev[N])$ is an array of length $N+1$ containing references to the last blocks of each layer; the $prev[j]$ element stores the hash of the block that was the last block of layer j at the time of creating this block (or null if this block didn't exist yet); $data$ is the set of atomic facts included in the block; t is the timestamp; η is a nonce used to achieve the required PoW complexity.

Constructing a new block. Let B_{prev} be the last block of the chain at the time of mining. The new B_{new} block is constructed as follows:

1. A copy of the $prev$ array from B_{prev} is created.
2. In this copy, the element $prev[\ell]$, where $\ell = \text{layer}(B_{prev})$, is replaced with $\text{hash}(B_{prev})$.
3. The current timestamp t and the initial value $n = 0$ are set.
4. The hash $h = H(prev||t||\eta||data)$ is calculated. If the resulting hash satisfies the network complexity (has at least D leading zeros), the block is considered ready; otherwise, η changes (for example, it is incremented) and the hash is recalculated until the condition is reached.

Thus, each new block “remembers” the last blocks of all layers at the time of its creation, while updating the reference for the layer of its immediate predecessor. Link redundancy is a central element of MLHA, which provides the ability to restore connectivity later after deleting a part of the history.

Initial block validation. Block B is considered valid if the following conditions are met:

1. $h = H(prev||t||\eta||data)$ is the hash that matches the calculated one.
2. Hash h satisfies the current PoW complexity requirement ($\ell \geq D$).
3. If the block is not a genesis block, then it must be constructed based on the specific previous block B_{prev} immediately preceding it in the chain. For such a block, the equality $prev[\ell(B_{prev})]$ must be $= \text{hash}(B_{prev})$, where $\ell(B_{prev})$ is the layer of the previous block. In other words, each non-genesis block must explicitly refer to its direct predecessor through the cell of the $prev$ array that corresponds to the layer of this predecessor.
4. The timestamp in the block is larger than the timestamp in the previous block.
5. If the block is located in the temporary security window and the block is not extreme to the border of the security window on the left, then it is verified whether all links from the predecessor are inherited correctly.
6. The content of blocks does not contradict the rules of a specific FOB instance (verified by an external block validation function. Allows you to customize the FOB for specific use-cases).

These conditions define the protocol for creating and first verifying blocks in MLHA. The next section is devoted to selecting and analyzing the WF chain skeleton, which is formed based on the entered layers and validation of the entire chain.

WF concept and properties

The redundant $prev$ link introduced in the MLHA block structure makes it possible to form an MLHA “skeleton” (WF) in any MLHA chain – a skeleton that preserves cryptographic connectivity with all blocks that have ever existed while growing at a rate significantly slower than that of the full history. Conceptually, the WF comprises exactly those blocks that cannot be deleted without losing the ability to restore the full history, and which at the same time serve as the minimal carrier of PoW for the entire chain. Formally, whether a block belongs to a WF is determined using the *isIndexInWaterflow* predicate ($WF(i)$), which implements a simple rule: a block remains in the skeleton if no later block has a strictly larger layer.

Definition 3: WF.

Let us assume a chain of blocks $B = (B_0, B_1, \dots, B_{m-1})$ that is valid according to the MLHA rules. For the index $i \in \{0, \dots, m-1\}$, we define the predicate WF:

$$WF(i) = \begin{cases} \text{true}, & \nexists j > i : \ell(B_j) > \ell(B_i) \\ \text{false}, & \exists j > i : \ell(B_j) > \ell(B_i) \end{cases},$$

where $\ell(B)$ is the layer of block B .

The WF of the chain B is the set of indices $W = \{i \in [0, m-1] \mid WF(i) = \text{true}\}$. Blocks with indices from W are WF blocks (B_W).

Lemma 1: WF criteria and properties.

For any blockchain built in accordance with the MLHA protocol, the following statements are true:

1. WF criteria. A block B_i is a WF block if and only if, among all blocks with indices $j > i$, there is not a single block with a layer strictly larger than $\ell(B_i)$, i.e., $WF(i) \Leftrightarrow \max_{j > i} \ell(B_j) \leq \ell(B_i)$.
2. Boundary blocks. Genesis-block B_0 and the most recent block B_{m-1} always belong to the WF: $B_0 \in WF$ and $B_{m-1} \in WF$.
3. Uniqueness and determinism. For a fixed chain B , the $WF(B)$ is defined unambiguously and does not depend on the order in which blocks are checked.

Statement 1 (WF criterion) is a direct consequence of the definition of the predicate WF.

Statement 2 on boundary blocks can be proved by the inverse method:

The genesis block. Let us assume that $WF(0) = \text{false} \rightarrow$ by the WF criterion, there must be an index $j > 0$, where $\ell(B_j) > \ell(B_0)$. However, the genesis block by definition has the maximum possible layer N (in binary representation, since its hash is artificially set to N zeros. No other block can have a layer greater than N (exceeding the hash bit limit). Hence, $\ell(B_j) > \ell(B_0)$ is impossible for any j . Contradiction, $\rightarrow WF(0) = \text{true}$.

The last block. Let us assume that the last block B_{m-1} is not included in the WF, i.e., $WF(m-1) = \text{false}$. Then there must exist an index $j > m-1$ with $\ell(B_j) > \ell(B_{m-1})$. However, the set of indices $j > m-1$ is empty, such a j does not exist. Contradiction, $\rightarrow WF(m-1) = \text{true}$.

Statement 3 on uniqueness and determinism is also proved by the inverse method:

Let us assume that the WF is not unique, i.e., there are two different sets $W_1 \neq W_2$, each of which satisfies the definition of a WF (it consists of indices whose blocks do not have blocks of a strictly larger layer on the right). Without limiting generality, we find the index $i \in W_1 \setminus W_2$. Since $i \notin W_2$, by definition there is an index $j > i$ such that $\ell(B_j) > \ell(B_i)$. But then the condition “there is no strictly larger layer on the right” for index i is violated, and i cannot belong to W_1 . This contradicts the choice of i . Similarly, we consider the case $i \in W_2 \setminus W_1 \rightarrow$ our assumption about the existence of two different WFs is false $\rightarrow$ WF is defined in a unique way and does not depend on any external factors, except for an ordered sequence of layers of blocks.

Next, we consider the size of WF and its asymptotic behavior.

Lemma 2: Logarithmic growth of the average WF length.

Let the block chain be built in accordance with the MLHA protocol. T is the total number of blocks created (including genesis), and L is the base of the hash number system. Then the mathematical expectation of the WF length $|W_T|$ satisfies the relation:

$$\mathbb{E}[|W_T|] = (L-1) \log_L T + 1 + o(1), \quad (T \rightarrow \infty).$$

In particular, for the binary representation ($L = 2$) at $T \rightarrow \infty$, the estimate $\mathbb{E}[|W_T|] \approx \log_2 T + 1$ is valid, and for the hexadecimal representation ($L = 16$) – $\mathbb{E}[|W_T|] \approx 15 \log_{16} T + 1$.

Consider a probabilistic model in which the hashes of all blocks are independent and evenly distributed in $\{0, 1\}^{512}$. The layer $k \in \{0, 1, \dots, N\}$ appears with probability:

$$p_k = \frac{L-1}{L} \cdot \frac{1}{L^k} (k < N), \quad p_N = \frac{1}{L^N}.$$

For $k < N$, the distribution function has the form $S_k = \Pr(\ell \leq k) = 1 - (1/L)^{k+1}$.

Genesis-block B_0 has a fixed maximum layer N and always enters the WF, adding +1 to its length.

Let us now consider the $T-1$ subsequent blocks $B_1, \dots, B_{T-1}$. A block B_i ($1 \leq i \leq T-1$) with a layer k falls into a WF if and only if none of the blocks $B_{i+1}, \dots, B_{T-1}$ has a layer strictly larger than k . For block B_i , there are $T-1-i$ blocks after it. The probability that they all have a layer $\leq k$ is $(S_k)^{T-1-i}$.

The mathematical expectation of the number of WF blocks in layer k among non-genesis blocks in the chain is:

$$E_k = p_k \sum_{i=1}^{T-1} (S_k)^{T-1-i} = p_k \sum_{m=0}^{T-2} (S_k)^m = p_k \left(\frac{1 - S_k^{T-1}}{1 - S_k} \right).$$

The last block B_{T-1} is always a WF block, with the multiplier $S_k^0 = 1$.

For layer N , the probability of p_N is negligible, and its contribution can be taken into account separately, but it does not change the asymptotes.

Thus, the full mathematical expectation of the WF length is:

$$\mathbb{E}[|W_T|] = 1 + \sum_{k=0}^{N-1} \frac{p_k}{1 - S_k} (1 - S_k^{T-1}) + \varepsilon_T,$$

where $\varepsilon_T \rightarrow 0$ for $T \rightarrow \infty$. Note that for all $k < N$, $\frac{p_k}{1 - S_k} = L - 1$ holds, since

$$\frac{p_k}{1 - S_k} = \frac{\frac{L-1}{L} \cdot \frac{1}{L^k}}{\frac{1}{L^{k+1}}} = L - 1.$$

For large T , the value of S_k^{T-1} decreases exponentially for all k for which $S_k < 1$, i.e., for $k < N$. For real T , layers with $k \geq \log_L T + c$ (for a certain constant c) give a negligible contribution, and the summation can be limited to the index $K_{\max} \approx \log_L T$. Let us estimate the sum $\sum_{k=0}^{N-1} (L-1)(1 - S_k^{T-1})$ asymptotically.

$K = \lfloor \log_L T \rfloor$. Let us split the sum into two parts:

$$\sum_{k=0}^{N-1} (L-1)(1 - S_k^{T-1}) = \sum_{k=0}^K (L-1)(1 - S_k^{T-1}) + o(1).$$

For $k \leq K$, we have $S_k = 1 - L^{-(k+1)}$, and $(1 - S_k^{T-1}) = 1 - (1 - L^{-(k+1)})^{T-1}$. For $T \gg L^{k+1}$, this expression is close to 1, and for $T \ll L^{k+1}$, it is close to 0. The boundary of the “maturation” of layer k is at $T \approx L^{k+1}$, i.e., $k \approx \log_L T - 1$. Hence, the sum $(L-1)(1 - S_k^{T-1})$ over k from 0 to K asymptotically behaves as $(L-1)$

multiplied by the number of layers for which $T \gg L^{k+1}$, plus the contribution from the transition layer. More strictly, using integral estimation, we can show that:

$$\sum_{k=0}^K (1 - S_k^{T-1}) = \log_L T + O(1), \quad (T \rightarrow \infty).$$

The mathematical expectation of the number of records in a sequence of independent random variables with a geometric distribution grows as $\log_L T$. Hence:

$$\mathbb{E}[|W_T|] = 1 + (L-1) \log_L T + O(1) = (L-1) \log_L T + 1 + o(1).$$

This completes the proof.

The above proof assumes that there is no minimum complexity. In a practical protocol with $D > 0$, the layer distribution is truncated from below: blocks with layer $k < D$ are not created. This leads to the fact that the mathematical expectation of the WF length is smaller, asymptotically preserving the form $(L-1) \log_L T + O(1)$. Thus, the lemma 2 gives an upper bound for the real system, which only strengthens the statement about logarithmic compression.

This reasoning demonstrates the scalability property of MLHA: the average size of the stored WF chain grows only logarithmically with the total number of blocks created, which is also confirmed experimentally.

Secure data deletion and audit

One of the central advantages of MLHA is the ability to selectively delete blocks that are not part of the WF, without losing the connectivity of the remaining chain and without compromising the ability to audit deleted data. This section introduces the safe deletion criterion and proves its correctness, as well as sets the auditability property for deleted blocks.

Let the chain $B = (B_0, B_1, \dots, B_{m-1})$ be valid according to the MLHA rules, and let i be the index of some block that is supposed to be deleted. $B_{Left} = B_{i-1}$ is the immediately preceding block (left), and $B_{Right} = B_{i+1}$ is the immediately following block (right). We restrict the boundary values i : $i \neq 0$ and $i \neq m-1$.

Definition 4: Secure deletion.

Deleting a B_i block is considered safe if all MLHA validation rules are successfully executed for each remaining block after it is removed from the chain.

Lemma 3: Safe deletion criteria.

Deleting the B_i block is safe if and only if the following conditions are met:

1. B_i is not a WF block, i.e., $WF(i) = \text{false}$.
2. B_{Right} block in its reference array in the hash slot of the B_{Left} block contains the hash of the B_{Left} block, i.e., $B_{Right}.prev[\ell(B_{Left})] = \text{hash}(B_{Left})$.
3. Deletion takes place outside the security windows (it will be considered when determining the validity of the MLHA chain).

If these conditions B_i are met, the chain remains valid after deleting B_i .

Proof:

The proof is carried out using the opposite method, establishing the necessity and sufficiency of each of the conditions.

Assume that deleting block B_i is safe, i.e., after it is excluded, the chain $B' = (B_0, \dots, B_{i-1}, B_{i+1}, \dots, B_{m-1})$ satisfies all MLHA validation rules.

- If condition 1 was violated, i.e., $WF(i) = \text{true}$, then block B_i would belong to the WF. By lemma 1, the WF of a chain is uniquely determined, and removing any element of it would destroy the uniqueness and determinism property of the WF for the chain, which would mean losing the full auditability property required from safe deletion. Hence, $WF(i) = \text{false}$.

- Suppose condition 2 was violated, i.e., $B_{Right}.prev[\ell(B_{Left})] \neq \text{hash}(B_{Left})$, however, after removing B_i block, the B_{Right} block must refer to B_{Left} as its immediate predecessor. However, according to the rules of primary block-level validation (see definition 2), each non-genesis block must contain *prev* the hash of its direct predecessor in $prev[\ell(\text{predecessor})]$. Since B_{Left} becomes the precursor of B_{Right} in B' , $B_{Right}.prev[\ell(B_{Left})] = \text{hash}(B_{Left})$ is necessary for B_{Right} to be valid, which contradicts the assumption that the condition is violated. So, condition 2 is true.

- Condition 3 (deletion outside security windows) will be explained later in the article.

Let conditions 1–3 be met. After deleting B_i , the chain B' differs from the valid chain B only by the absence of an element with index i . For each block $B_j \in B', j \neq i+1$, the *prev* reference array and hash of the previous block have not changed, and each block continues to satisfy the validation rules. It remains to check the block B_{i+1} (now immediately following B_{i-1}). Condition 2 holds $B_{i+1}.prev[\ell(B_{i-1})] = \text{hash}(B_{i-1})$, i.e., B_{i+1} correctly references the new predecessor through the required slot. Therefore, in the aggregate, B' is valid.

Corollary 1: Neighbor continuity invariant:

If you repeatedly apply safe deletion within the safe deletion criterion in any valid order, the resulting chain always retains the property: for any of its elements B_k (except B_0), B_k is executed $B_k.prev[\ell(B_{k-1})] = \text{hash}(B_{k-1})$, where B_{k-1} is the immediately preceding block in this resulting chain. In other words, the chain remains continuous in the sense of hash links connecting each block to its left neighbor through the layer of the latter.

Corollary 2: Auditability of data:

Let the block B_i be safely removed from the network, and let the local node B_i not remove block B_i . Then the local node has a sequence B' of locally preserved blocks, starting with B_i and ending with some WF block on the right (the WF of the local node is synchronized with the WF of the network), such that each subsequent block of the sequence B' contains in its *prev* array the hash of the previous one. This sequence can be found by using the optimal *prev* link traversal mechanism, resulting in a continuous chain with the beginning at B_i and the end B of the WF block closest to B_i on the right. Since WF blocks are stored by all full nodes of the network, any node that receives the sequence B' can independently verify the authenticity of B_i by sequentially calculating the hashes of the chain B' from B_i until the result of the hash is verified with the *prev hash of the WF block*. No external cryptographic proofs are required, except for the standard hash function used in the protocol.

Thus, lemma 3 and its consequences form the foundation on which to build: the ability to customize the cleaning policy (and therefore customize the individually defined data lifecycle at the application level) within the criteria of safe cleaning; data audit mechanisms; mechanisms for distributed storage of specific data blocks in specific MLHA nodes at the level of top-level requirements, etc.

Condition 3 regarding security windows will be integrated into the overall chain validation procedure at the network consensus level, which will complete the holistic definition of the secure deletion model.

Consensus protocol based on the lexicographic order of WF vectors

The proposed MLHA model provides a unique feature: a complete node stores only a WF chain skeleton, the size of which grows logarithmically, and yet it is able to independently and reliably select a canonical chain among several competing candidates. The central element of such a consensus is a strict lexicographic order relation on the set of WF vectors, which, as will be shown below, is monotonous with respect to adding new blocks and induces a preference equivalent to the classical principle of “greatest expected performance”.

Definition 5: WF vector.

Let B be a valid MLHA chain, $WF(B) = (B_0, \dots, B_{s-1})$ be its WF, where $i_0 = 0$ (genesis), $i_{s-1} = m-1$ (last block), and layers of WF blocks $\ell_t = \ell(B_t)$. The WF vector is the ordered set $v(B) = (\ell_0, \ell_1, \dots, \ell_{s-1})$.

By construction (lemma 1), the genesis always has a maximal layer N , and $v(B)$ is uniquely determined by the chain.

Definition 6: Lexicographic order.

For two vectors $u = (u_0, \dots, u_{a-1})$ and $v = (v_0, \dots, v_{b-1})$, we set $u < v$ if there exists an index $j < \min(a, b)$ such that $u_i = v_i$ for all $i < j$, and $u_j < v_j$, or if $a < b$ and $u_i = v_i$ for all $i < a$. Accordingly, $u \leq v$ means $u < v$ or $u = v$.

The algorithm for selecting the canonical max chain (hereinafter referred to as the max algorithm) for two valid chains A and B works as follows:

1. Select $WF(A)$ and $WF(B)$ and plot their vectors v_A and v_B .
2. If $v_A < v_B$, return B ; if $v_B < v_A$, return A .
3. If $v_A = v_B$ (the lengths and layers are exactly the same), to ensure determinism, compare the numerical values of the hashes of the first different blocks of chains (for example, as 512-bit integers); return the chain with a larger hash value.

Lemma 4: strict monotony of the WF vector

Let B be a valid MLHA chain, and B' is obtained from B by adding one new block B_{new} (followed by implicit updating of the set WF according to definition 3). Denote by $v = v(WF(B))$ and $v' = v(WF(B'))$ layer vectors of WF block chains B and B' , respectively. Then $v' > v$.

Proof:

Let $K = \ell(B_{new})$ be the layer of the new block. Consider how the WF changes after enabling B_{new} and redefining the WF set according to the criterion (no block with a strictly large layer on the right). There are two possible cases:

1. The new B_{new} block does not exclude other blocks from the WF . This means that after updating the WF , all blocks that were part of $WF(B)$ retain their status. Since $WF(B')$ contains all the elements of $WF(B)$ and another block B_{new} (located to the right of all the previous ones), the vector v' is obtained from v by appending the element K to the end. Thus, $|v'| = |v| + 1$ for all indices $i < |v|$ completed $v'_i = v_i$. By the definition of lexicographic order (the second condition in definition 6), $v' > v$.
2. The new block excludes $m \geq 1$ of the last WF blocks. Let $v = (v_0, v_1, \dots, v_{s-1})$ be the source vector, and let r be the index of the first of the excluded blocks ($0 \leq r \leq s-1$), so that the blocks $B_r, B_{r+1}, \dots, B_{s-1}$ cease to exist be a WF . By construction, all these blocks have layers strictly smaller than K : otherwise, a block with a larger or equal layer could not be excluded, since there would be no block to the right with a layer larger than its own. Hence, $v_r < K, v_{r+1} < K, \dots, v_{s-1} < K$. The new WF vector takes the form $v' = (v_0, v_1, \dots, v_{r-1}, K)$. Compare v and v' lexicographically. The first r elements match: $v'_0 = v_0, \dots, v'_{r-1} = v_{r-1}$. At position r , we have $v'_r = K > v_r$. Since the vector v' has a higher value on the first element that differs, the strict inequality $v' > v$ is satisfied.

Both cases exhaust possible changes to the WF when adding a new block. In all cases $v' > v$. The lemma is proved.

Corollary 3: Consistency with the classical PoW principle.

Lemma 4 states that when a new block is added, the WF vector of the chain progresses strictly lexicographically. Thus, the max algorithm will never prefer a shorter chain. Now we show that such a lexicographic order is in full agreement with the PoW fundamental principle: the chain representing the largest amount of computational work is recognized as canonical.

Each WF block of layer k is an objective proof that on average L^k hash operations were spent to create it. Although the specific sum $\sum L^{\ell(B_i)}$ along the WF does not have to be monotonic (statistical fluctuations can lead to the fact that a chain of many low layers formally gives a larger sum than a chain with one very high layer), the lexicographic order on the vectors adequately reflects exactly the expected costs. The reason is that a higher layer at the first different position means an exponentially rarer event that requires an exponentially larger number of attempts, and the probability that a chain

with such a vector was built with less total effort decreases rapidly with the growth of the layer difference. Consequently, of the two competing chains, the one with the lexicographically larger WF vector was overwhelmingly likely to require more computational resources.

Thus, the MLHA consensus inherits the main property of the classic PoW: an honest chain supported by the majority of computing power will have the lexicographically largest WF vector in the long run and will therefore be accepted by all nodes of the network.

Hashrate estimation via WF

Since each WF block of layer k requires on average L^k hash operations, the value $\sum L^{\ell(B_i)}$ calculated from a sufficiently long sliding time window Δt , which includes N WF blocks (each block has a timestamp), can be considered as an estimate of the expected work performed by the network for the corresponding time interval Δt . This estimate, although not strictly unbiased, can serve as an indicator of the current hashrate for the purpose of dynamically adapting the minimum mining complexity, especially when normalized to the average fraction of WF blocks or when used in relative (differential) mode. Dividing this amount by Δt , we get a practical estimate of the current network hashrate:

$$\hat{H} = \frac{1}{\Delta t} \sum_{i=1}^N L^{\ell(B_i)}.$$

This estimate relies exclusively on WF, does not require storing remote blocks, and naturally averages random fluctuations, making it a convenient tool for dynamically adapting security parameters.

Threat model and security parameterization

The WF and lexicographic consensus properties set out above ensure that with a fair majority, the canonical chain accumulates the most expected work. However, in a real decentralized network, it is necessary to additionally protect against two fundamentally different classes of attacks that exploit the features of the MLHA architecture: immediate link forgery attacks and deep reorganization attacks.

Attack vector I: link forgery with instant deletion.

The attacker creates a B_{adv} block, in which he intentionally specifies incorrect values in the *prev-Hash* fields for low layers, and distributes it over the network. As the last block of the chain, B_{adv} temporarily enters the WF and can be accepted by nodes if its layer allows you to win the race. Malicious links are not checked by the consensus protocol, because they belong to layers below the block layer itself. If a new block that intercepts the WF appears shortly after, the B_{adv} is no longer WF and can be erased from local storage according to the safe deletion criterion. As a result, history auditing becomes impossible for some blocks, because links to deleted blocks are invalid.

Attack vector II: deep reorganization (selfish mining).

The attacker secretly builds an alternative chain, starting with a block located much deeper than the current tip of the honest chain. Over a long period of time, he can accumulate a WF vector that exceeds the honest one if he managed to find blocks with abnormally high layers. When publishing such a fork, nodes that use the max algorithm may be forced to switch to it, rolling back already confirmed facts and breaking the stability of the registry.

Solution:

To neutralize attack I, it is necessary that the block remains available for verification by all nodes for some time before being deleted. During this period, each node of the network, when receiving a candidate chain, must check the correctness of *prev data* within the framework of the *prev* array copying principle in window A . Deleting any block will automatically break the validation of the principle of copying *prev data* from block to block.

To neutralize attack II, it is necessary to limit the maximum depth of automatic reorganization: a node must reject forks that have a common ancestor with the local chain older than a certain threshold T .

These two solutions, at first glance, contradict each other: for the reliability of the audit, it is desirable to increase A , while to minimize the damage from reorganizations, it is desirable to reduce T . However, both defense mechanisms, A and T , regulate the same fundamental parameter: the time during which the system remains vulnerable to attacks based on its instability. Auditing requires that the block is online long enough before it can be deleted; preventing deep reorganizations requires that the block cannot be undone after it has already been verified. Therefore, it is technically justified to put $A = T = \Delta$. The single window Δ defines the instability horizon: all blocks that go beyond the left window Δ are considered finalized – blocks and link validity are verified by the network, and attempts to cancel them are rejected by the switching policy. Inside the window Δ , there is a probabilistic consensus, in which a race at the tip of the chain is acceptable, and any damage from it is limited by the depth Δ .

We can now formulate a central MLHA security statement linking the *network hashrate*, the *security window* Δ , and the required *mining difficulty* D .

Theorem 1: Upper bound on the probability of a successful attack.

Let the attacker control a fraction of $q < 0.5$ of the total network hashrate, and let the honest network have the hashrate H_{hon} . Assume that the minimum mining difficulty is D leading zeros, and the single security window is Δ . Then the probability that an attacker will be able to carry out an attack of the first or second type (i.e., replace historical links or impose a deep reorganization) does not exceed

$$P_{attack} \leq \sum_{k=D}^N \left(1 - e^{-H_{att} L^{-k} \Delta}\right) \cdot e^{-H_{hon} L^{-k} \Delta},$$

where $H_{att} = q \cdot H_{hon} / (1-q)$ is the attacker's hashrate, N is the maximum layer, and L is the base of the number system.

Rationale: Both types of attacks require that the Δ attacker create at least one block with a sufficiently high layer that will give his fork a lexicographic advantage, while an honest network should not create any blocks with a comparable or larger layer. The processes of generating layer blocks $\geq k$ by an attacker and an honest network can be approximated by independent Poisson flows with intensities $H_{att} L^{-k}$ and $H_{hon} L^{-k}$, respectively. Summing over all layers $k \geq D$ the probabilities that an attacker will get at least one success, and an honest network-none, we get the upper bound of the desired probability (union bound).

Corollary 4: Parameterization of minimal complexity.

To ensure that the probability of a successful attack of any type does not exceed the specified threshold P_{max} , the minimum mining difficulty D must satisfy the inequality

$$D \geq \log_L \left(\frac{H_{att} \cdot \Delta}{P_{max}} \right),$$

where $H_{att} = q \cdot H_{hon} / (1-q)$ is the attacker's hashrate.

The formula provides a simple and measurable criterion for configuring the protocol: knowing the current hashrate H_{hon} (estimated through a WF) and choosing an acceptable risk level P_{max} (for example, 10^{-9}), the network can dynamically adjust the required number of leading zeros D , maintaining security even if the attacker uses the optimal strategy of secret mining and timestamp manipulation.

Results and discussion

Comparative analysis with analogs

Table 1 shows that MLHA occupies a unique niche: unlike NiPoPoW/FlyClient, which address the problem only for light clients, MLHA radically reduces the amount of data stored on each full node.

Table 1

MLHA comparison with key approaches to blockchain compression and auditing

Criteria	MLHA criterion (this paper)	NIPoPoW / FlyClient	Mina (IVC, zk SNARK)
Data growth at the full node	$O(\log_L T)$ – logarithmic of the number of blocks created	$O(T)$ (the full node stores the entire history)	$O(1)$ (constant, ~22KB)
Trustless history audit	Yes, through a WF and a continuous chain of headers	Requires requests to the full node and additional Merkle-proofs	No, the audit requires an archive node with intermediate proofs
Post-quantum persistence	Yes (hashes)	Yes (hashes)	None (based on elliptic curves)
Trusted setup	No	No	Universal (require heavy calculations)
Consensus mechanism	WF consensus mechanism (lexicographic comparison of skeletons), equivalent to PoW	Classic PoW (longest chain)	Ouroboros (Proof of Stake)
Compression configuration flexibility	Is high (choice of base L and cleaning policies)	Fixed (defined by the protocol)	Maximum compression (constant), but without the ability to store history

Compared to Mina, MLHA does not require heavy calculations, has post-quantum stability, and retains the ability to independently audit history without external archivists. Thus, MLHA provides simultaneous fulfillment of all four requirements formulated in the introduction: logarithmic growth of stored data, trustless auditing, decentralization, and post-quantum persistence.

Prototype results

For experimental verification of the developed theoretical propositions, a software prototype of MLHA in Java was created using the SHA-512 hash function in binary representation (base of the number system $L = 2$). The simulation was performed on a machine running Arch Linux with an Intel i7-10850H processor (12 cores, up to 5.1 GHz) and 32 GB of RAM. All results were averaged over an ensemble of 1000 independently evolving blockchains, which made it possible to almost completely eliminate random fluctuations and identify stable patterns.

Fig. 1 shows the system's behavior when using the “no more than 1000 blocks” cleanup policy, which allows non-WF blocks to accumulate until the total number of blocks in the chain exceeds the threshold of 1000, after which the safe deletion mechanism (size containment within 1000) is triggered. The abscissa axis shows the total number of created blocks T (from 0 to 11500), and the ordinate axis shows the chain size in blocks. The four curves reflect, respectively:

1. The actual average (over time and over 1000 implementations) number of blocks in the chain.
2. The actual global maximum of the chain size for the entire history.
3. The actual average length of the WF.
4. The theoretical updated (adjusted) average length of the WF.

Fig. 1 shows that the average chain size increases monotonically at the initial stage, after which it asymptotically approaches the 1000 block mark. The global maximum quickly reaches the threshold value and then does not change, which confirms the correct operation of the cleaning policy and the correct restriction of unlimited chain growth. Against this background, the behavior of a WF is particularly remarkable: its average length remains extremely small and does not depend on how many non-WF blocks are accumulated in the chain. The actual and theoretical curves for the WF merge into one line, which clearly demonstrates the high accuracy of the refined analytical model even at relatively short time intervals.

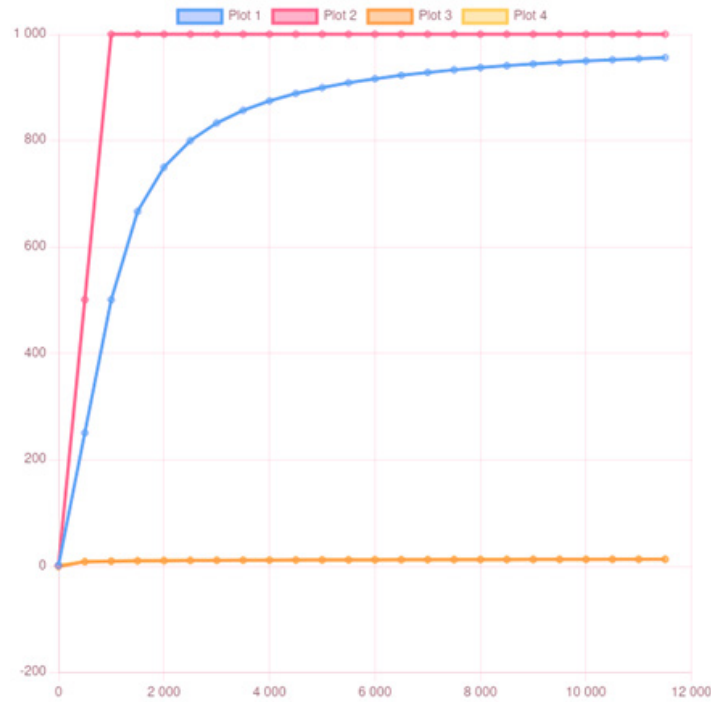


Fig. 1. Dynamics of the chain size under the “no more than 1000 blocks” restriction policy

Fig. 2 shows in detail the scalability of MLHA over the interval up to $T = 20000$ created blocks in the binary number system. Here, only the WF characteristics are plotted along the ordinate axis:

1. The actual average for all blockchains and all time points.
2. Global maximum WF length among 1000 chains.
3. Theoretical approximate value of $\log_2 T$.
4. A theoretical refined value that takes into account +1 and the exponentially decaying tail of the distribution.

The actual average and the refined theoretical curve are indistinguishable; their divergence is tenths of a block over the entire range studied. This is a direct experimental confirmation of lemma 2 on the logarithmic growth of a WF. The approximate formula $\log_2 T$ (removed the tail and 1) gives slightly lower values (especially at small T), which is explained by the neglect of the tail, but with increasing T , it quickly approaches the value of the exact curve -1 .

Special attention should be paid to the global maximum length of the WF. It has a characteristic step structure: each jump up occurs at the moment when one of the 1000 chains “delays” the appearance of a block with a new, higher layer, which should “displace” part of the chain. This step pattern is a visual representation of the probabilistic nature of MLHA consensus and is fully consistent with the predictions of the theory. It is important to note that the global maximum of 1000 chains does not even exceed 50 blocks with 20000 created, which emphasizes the practical compactness of the WF.

Fig. 3 shows the results of the simulation with the “no more than 1000 blocks” policy for the interval up to $T = 635'000$ blocks created. Four curves that characterize both the complete chain and its WF skeleton are combined on one field:

1. The actual average number of blocks in the chain.
2. The actual global maximum of the chain size.
3. The actual average number of blocks in the WF.
4. Theoretical refined value of the number of blocks in a WF, taking into account +1 and the exponentially decaying tail of the distribution.

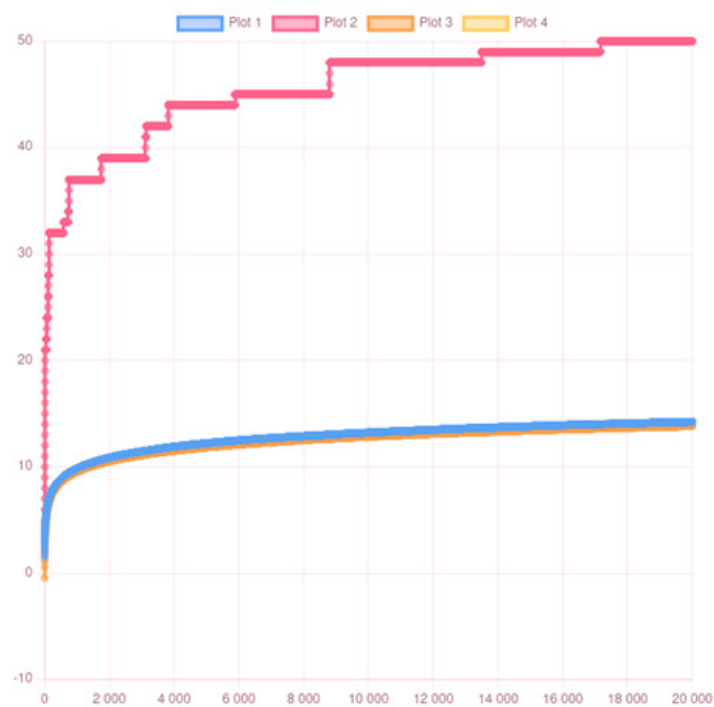


Fig. 2. Logarithmic WF growth and verification of the theoretical estimate



Fig. 3. Comparison of full chain and WF under controlled growth

The behavior of the complete chain (lines 1 and 2) demonstrates the rapid achievement of a near-equilibrium state. After a period of initial linear growth, the average number of blocks stabilizes near the specified threshold of 1000 blocks, and the global maximum, once touching this value, then

remains unchanged. This clearly demonstrates that the cleanup policy effectively limits the size of the chain, preventing its uncontrolled expansion.

Against the background of these two almost horizontal lines, the behavior of the WF (lines 3 and 4) looks particularly revealing. Despite the fact that the full chain constantly contains about 1000 blocks, its skeletal skeleton (WF) remains several orders of magnitude smaller. The actual and theoretical curves for the WF are indistinguishable, which once again confirms the high accuracy of the developed analytical model.

The scale of the graph makes this logarithmic growth of the WF visually almost invisible against the background of the stabilized size of the full chain. And it doesn't make sense to make a comparison chart with the classic PoW at this scale. This is the key advantage of MLHA: the main amount of data stored can be dictated by application policy (in this case, 1000 blocks), but the cryptographic skeleton that is critical for consensus and audit remains exponentially more compact. Even after more than half a million blocks have been created, it is enough to store only a few dozen WF blocks to fully verify and audit the entire network history. A long simulation horizon ($T = 635'000$) and averaging the result over 1000 parallel runs strongly proves that this property is a stable, not short-term effect, and that the system does not degrade over time.

Thus, Fig. 3 serves as a clear demonstration of the effectiveness of the proposed method: it demonstrates how the MLHA architecture allows you to completely separate chain size management from the fundamental requirements of consensus (WF storage), while ensuring strict mathematical predictability of the latter.

Thus, the experimental part of the work fully confirms the viability and scalability of the proposed method. The resulting graphs are a clear illustration of how the structural redundancy of PoW provides an efficient mechanism for controlled chain compression without loss of cryptographic completeness.

Synchronization mechanism and WF block addressing

The proposed WF structure makes it possible to implement an efficient full node bootstrapping procedure and a unified mechanism for addressing arbitrary blocks without storing the full history.

When a node connects to the network, it performs two actions:

1. Gets the current WF from peers.
2. Requests and receives all blocks located on the network within the security window Δ and not included in the WF.

After that, the node is able to independently verify the consensus state and participate in it, storing only $O(\log T)$ WF blocks and a small amount of data for validation, if the node's policy does not allow skipping primary synchronization and compressing the security window Δ .

WF addressing

Since each block B in MLHA is characterized by a layer $\ell(B)$ and a position in the chain, it can be uniquely identified by the vector $\langle c_{\ell_{\max}}, c_{\ell_{\max}-1}, \dots, c_0 \rangle$, where c_k is the sequence number of the block in the sequence of WF blocks of layer k containing B or associated with it. In other words, the address specifies the path in the hierarchy of layers from the rarest (genesis) to the most frequent (layer 0). This method of addressing allows you to search for blocks in logarithmic time, simplifies the work of decentralized auditing, and is itself a naturally routable unit of the network.

Thus, WF not only provides compact storage, but can also serve as a universal mechanism for navigating the entire history of the blockchain.

Conclusion

MLHA offers a unique combination of properties: it makes the blockchain compact for a regular node, but at the same time retains security, post-quantum stability, and does not require creating "master keys" or performing complex cryptographic ceremonies at launch. In other words, consensus-critical data now grows logarithmically, and individual archive nodes can be responsible for storing the full history.

The key scientific result is the development of a fundamentally new method for organizing the blockchain, in which the structural redundancy of PoW itself is used to build a sparse but cryptographically complete chain skeleton that can replace the full history with minimizing losses in PoW. It is shown that the structural redundancy of PoW-hashes can be effectively used to create optimizations both within the framework of the blockchain structure and for consensus algorithms. The novelty of the work is determined by the following main results:

- The concept of WF is introduced. The WF membership criterion, its uniqueness and determinism are proved (lemma 1).
- It is proved that the mathematical expectation of the WF length grows as $O(\log_L T)$, where L is the base of the hash number system, and T is the total number of blocks created (lemma 2). This result establishes the fundamental possibility of exponential compression of the blockchain's history.
- A criterion for secure block deletion (lemma 3) has been developed, which guarantees the preservation of cryptographic connectivity and the ability to audit any deleted block. It is shown that the audit does not require external cryptographic proofs or trusted archivists.
- A consensus protocol based on lexicographic comparison of WF vectors is proposed and its strict monotonicity is proved (lemma 4). This consensus is consistent with the principles of classical PoW (corollary 3).
- A probabilistic threat model covering different attack vectors is constructed. The concept of a single security window linking the network hashrate, minimum mining complexity, and acceptable attack probability is introduced. An upper bound on the probability of a successful attack is obtained (theorem 1) and the minimum complexity parameterization formula is derived (corollary 4).
- A method for estimating the current network hashrate is developed, based solely on WF blocks, which does not require storing the full history.

The results obtained are of fundamental importance for the theory of distributed ledgers, as they demonstrate the possibility of building a PoW blockchain in which each full node stores a sublinear amount of data, without sacrificing either decentralization, audibility of history, post-quantum stability, or processor power. The MLHA method opens up a new research area-structural compression of blockchains based on internal PoW statistics.

From a practical point of view, MLHA can become the basis for creating lightweight decentralized registries in areas such as the Internet of Things (IoT), distributed document management, certification and logging systems, where node resources are limited, but the requirements for long-term data verifiability are high. The ability to adjust the base of the L number system allows you to flexibly adjust the balance between compression ratio and overhead.

REFERENCES

1. **Razuvaev D.D., Ustinov S.M.** Mitigating data growth in PoW blockchains: Storage reduction methods for scalability without compromising decentralization. *Computing, Telecommunications and Control*, 2025, Vol. 18, No. 4, Pp. 87–101. DOI: 10.18721/JCSTCS.18408
2. **Kaveshnikov A.V., Bragin D.S., Vaizov V.Kh., Kaveshnikov V.S.** The possibilities of blockchain technologies application in the healthcare sector. *Siberian Scientific Medical Journal*, 2025, Vol. 45, No. 5, Pp. 78–90. DOI: 10.18699/SSMJ20250507
3. **Pal T., Bommiseti V., Suvarnakanti P., Dudala S.S., Asha S., Jaiswal V.** The revolutionary role of blockchain technology in the agri-food sector focusing on the food supply chain. *Food Reviews International*, 2025, Vol. 41, No. 3, Pp. 770–801. DOI: 10.1080/87559129.2024.2418464
4. **Lingeshwaran G., Raman J.M.H.** Blockchain enhanced IoT devices: Review. *International Journal of Nexus of Knowledge*, 2025, Vol. 1, No. 2.

5. Altarawneh A., Herschberg T., Medury S., Kandah F., Skjellum A. Buterin's Scalability Trilemma viewed through a State-change-based Classification for Common Consensus Algorithms. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, Pp. 0727–0736. DOI: 10.1109/CCWC47524.2020.9031204
6. Miller L., Pacaud D., Deroousseaux-Lebert N., Anceaume E., Ludinard R. Mining in Logarithmic Space with Variable Difficulty. *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, Pp. 888–902. DOI: 10.1145/3719027.3744874
7. Al-Janabi S. Post-Quantum Blockchain: Challenges and Opportunities. *arXiv:2508.17071*, 2025. DOI: 10.48550/arXiv.2508.17071
8. Sezer B.B., Akleylek S., Nuriyev U. PP-PQB: PP-PQB: Privacy-Preserving in Post-Quantum Blockchain-Based Systems: A Systematization of Knowledge. *IEEE Access*, 2025, Vol. 13, Pp. 41382–41405. DOI: 10.1109/ACCESS.2025.3545943
9. Heo J.W., Ramachandran G.S., Dorri A., Jurdak R. Blockchain data storage optimisations: a comprehensive survey. *ACM Computing Surveys*, 2024, Vol. 56, No. 7, Art. no. 179. DOI: 10.1145/3645104
10. Morsidi M., Tajuddin S., Shah Newaz S.H., Patchmuthu R.K., Lee G.M. A Review on Blockchain Sharding for Improving Scalability. *Future Internet*, 2025, Vol. 17, No. 10, Art. no. 481. DOI: 10.3390/fi17100481
11. Shor P.W. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, Pp. 124–134. DOI: 10.1109/SFCS.1994.365700
12. Boughdiri M., Abdellatif T., Guegan C.G. A systematic literature review on blockchain storage scalability. *IEEE Access*, 2025, Vol. 13, Pp. 102194–102219. DOI: 10.1109/ACCESS.2025.3578451

INFORMATION ABOUT AUTHOR / СВЕДЕНИЯ ОБ АВТОРЕ

Daniil D. Razuvaev

Разуваев Даниил Дмитриевич

E-mail: Razuvaev_DD@mail.ru

ORCID: <https://orcid.org/0009-0009-6961-8814>

Submitted: 01.06.2026; Approved: 07.09.2026; Accepted: 16.09.2026.

Поступила: 01.06.2026; Одобрена: 07.09.2026; Принята: 16.09.2026.