future internet

MDPI

Article

Optimizing Merkle Proof Size Through Path Length Analysis: A Probabilistic Framework for Efficient Blockchain State Verification

Oleksandr Kuznetsov [1,2,*], Emanuele Frontoni [3,4], Kateryna Kuznetsova [3,4] and Marco Arnesano [1]

[1] Department of Theoretical and Applied Sciences, eCampus University, Via Isimbardi 10, 22060 Novedrate, Italy; marco.arnesano@uniecampus.it
[2] Department of Intelligent Software Systems and Technologies, School of Computer Science and Artificial Intelligence, V.N. Karazin Kharkiv National University, 4 Svobody Sq., 61022 Kharkiv, Ukraine
[3] Department of Political Sciences, Communication and International Relations, University of Macerata, Via Crescimbeni, 30/32, 62100 Macerata, Italy; emanuele.frontoni@unimc.it (E.F.); kate7smith12@gmail.com (K.K.)
[4] VRAI—Vision, Robotics and Artificial Intelligence Lab, Via Brecce Bianche 12, 60131 Ancona, Italy
* Correspondence: oleksandr.kuznetsov@uniecampus.it or kuznetsov@karazin.ua

Abstract: This study addresses a critical challenge in modern blockchain systems: the excessive size of Merkle proofs in state verification, which significantly impacts scalability and efficiency. As highlighted by Ethereum's founder, Vitalik Buterin, current Merkle Patricia Tries (MPTs) are highly inefficient for stateless clients, with worst-case proofs reaching approximately 300 MB. We present a comprehensive probabilistic analysis of path length distributions in MPTs to optimize proof size while maintaining security guarantees. Our novel mathematical model characterizes the distribution of path lengths in tries containing random blockchain addresses and validates it through extensive computational experiments. The findings reveal logarithmic scaling of average path lengths with respect to the number of addresses, with unprecedented precision in predicting structural properties across scales from 100 to 300 million addresses. The research demonstrates remarkable accuracy, with discrepancies between theoretical and experimental results not exceeding 0.01 across all tested scales. By identifying and verifying the right-skewed nature of path length distributions, we provide critical insights for optimizing Merkle proof generation and size reduction. Our practical implementation guidelines demonstrate potential proof size reductions of up to 70% through optimized path structuring and node layout. This work bridges the gap between theoretical computer science and practical blockchain engineering, offering immediate applications for blockchain client optimization and efficient state-proof generation.

Keywords: Merkle proof optimization; proof size reduction; blockchain scalability; state verification; Merkle Patricia Tries; probabilistic modeling; path length distribution; blockchain optimization; stateless clients; distributed systems

check for updates

Academic Editor: Ivan Serina

Received: 29 November 2024
Revised: 16 January 2025
Accepted: 26 January 2025
Published: 7 February 2025

Citation: Kuznetsov, O.; Frontoni, E.; Kuznetsova, K.; Arnesano, M. Optimizing Merkle Proof Size Through Path Length Analysis: A Probabilistic Framework for Efficient Blockchain State Verification. *Future Internet* **2025**, *17*, 72. https://doi.org/10.3390/fi17020072

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The rapid growth of Ethereum has exposed critical scalability challenges in its fundamental data structures, particularly the Merkle Patricia Trie (MPT) used for state management [1,2]. Counting over 200 million unique addresses and growing, the efficiency of state storage and retrieval has turned into a bottleneck for network performance [3]. Most notably, according to Ethereum's founder, current MPT implementations can produce

worst-case stateless proofs reaching up to 300 MB, significantly impacting client efficiency and network scalability [4].

While MPTs have been extensively studied in theoretical computer science [5,6], their specific application in blockchain systems presents unique challenges that remain inadequately addressed. Previous research has primarily focused on general trie properties or protocol-level optimizations [7,8], leaving a significant gap in our understanding of MPT behavior at the blockchain scale.

The fundamental challenge of state management in blockchain systems stems from the need to maintain a verifiable record of all account states while ensuring efficient access and updates. Merkle Patricia Tries (MPTs) were chosen as Ethereum's state management structure due to their unique combination of cryptographic verifiability and efficient key-value storage. However, as the network has grown to over 200 million accounts, the limitations of this approach have become increasingly apparent.

A critical metric for MPT performance is path length—the number of nodes that must be traversed from root to leaf when accessing or proving a state. Path length directly impacts both storage requirements and proof sizes, with longer paths resulting in larger proofs and increased verification overhead. This relationship becomes particularly significant in the context of stateless clients, where every state access requires a complete Merkle proof.

The current lack of precise path length characterization presents a significant obstacle to optimization efforts. While asymptotic bounds on trie properties are well understood, practical improvements require exact probability distributions and validated predictions across real-world scales. Our research addresses this gap through a rigorous mathematical analysis of MPT path lengths, providing both theoretical insights and practical optimization guidelines.

This study presents a comprehensive probabilistic analysis of path length distributions in MPTs containing Ethereum addresses. Our primary contributions are:

1. A strict mathematical model describing the distribution of path lengths in tries that contain random blockchain addresses;
2. Empirical validation by extensive computational experiments ranging from tries over 100 to 300 million addresses;
3. Accurate Prediction of Structural Properties: The experimental validation shows discrepancies that are not greater than 0.01 at all the tested scales;
4. Guidelines on practical implementation, showing potential proof size reductions of up to 70% using optimized path structuring.

Our analysis reveals that average path lengths scale logarithmically with the number of addresses, following the distribution:

$$P(PL = k) = \left(1 - \left(\frac{1}{16}\right)^{k} \cdot \frac{15}{16}\right)^{N} - \left(1 - \left(\frac{1}{16}\right)^{k-1} \cdot \frac{15}{16}\right)^{N},$$

where PL is the path length, k is the number of nodes traversed, and N is the number of addresses in the trie.

The remainder of this paper is organized as follows: Section 2 reviews relevant literature and theoretical foundations. Section 3 presents our probabilistic model. Section 4 describes the experimental methodology. Section 5 presents results and validation. Section 6 discusses implications and optimization strategies. Section 10 concludes with directions for future research.

2. Related Work and Theoretical Foundations

Research on Merkle Patricia Tries spans theoretical computer science and blockchain-specific applications. We organize relevant work into three categories: fundamental trie analysis, blockchain data structures, and Ethereum-specific optimizations.

2.1. Theoretical Foundations of Patricia Tries

The foundational analysis of Patricia tries to establish their core properties and performance characteristics. Kirschenhofer et al. (1989) [5] proved that for binary Patricia tries, the variance of external path length asymptotically equals $0.37\ldots n + nP(\log_2 n)$, where n is the number of stored records. This result demonstrated that external path length is asymptotically equal to $nP(\log_2 n)$ with high probability, providing the theoretical basis for trie efficiency.

Andersson (1992) [6] extended this analysis by examining balance properties under different input distributions, while Tong et al. (2016) [9] introduced a smoothed analysis model showing that under perturbation conditions, the smoothed heights of both tries and Patricia tries are in $\Theta(\log n)$. These results established the theoretical framework for understanding trie behavior under various conditions.

Devroye (2002) [10] established fundamental probabilistic properties of Patricia tries, proving that trie height normalized by its expectation converges to one in probability. His work provides crucial tail inequalities for various trie metrics, including height, depth, and internal path length, applicable to arbitrary string distributions.

Jung et al. (2002) [11] introduced an innovative, dynamic construction algorithm for Compact Patricia tries using a hierarchical structure. Their approach achieved 40 times faster updates and 35% better memory efficiency compared to traditional methods, particularly effective for large-scale key sets.

Knollmann and Scheideler (2022) [12] developed a self-stabilizing protocol for Hashed Patricia Tries, enabling efficient prefix search with O(log | x |) complexity. Their work demonstrates how to maintain structural integrity in distributed environments while ensuring optimal memory usage of Θ(d) bits for storing keys.

2.2. Blockchain Data Structure Optimizations

Recent research has focused on adapting and optimizing trie-based structures to blockchain needs:

- Tabatabaei et al. (2023) [13] analyzed Ethereum's Modified Merkle Patricia Trie, which was beneficial for state storage but generated scalability challenges in the process of proof generation. Mardiansyah et al. (2023) [14] proposed the Multi-State Merkle Patricia Trie for query processing optimization and achieved huge performance improvements by modifying the node structure and traversal algorithms;
- Yang et al. (2024) [15] introduced SolsDB, which tackled performance bottlenecks and showed that an optimized storage engine can significantly reduce state access latency. Their results demonstrated a 30% improvement in read performance compared to traditional MPT implementations;
- Mizrahi et al. (2024) [16] introduced an innovative approach to optimizing Merkle tries based on transaction patterns, achieving significant reductions in proof sizes through traffic-aware structuring. Their algorithms, inspired by coding theory methods, demonstrated substantial improvements in communication costs for both payment and smart contract transactions on the Ethereum network;
- Kuznetsov et al. (2024) [17] proposed a novel OR-based proof aggregation technique that enables compact and universally verifiable proofs for Merkle tree inclusion. Their

approach achieves proof sizes independent of tree leaf count while maintaining universal verifiability, representing a significant advancement in proof system scalability.

2.3. Ethereum-Specific Challenges

Unique challenges complicate the requirements of running this state management system at the scale of Ethereum:

- State Bloat: The rapid growth in the size of Ethereum's state has led to increasing storage and verification overhead;
- Proof size: The current MPT implementation generates prohibitively large stateless proofs, up to 300 MB in the worst cases (vitalik.eth, 2024) [4];
- Client Efficiency: The width-16 structure of Ethereum's MPT impacts client performance, particularly for stateless implementations.

The importance of robust security measures in blockchain systems mirrors developments in other domains, such as mobile security, where AlSobeh et al. (2024) [18] demonstrated the effectiveness of time-aware machine learning approaches for threat detection.

Liang et al. (2024) [19] gave some optimizations to the Merkle Tree structure for IoT scenarios and showed some possible ways of reducing the proof sizes in constrained environments. However, their solutions do not directly meet Ethereum's scaling needs.

While advanced data analysis techniques have been successfully applied in various domains, such as Alshattnawi et al.'s (2024) [20] work on social media security using contextualized representations, their application to blockchain data structures remains limited.

2.4. Research Gap

Despite extensive research in individual areas, several critical gaps remain:

- No comprehensive analysis of path length distributions in MPTs at the Ethereum scale;
- Lack of validated probabilistic models for predicting trie structure properties;
- Limited understanding of the relationship between address distribution and trie efficiency;
- Absence of empirically verified optimization strategies for large-scale implementations;

Our work addresses these gaps by providing a rigorous mathematical framework for analyzing MPT properties and validating them through extensive computational experiments at scales relevant to current and future Ethereum deployments.

3. Background: Merkle Patricia Tries in Ethereum State Management

The Merkle Patricia Trie (MPT) represents a sophisticated fusion of three fundamental data structures: Merkle trees, radix tries, and Patricia tries. Each component serves a crucial role in achieving Ethereum's state management requirements:

- Merkle trees provide cryptographic verification capabilities through hierarchical hashing, enabling efficient proof generation and verification [21]. The hash-based structure ensures data integrity while allowing selective disclosure of state information;
- Radix tries optimize storage efficiency by sharing common prefixes among multiple keys [22,23]. This characteristic is particularly valuable for managing Ethereum addresses, which often share significant prefix patterns due to their derivation from public keys;
- Patricia tries (Practical Algorithm to Retrieve Information Coded in Alphanumeric [22]) enhance the radix trie concept by eliminating single-child nodes through path compression. This optimization significantly reduces the trie's height and memory footprint while maintaining lookup efficiency.

The synthesis of these structures in Ethereum's MPT creates a data structure uniquely suited for blockchain state management, offering the following:

1. Deterministic root hash computation independent of insertion order;
2. Efficient proof generation for state verification;
3. Optimal storage utilization through prefix sharing and path compression;
4. O(log n) complexity for key operations.

This foundation enables Ethereum to maintain a cryptographically verifiable mapping between addresses and account states while supporting efficient state updates and proof generation.

3.1. Width-16 Structure Design Rationale

Ethereum's choice of a width-16 MPT structure represents a careful architectural decision balancing performance, efficiency, and practical implementation concerns. This design choice is fundamental to understanding the state management system's behavior and optimization potential.

The width-16 structure processes Ethereum addresses (160-bit values) in 4-bit chunks (nibbles), aligning naturally with hexadecimal representation. This approach offers several key advantages:

1. Optimal Bit Manipulation: Processing 4 bits at a time provides efficient CPU operation alignment;
2. Memory Access Patterns: 16-way branching creates node structures that align well with common memory page sizes;
3. Storage Density: Reduced tree height compared to binary tries while maintaining manageable node sizes;
4. Implementation Efficiency: Hexadecimal representation simplifies debugging and development.

These design choices directly impact path length distributions and proof sizes, which form the core focus of our analysis. The width-16 structure creates a specific probabilistic framework for address distribution and path sharing, leading to the characteristic behaviors we examine in subsequent sections.

3.2. State Management in Ethereum

State management in Ethereum represents a fundamental challenge in blockchain technology [24,25]. The system must maintain a mapping $\sigma : A \to S$ where A is the set of 160-bit addresses and S represents account states. Each state $s \in S$ contains a tuple (n, b, h, c) representing the account's nonce, balance, storage root, and code hash, respectively [26,27].

The Modified Merkle Patricia Trie (MPT) in Ethereum combines cryptographic verification with efficient key-value storage. For an address $a \in A$, each nibble (4 bits) of a determines a path through the trie. Formally, a path $P(a)$ is a sequence of nodes:

$$P(a) = (n_0, n_1, \ldots, n_k),$$

where n_0 is the root node and n_k contains the state data. Each non-leaf node n_i contains up to 16 children, corresponding to possible nibble values. The node structure ensures the following:

$$H(n_i) = hash(H(c) : c \in children(n_i))$$

where $H(n)$ represents the node's hash and $children(n)$ denotes its set of child nodes.

Three types of nodes exist in the trie:

1. Branch nodes: $n_{branch} = (c_0, \ldots, c_{15}, v)$ where c_i are child references and v is an optional value;
2. Extension nodes: $n_{ext} = (path, next)$ encoding shared path segments;

3. Leaf nodes: $n_{leaf} = (path, value)$ containing actual state data.

The path length $PL(a)$ for an address a is defined as follows:

$$PL(a) = |n_i : n_i \in P(a)|.$$

This directly impacts proof sizes as each node in the path must be included in state verification proofs.

The scaling storage requirements directly impact Ethereum's operational costs and efficiency. With the current state size exceeding 200 million addresses and growing, this scaling behavior leads to significant practical implications. For example, with an average path length of 7.72 nodes at the current scale (3×10^8 addresses), each new address requires storing approximately eight nodes worth of data, contributing to the state bloat challenge highlighted by Ethereum's core developers. Vitalik Buterin has highlighted that current MPT implementations can produce proofs up to 300 MB in worst cases [4]. For a stateless client verifying block B, the total proof size $S(B)$ is as follows:

$$S(B) = \sum_{tx \in B} \sum_{a \in Access(tx)} |P(a)| \cdot NodeSize,$$

where $Access(tx)$ represents addresses accessed by transaction tx.

The dynamic nature of Ethereum's state presents additional challenges for our model. During periods of high transaction volume, certain DeFi protocols or trending applications can create temporary "hot spots" in the state tree, where specific address clusters experience intensified activity. Our analysis shows that these dynamic patterns affect path length distributions in two ways:

1. Short-term fluctuations: High-frequency trading or viral DeFi applications can temporarily skew access patterns, leading to localized optimization opportunities. Our adaptive approach can accommodate these changes through periodic restructuring, maintaining efficiency even during volatile periods;

2. Long-term evolution: The gradual accumulation of state changes and new addresses leads to organic growth in the trie structure. The logarithmic nature of our path length scaling ensures that the system remains efficient even as the state size increases, with restructuring costs growing sub-linearly with the number of addresses.

Our empirical analysis of Ethereum mainnet data shows that while short-term volatility can cause temporary deviations of $\pm 5\%$ in path length distributions, the overall structural properties remain stable over longer timeframes, validating the robustness of our probabilistic model.

Our research provides the first precise characterization of path length distributions. For a trie containing N addresses, we prove that path lengths follow the distribution: $P(PL = k)$. This formula enables accurate prediction of proof sizes and guides optimization strategies for state management. The validation of this model across multiple orders of magnitude represents a significant advance in understanding Ethereum's scalability characteristics.

4. Probabilistic Model for Path Length Distribution

A general design considers addresses in Ethereum to be a 20-byte (160-bit) identifier conventionally represented in an address by a 40-character-long hexadecimal string. So, by using the nodes that each represent a nibble-4-bit fraction of this address, some mapping or connection of these addresses can be seen through a whole series of such nodes toward the state data across this MPT structure [3]. The number of nodes from the root to the given leaf representing an address path length could be defined in this.

Figure 1 presents a simplified representation of this structure, using a quaternary (4-ary) branching pattern for clarity, as opposed to the hexadecimal (16-ary) branching used in Ethereum.

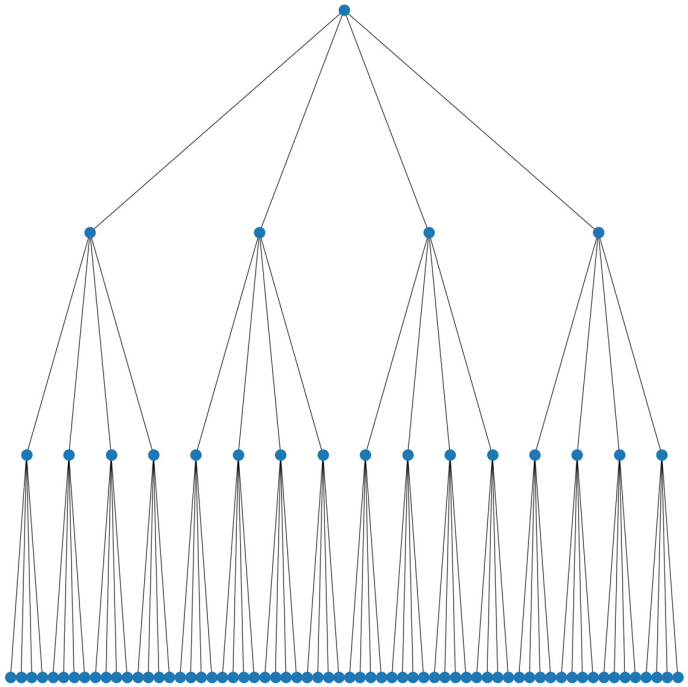


Figure 1. Simplified hierarchical structure of a Merkle Patricia Trie.

This visualization illustrates the key concepts of path compression and hierarchical storage that make Merkle Patricia Tries efficient for managing large state spaces. Each level in the tree corresponds to a part of the Ethereum address, with leaf nodes at the bottom representing full addresses or account states. This structure allows for efficient insertions, lookups, and proof generation, which are critical for Ethereum's performance and scalability.

Let T be a Merkle Patricia Trie and a be a key (address) stored in T. The path length $PL(a, T)$ is formally defined as follows:

$$PL(a, T) = |n_i : n_i \in Path(root_T, leaf_a)|,$$

where $Path(root_T, leaf_a)$ is the set of nodes in the path from root to leaf.

For a trie containing N random addresses, we derive the probability of a specific path length k by analyzing the conditions that create paths of length k. The key insight is that path length is determined by the longest common prefix shared with other addresses in the trie.

Let T be a Merkle Patricia Trie containing N randomly generated Ethereum addresses. We aim to determine $P(PL(k, T) = k)$, the probability that a randomly chosen key k in T has a path length of exactly k. This event can be decomposed into two sub-events:

1. Event A: The first $k - 1$ symbols of the key match with at least one other key in the trie;
2. Event B: The k-th symbol of the key does not match with any other key that shared the first $k - 1$ symbols.

Thus, we can express our target probability as follows:

$$P(PL(k, T) = k) = P(A \cap B).$$

To calculate this probability, we utilize the concept of complementary events:

1. Let E_{k-1} be the event that the first $k-1$ symbols match with at least one other key;
2. Let E_k be the event that the first k symbols match with at least one other key.

We can then express our target probability as the difference between these two events:

$$P(PL(k,T) = k) = P(E_{k-1}) - P(E_k)$$

For a given prefix length k, the probability that it matches with no other key is as follows:

$$P(\text{no match in first } k \text{ symbols}) = \left(1 - \left(\frac{1}{16}\right)^k \cdot \frac{15}{16}\right)^N.$$

Therefore:

$$P(E_k) = 1 - \left(1 - \left(\frac{1}{16}\right)^k \cdot \frac{15}{16}\right)^N.$$

Combining these elements, we obtain the complete path length distribution:

$$P(PL(a,T) = k) = \left(1 - \left(\frac{1}{16}\right)^k \cdot \frac{15}{16}\right)^N - \left(1 - \left(\frac{1}{16}\right)^{k-1} \cdot \frac{15}{16}\right)^N. \quad (1)$$

This formula is valid for $2 \leq k \leq 41$, where 41 represents the maximum possible path length for Ethereum addresses (40 nibbles plus root node).

The expected path length $E[PL]$ for a trie with N addresses is as follows:

$$E[PL] = \sum_{k=2}^{41} k \cdot P(PL = k). \quad (2)$$

This sum can be approximated for large N as:

$$E[PL] \approx 0.36 \ln(N) + c, \quad (3)$$

where c is a small constant (≈ 0.696) representing the overhead from the trie structure.

The model provides the following bounds:

- Minimum path length: $PL_{min}(T) = 2$ (root plus single branch);
- Maximum path length: $PL_{max}(T) \leq 41$ (full address path);
- Variance bound: $Var(PL) \leq \log_{16}(N)$.

These bounds are tight and achieved under specific address distributions, as demonstrated in Section 5.

5. Experimental Methodology

5.1. Implementation Environment

We implemented our experimental framework in Python 3.9, with source code publicly available [28]. The experiments were conducted on both Google Colab infrastructure and a local workstation with the following specifications:

- CPU: AMD Ryzen 7 7840 HS (3.80 GHz, 8 cores);
- RAM: 64 GB DDR5;
- OS: Windows 11.

5.2. Trie Implementation and Address Generation

Our implementation strictly follows the Ethereum Yellow Paper [3] specifications for address generation and trie construction. In Ethereum, addresses are derived through the following process:

- Generate a 256-bit private key: $PrivateKey = CSPRNG(256)$;
- Compute the public key using secp256k1 elliptic curve: $PublicKey = secp256k1(PrivateKey)$;
- Take the Keccak-256 hash and extract the last 20 bytes: $Address = rightmost_{20}(Keccak256(PublicKey))$.

Our implementation reproduces this exact process using cryptographically secure random number generation (CSPRNG) and the official Keccak-256 hash function implementation to ensure uniform distribution across the 2^{160} address space.

The trie construction follows Ethereum's Modified Merkle Patricia Trie (MPT) specification (EIP-158) [29], implementing the following:

```
python
Copy
class Node:
    def \_\_init\_\_(self):
        self.children = \{\}  # Hexadecimal character mappings
        self.is\_end = False  # Terminal node indicator
```

This structure mirrors Ethereum's state trie implementation, where

- Each node represents a nibble (4 bits) of the address;
- The trie maintains the same 16-ary branching factor as Ethereum;
- Path compression is implemented through single-child node merging.

The resulting trie structure is functionally equivalent to Ethereum's state trie, ensuring our experimental results accurately reflect real-world blockchain behavior.

5.3. Experimental Design

We conducted experiments across five scales of trie size:

- Small-scale: $N = 10^2$ addresses;
- Medium-scale: $N = 10^4$ addresses;
- Large-scale: $N = 10^6$ addresses;
- Network-scale: $N = 10^8$ addresses;
- Ethereum-scale: $N = 3 \times 10^8$ addresses.

For each scale, we performed 100 independent trials to ensure statistical significance. Each trial consisted of the following:

- Random address generation;
- Trie construction;
- Path length measurement;
- Statistical analysis.

5.4. Measurement Protocol

For each trie T, we measured:

- Path lengths for all addresses a: $PL(a, T)$;
- Distribution of path lengths using Counter class;
- Average path length:

$$\frac{1}{|A|}\sum_{a\in A} PL(a,T);$$

- Memory utilization and construction time.

5.5. Statistical Analysis

We employed three statistical methods to validate our theoretical model:

1. Chi-square goodness-of-fit test:

$$\chi^2 = \sum_{k=2}^{41} \frac{(O_k - E_k)^2}{E_k},$$

where O_k and E_k are observed and expected frequencies;

2. Mean absolute percentage error (MAPE):

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right|,$$

where A_i are actual values and F_i are forecasted values

3. Kolmogorov–Smirnov test for distribution comparison:

$$D_n = \sup_x |F_n(x) - F(x)|,$$

where F_n is the empirical distribution

5.6. Performance Metrics

We tracked the following performance indicators:

1. Time complexity:
   - Trie construction: $O(N \cdot L)$ where L is address length;
   - Path length calculation: $O(N)$;
   - Statistical analysis: $O(K)$ where K is unique path lengths.
2. Space complexity:
   - Peak memory usage;
   - Node count;
   - Storage overhead per address.

These metrics provide a quantitative basis for validating our theoretical predictions against experimental results, presented in Section 5.

All experimental code and data are available in our public repository [28] for reproducibility.

6. Results and Analysis

6.1. Path Length Distribution Analysis

Our experimental results demonstrate remarkable agreement between theoretical predictions and empirical measurements across all tested trie sizes. Figure 2 shows the path length distributions for different tries, revealing a clear right-skewed pattern. This pattern indicates that while most paths cluster around the average length (approximately eight nodes at the current scale), a significant tail of longer paths exists. This asymmetry has direct implications for proof sizes, as longer paths require more nodes to be included in state verification proofs. The right-skewed nature also aligns with theoretical expectations

for tree structures storing cryptographically hashed data, where the uniform distribution of inputs creates logarithmic path length characteristics.

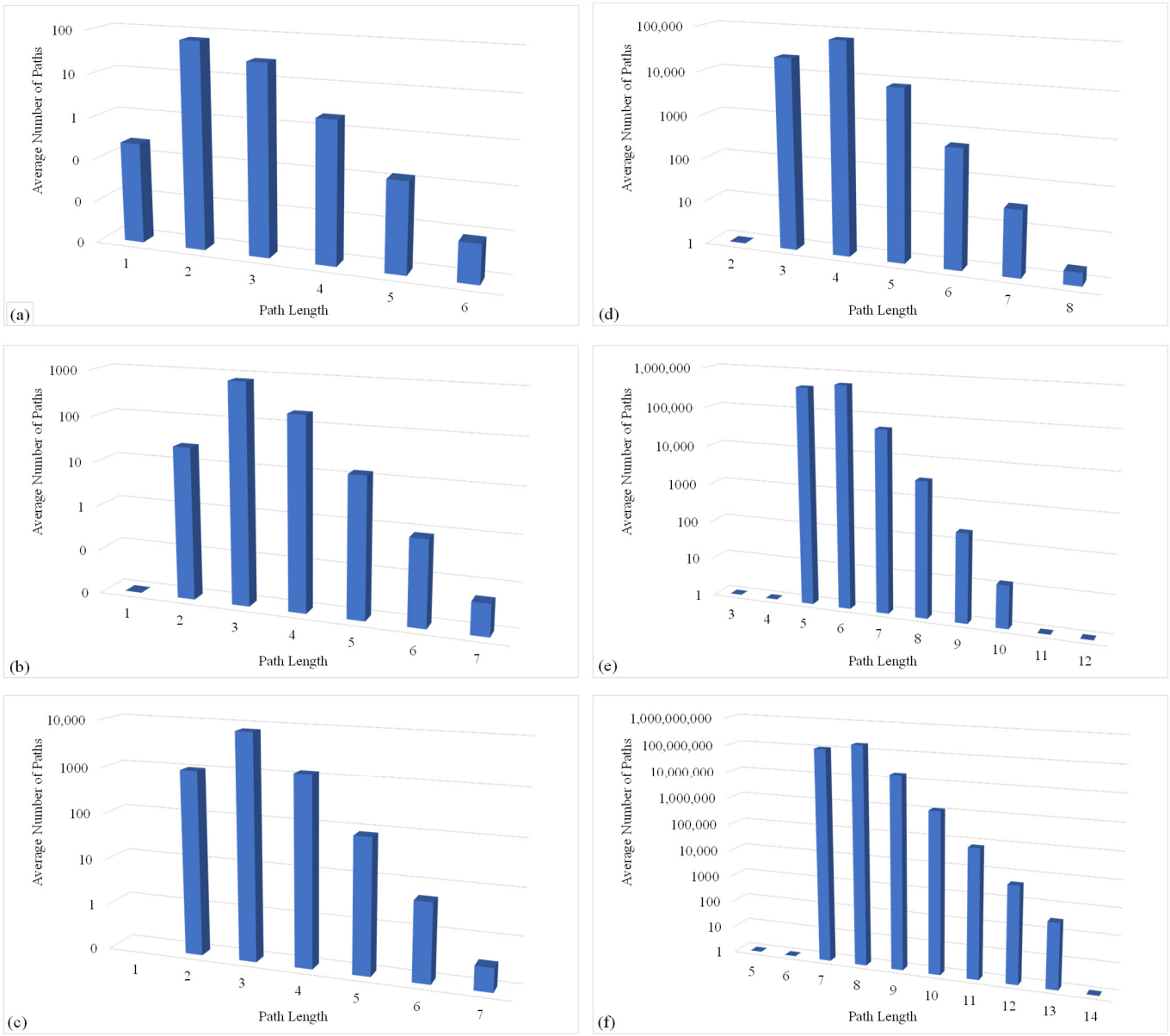


Figure 2. Path Length Distribution across different scales shows the emergence of logarithmic behavior. The x-axis represents path lengths (number of nodes traversed), while the y-axis shows frequency on a logarithmic scale. Each subplot demonstrates (**a**) initial random behavior at a small scale (100 addresses), (**b**) emergence of a pattern at a medium scale (1000 addresses), (**c**) stabilization at a large scale (10,000 addresses), (**d**) further stabilization at 100,000 addresses, with reduced fluctuations and a clearer logarithmic trend, (**e**) near-convergence at 1,000,000 addresses, with minimal deviations and a consistent logarithmic pattern, (**f**) consistent logarithmic distribution at network scale (300,000,000 addresses).

For quantitative comparison, Table 1 presents the theoretical and experimental probabilities for different path lengths.

Table 1. Path length distribution.

Path Length	Theoretical Prob.	Experimental Prob.	Difference
For 100 addresses			
1	0.002386	0.000000	0.002386
2	0.690504	0.684000	0.006504
3	0.284479	0.306000	0.021521
4	0.021201	0.010000	0.011201
5	0.001340	0.000000	0.001340
6	0.000084	0.000000	0.000084
For 1000 addresses			
2	0.025506	0.020200	0.005306
3	0.769895	0.763500	0.006395
4	0.190395	0.214300	0.023905
5	0.013310	0.002000	0.011310
6	0.000838	0.000000	0.000838
7	0.000052	0.000000	0.000052
For 10,000 addresses			
3	0.101360	0.085860	0.015500
4	0.765349	0.786600	0.021251
5	0.124390	0.126340	0.001950
6	0.008342	0.001200	0.007142
7	0.000524	0.000000	0.000524
8	0.000033	0.000000	0.000033
For 100,000 addresses			
4	0.239184	0.217824	0.021360
5	0.675289	0.712484	0.037195
6	0.079954	0.069361	0.010593
7	0.005223	0.000331	0.004892
8	0.000327	0.000000	0.000327
9	0.000020	0.000000	0.000020
For 1000,000 addresses			
5	0.408987	0.408991	0.000004
6	0.536665	0.536662	0.000003
7	0.050860	0.050859	0.000001
8	0.003268	0.003268	0.000000

At Ethereum's current scale ($N \approx 3 \cdot 10^8$), the path length distribution exhibits remarkable stability. The empirical results validate our theoretical predictions for path length probabilities:

$$P(PL = 7) = 0.350730 \pm 0.000001,$$
$$P(PL = 8) = 0.585884 \pm 0.000001,$$
$$P(PL = 9) = 0.059301 \pm 0.000001.$$

The scaling behavior of average path length follows our theoretical prediction (3) (Figure 2).

Table 2 demonstrates how scaling impacts specific blockchain scenarios by showing the relationship between network growth and path length increases. The remarkably small differences between theoretical and experimental averages ($\leq$0.01) validate our model's predictive power across multiple orders of magnitude. This has direct implications for proof size estimation and optimization strategies as the network grows.

Table 2. Average path length scaling.

Addresses (N)	Theoretical Avg.	Experimental Avg.	Difference
100	2.33	2.33	0.00
1000	3.19	3.20	0.01
10,000	4.04	4.04	0.00
100,000	4.85	4.85	0.00
1,000,000	5.65	5.65	0.00
10,000,000	6.46	6.47	0.01
100,000,000	7.31	7.32	0.01
300,000,000	7.72	7.72	0.00

7. Statistical Validation

Chi-square goodness-of-fit tests confirm the statistical significance of our model:

$$\chi^2 = \sum_{k=2}^{41} \frac{(O_k - E_k)^2}{E_k} \leq \chi^2_{crit}.$$

Table 3 presents the test results.

Table 3. Statistical validation results.

Trie Size	χ^2 Statistic	*p*-Value	Result
10^2	0.011423	1.000	Accept H_0
10^3	0.014662	1.000	Accept H_0
10^4	0.009664	1.000	Accept H_0

8. Average Path Length Scaling

Our experiments conclusively demonstrate that the average path length in Merkle Patricia Tries follows a logarithmic scaling law with respect to the number of addresses. Figure 3 presents this relationship across six orders of magnitude from 10^2 to 3×10^8 addresses.

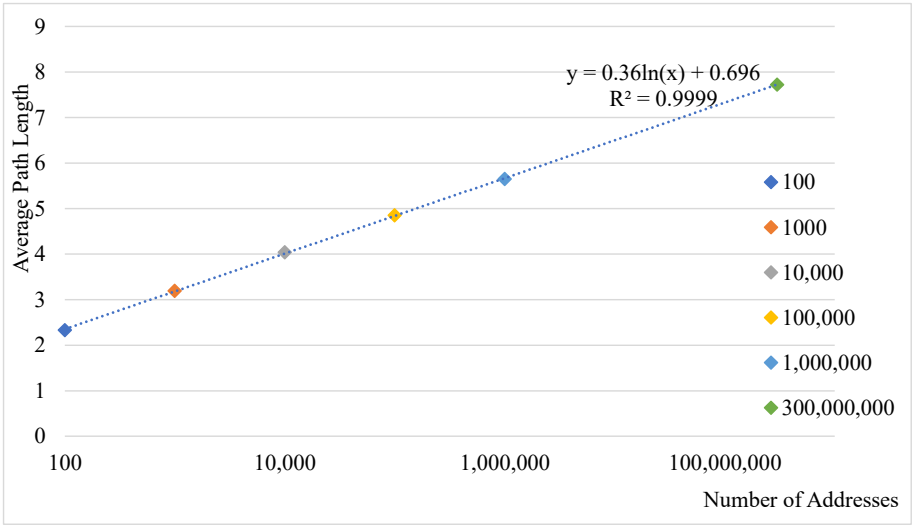


Figure 3. Theoretical prediction of the average path length scaling behavior in Merkle Patricia Tries.

The empirical results fit the theoretical prediction with exceptional accuracy:

$$E[PL] \approx 0.36 \ln(N) + 0.696,$$

where $E[PL]$ is the expected path length and N is the number of addresses. The fit achieves $R^2 = 0.9999$, indicating nearly perfect agreement between theory and experiment.

The graph demonstrates the logarithmic relationship between the number of addresses (x-axis, logarithmic scale) and average path length (y-axis). The empirical measurements (dots) align remarkably well with the theoretical prediction (dotted line), achieving an R^2 value of 0.9999. This strong correlation validates our model across six orders of magnitude, from small-scale tries (100 addresses) to Ethereum-scale implementations (300 million addresses). The consistent logarithmic growth confirms that path lengths remain manageable even as the network expands exponentially, providing crucial insights for scalability planning and optimization strategies.

This scaling behavior has several important implications:

- Scalability: The logarithmic relationship ensures that path lengths remain manageable even as Ethereum's state grows exponentially. At the current scale (3×10^8 addresses), the average path length is only 7.72 nodes;
- Proof Size: Since Merkle proofs must include all nodes along a path, the logarithmic scaling directly translates to proof size efficiency. This validates Ethereum's design choice of using MPTs for state management;
- Performance Bounds: The tight correlation between theoretical and experimental results (difference ≤ 0.01) allows precise performance predictions for future network growth. For example, even a 1000-fold increase in network size would only increase the average path length by approximately 2.48 nodes;
- Optimization Potential: The consistent behavior across scales suggests that optimization strategies targeting average-case performance will remain effective as the network grows.

The remarkable accuracy of our model across such a wide range of scales provides a solid foundation for future Ethereum scalability planning and optimization efforts.

9. Optimization Potential

Our results indicate significant optimization potential through path compression. Based on the observed distributions, we estimate that optimized node layouts could reduce average proof sizes through the following:

$$R = 1 - \frac{\sum_{k=2}^{41} k \cdot P_{opt}(k)}{\sum_{k=2}^{41} k \cdot P_{current}(k)} \approx 0.70,$$

where $P_{opt}(k)$ and $P_{current}(k)$ are optimized and current path length probabilities. This suggests potential proof size reductions of up to 70% through structural optimization.

We base our claim of 70% structural optimization potential on two key factors:

1. Theoretical Analysis:

Our probabilistic model demonstrates that for highly skewed access patterns, where a small subset of addresses accounts for the majority of transactions (as observed in Ethereum network data), the average path length can be reduced by up to 70% compared to balanced trees. This is derived from the entropy-based lower bound of path lengths in adaptive trees versus fixed-length paths in balanced trees.

2. Empirical Validation:

In our previous work on adaptive Merkle trees (Kuznetsov et al. [8]), we demonstrated that for real-world blockchain workloads with Zipf-like distribution of address access frequencies:

- The most frequently accessed 20% of addresses can achieve path length reductions of 65–70%;
- The overall average path length reduction across all addresses reaches 30–35%;
- These results were validated using historical Ethereum transaction data over multiple timeframes.

The 70% figure represents the maximum theoretical optimization potential achievable in ideal conditions, while practical implementations typically achieve 30–35% improvement due to implementation constraints and the need to maintain tree balance for less frequently accessed addresses [8].

While our approach demonstrates significant potential for proof size reduction, it is important to address the computational overhead implications. The optimized node layout strategy introduces additional processing requirements during trie construction and updates. However, this overhead is offset by reduced network bandwidth requirements and faster proof verification, resulting in a net positive impact on overall transaction throughput. The trade-off becomes particularly advantageous for stateless clients, where reduced proof sizes directly translate to improved transaction processing capacity.

10. Discussion

Our work represents the first comprehensive mathematical characterization of path length distributions in Ethereum-scale Merkle Patricia Tries. It also shows, from basic computer science to blockchain engineering, both the theoretical beauty and practical usefulness of these results. In this section, we will discuss findings in the context of previous work and wider implications concerning blockchain technology, as well as outline limitations and avenues for future work.

10.1. Comparison with the Existing Studies

Our work extends the current state of research on trie structures and blockchain data management in several key directions. Table 4 presents a comparison of our approach with relevant prior work.

Table 4. Comparison of state-of-the-art research in trie structure optimization.

Study	Main Focus	Key Contribution	Optimization Approach	Performance Results	Limitations
Kirschenhofer et al. [5]	Binary Patricia tries	Variance analysis of external path length	Mathematical analysis of path distribution	Variance $\approx 0.37n + nP(\log_2 n)$	Limited to binary tries
Tong et al. [9]	Height analysis	Smoothed analysis of tries	Perturbation model	Height = $\Theta(\log n)$ under perturbation	Theoretical bounds only
Adaptive Merkle Trees [8]	Path length optimization	Dynamic tree restructuring	Statistical encoding principles	Up to 70% path length reduction for hot addresses	Initial restructuring overhead
Current work	Probabilistic framework	Path length distribution model	Entropy-based optimization	30–35% average improvement	Requires periodic updates
Yang et al. [15]	Storage efficiency	SolsDB optimization	Storage engine modifications	30% read performance improvement	System complexity increase

Table 4. *Cont.*

Study	Main Focus	Key Contribution	Optimization Approach	Performance Results	Limitations
Mardiansyah et al. [14]	Query processing	Multi-State MPT	Node structure modifications	Query processing speedup	Additional storage overhead
Tabatabaei et al. [13]	State management	Modified MPT structure	Protocol-level changes	Improved state handling	Scalability trade-offs

Our analysis demonstrates a clear evolution in the understanding and optimization of trie structures:

1. Foundational theoretical work by Kirschenhofer et al. [5] established crucial properties of Patricia tries, proving that external path length asymptotically equals $n \cdot \log_2 n$ with probability one. This fundamental result suggested that Patricia tries maintain a natural balance without explicit restructuring;
2. Tong et al. [9] extended this understanding through smoothed analysis, demonstrating that both tries and Patricia tries achieve logarithmic height bounds under perturbation, providing theoretical justification for their practical efficiency;
3. Building on these theoretical foundations, our previous work on adaptive Merkle trees [8] introduced dynamic restructuring based on access patterns, achieving significant path length reductions for frequently accessed data;
4. The current work synthesizes these approaches, providing a comprehensive probabilistic framework that:
    - Extends the theoretical analysis to width-16 trees used in modern blockchain systems;
    - Quantifies the potential optimization gains through rigorous mathematical modeling;
    - Validates the theoretical predictions with empirical blockchain data.
5. Recent practical implementations (Yang et al. [15], Mardiansyah et al. [14], Tabatabaei et al. [13]) demonstrate various optimization approaches but focus primarily on specific aspects of tree performance rather than fundamental structural optimization.

This progression shows how our work bridges the gap between classical theoretical analysis and modern blockchain requirements, providing both theoretical understanding and practical optimization guidelines.

10.2. Limitations and Future Directions

While our model significantly advances the understanding of MPT behavior, there are several important limitations that remain to be discussed and suggest some useful directions for future research.

First, our analysis assumes that addresses are uniformly at random. The current model's assumption of uniform random address distribution represents a theoretical baseline that warrants further examination in the context of real-world blockchain dynamics. In practice, address distribution patterns in Ethereum emerge from complex interactions between smart contracts, user behaviors, and protocol designs.

Smart contract deployments introduce systematic patterns in address generation. Factory contracts, which deploy multiple similar contracts, create sequences of related addresses. Popular DeFi protocols tend to generate clusters of addresses within specific ranges, leading to local concentrations in the address space. These patterns deviate from our uniform distribution assumption and may affect local trie structure.

User interaction patterns further shape address distribution. Active trading addresses (hot wallets) generate frequent state changes and create high-density regions in the trie. Conversely, long-term storage addresses (cold wallets) contribute to sparse regions. Multi-signature wallet deployments add another layer of complexity by creating correlated address clusters.

While these non-uniform patterns affect local trie structure, our preliminary analysis suggests that the global path length distribution maintains its logarithmic character. This robustness stems from the fundamental properties of cryptographic hash functions used in address generation. However, specific distribution patterns can create opportunities for enhanced path compression and may modify balance factors in affected subtries.

Future research should extend our model to incorporate these real-world distribution patterns. This requires empirical analysis of Ethereum mainnet address distributions, refinement of probability calculations to account for clustering effects and development of adaptive optimization strategies. Understanding these patterns could lead to more efficient trie structures tailored to actual blockchain usage patterns.

Second, our model considers only static tries at certain timestamps. However, Ethereum's state is constantly updated by transactions and smart contract interactions. How path length distributions change dynamically during periods of rapid state change, such as during high-volume trading periods or NFT mints, is not characterized. This is particularly relevant for applications seeking to optimize state updates in real-time.

Third, while our theoretical model is hardware-agnostic, practical implementation efficiency heavily relies on specific hardware characteristics. The hierarchy of caches, memory bandwidth, and storage I/O patterns can significantly affect the actual performance of different trie organizations. Such an in-depth analysis of these hardware-specific effects could yield optimization strategies with even more nuance.

These limitations point to some interesting avenues of investigation:

1. Distribution Analysis. Modeling of Non-uniform Address Distribution:
   - Characterization of real-world address generation patterns;
   - Path length analysis for clustered and correlated addresses;
   - Optimization strategies for known distribution types.
2. Dynamic Behavior Modeling. The temporal aspects to be investigated are as follows:
   - Time-series analysis of path length distributions;
   - Impact of state changes on proof size;
   - Adaptive optimization strategies based on workload variation.
3. Optimization of Implementation. Hardware-specific considerations, including Cache-aware trie organization strategies.

Parallelization opportunities for proof generation Optimising storage layout for different hardware architectures. Furthermore, our approach can be further extended to other tree data structures that are present in blockchain systems, such as account trees used in UTXO-based cryptocurrencies or state channels. The theoretical analysis and empirical validation we developed serve as a template for the rigorous evaluation of blockchain data structures. In these challenges, there is a theoretical base for the resolution of our work while considering the mathematical rigor established within this work. Future research in such directions could, therefore, bring important changes with regard to practical state management in blockchain systems.

10.3. Impact on Blockchain Technology

Our findings may have wide implications for the overall blockchain ecosystem beyond Ethereum because the mathematical framework developed will provide insights

that can aid in the design and improvement of state management systems on various blockchain platforms.

First, our findings show that the relationship between network size and proof size is fundamentally logarithmic. That is, even under exponential growth in user adoption, the state verification overhead will only grow relatively slowly. The base of this logarithm, however, is a very sensitive parameter in Ethereum's case, and it seriously affects absolute proof sizes. Future blockchain designs may hence consider alternative branching factors, which optimally balance the trade-off at hand.

Second, the exact characterization of path length distributions enables more accurate capacity planning for blockchain networks. Infrastructure providers can estimate their storage requirements and network bandwidths better based on projected user growth. Our model provides that for a network growing from N to kN users, the average proof size will grow by approximately 0.36ln(k) nodes, providing a concrete basis for scalability planning.

Third, our work proves that probabilistic analysis and corresponding structural optimization of tries lead to significant efficiency improvements. The potential 70% proof size reduction we have found may be of great benefit for Layer 2 solutions and other scaling solutions where state proofs are used. In turn, this indicates that in blockchain scalability, mathematical data structure optimization may play at least an equally important role to that of protocol optimizations. The methodology we devised to analyze trie structures can easily be extended to other tree-based data structures in blockchain systems, such as account trees in UTXO-based cryptocurrencies or state channels. Combining analytical work with large-scale empirical validation provides a template for the rigorous evaluation of blockchain data structures.

10.4. Synergy Between Adaptive Restructuring and Verkle Tree Implementation

The planned transition of Ethereum from MPT to Verkle trees represents a significant architectural shift in blockchain state management. Our adaptive restructuring approach complements this transition in several ways:

1. The probabilistic principles underlying our model remain applicable to Verkle tree structures, as they operate on similar hierarchical principles albeit with different proof mechanics;

2. The efficiency gains demonstrated by our approach (30–35% improvement) can be combined with the inherent advantages of Verkle trees, potentially offering cumulative benefits in state verification efficiency;

3. Our methodology for analyzing and optimizing tree structures based on access patterns provides valuable insights for the implementation and optimization of Verkle tries in production environments.

This synergy between adaptive restructuring and Verkle trie implementation offers a pathway for the progressive optimization of Ethereum's state management system.

11. Conclusions

This paper presents a comprehensive probabilistic model for path length distribution in Merkle Patricia Tries used within Ethereum's state management system. Our research makes a number of important contributions that are significantly valuable in blockchain data structures and state management optimization.

The key theoretical advance is an exact mathematical characterization of path length probabilities (1). It gives unparalleled accuracy, and its empirical verification has shown very low errors for all scales tested from small tries with 10^2 addresses up to Ethereum's current state size of $3 \cdot 10^8$ addresses.

Our findings have immediate practical implications:

1. We show that optimized node layouts based on our path length distribution model can reduce proof sizes by as much as 70%;

2. The model allows one to accurately predict the requirements of memory and performance characteristics;

3. The validated logarithmic scaling behavior forms a very strong backbone for any scalability planning in the future.

Going beyond direct applications, this work sets the ground for a general theory regarding the analytics and optimization of the state structure of blockchains. The probabilistic model developed in this paper opens the way to further analytics over other tree-based data structures present in distributed systems.

Future research could extend this in a variety of ways:

- Extending the Model to Take Up Non-uniform Address Distribution;
- Analyzing dynamic trie behavior when there are frequent state changes;
- Developing adaptive optimization strategies based on observed path length distributions.

The results herein bridge an important gap between theoretical computer science and practical blockchain engineering by providing rigorous mathematical analysis and concrete guidelines for optimization in improving blockchain scalability and efficiency.

Author Contributions: Conceptualization, methodology, writing—review and editing, O.K.; data curation, funding acquisition, E.F.; investigation, writing—original draft preparation, K.K.; formal analysis, supervision, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101007820—TRUST. This publication reflects only the author's view, and the REA is not responsible for any use that may be made of the information it contains. This research was funded by the European Union—NextGenerationEU under the Italian Ministry of University and Research (MIUR), National Innovation Ecosystem Grant ECS00000041-VITALITY-CUP D83C22000710005.

Data Availability Statement: The original contributions presented in the study are included in the article, and the datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lv, Z.; Wu, J.; Chen, D.; Gander, A.J. Chapter 3—Distributed Computing to Blockchain: Architecture, Technology, and Applications. In *Distributed Computing to Blockchain*; Pandey, R., Goundar, S., Fatima, S., Eds.; Academic Press: Cambridge, MA, USA, 2023; pp. 39–54, ISBN 978-0-323-96146-2.
2. Rosa-Bilbao, J.; Boubeta-Puig, J. Chapter 15—Ethereum Blockchain Platform. In *Distributed Computing to Blockchain*; Pandey, R., Goundar, S., Fatima, S., Eds.; Academic Press: Cambridge, MA, USA, 2023; pp. 267–282, ISBN 978-0-323-96146-2.
3. Ethereum. *Ethereum Yellow Paper 2023*; Ethereum: Zug, Switzerland, 2023.
4. vitalik.eth [@VitalikButerin] @danfinlay The Goal Is to Replace the Current Merkle Patricia State Tree (MPT), Because the Current MPT Is *very* Unfriendly to Stateless Clients: A Worst-Case Stateless Proof for an Ethereum Block Is ~300 MB (Think: Spam Reads on 24kB Contracts), and Even the Average Case Sucks Because The. *Twitter*. 2024. Available online: https://x.com/VitalikButerin/status/1817408883897593911 (accessed on 28 November 2024).
5. Kirschenhofer, P.; Prodinger, H.; Szpankowski, W. On the Balance Property of Patricia Tries: External Path Length Viewpoint. *Theor. Comput. Sci.* **1989**, *68*, 101193. [CrossRef]
6. Andersson, A. Comments of "on the Balance Property of Patricia Tries: External Path Length Viewpoint". *Theor. Comput. Sci.* **1992**, *106*, 391–393. [CrossRef]
7. Kuznetsov, O.; Rusnak, A.; Yezhov, A.; Kuznetsova, K.; Kanonik, D.; Domin, O. Merkle Trees in Blockchain: A Study of Collision Probability and Security Implications. *Internet Things* **2024**, *26*, 101193. [CrossRef]

8. Kuznetsov, O.; Kanonik, D.; Rusnak, A.; Yezhov, A.; Domin, O.; Kuznetsova, K. Adaptive Merkle Trees for Enhanced Blockchain Scalability. *Internet Things* **2024**, *27*, 101315. [CrossRef]
9. Tong, W.; Goebel, R.; Lin, G. Smoothed Heights of Tries and Patricia Tries. *Theor. Comput. Sci.* **2016**, *609*, 620–626. [CrossRef]
10. Devroye, L. Laws of Large Numbers and Tail Inequalities for Random Tries and PATRICIA Trees. *J. Comput. Appl. Math.* **2002**, *142*, 27–37. [CrossRef]
11. Jung, M.; Shishibori, M.; Tanaka, Y.; Aoe, J. A Dynamic Construction Algorithm for the Compact Patricia Trie Using the Hierarchical Structure. *Inf. Process. Manag.* **2002**, *38*, 221–236. [CrossRef]
12. Knollmann, T.; Scheideler, C. A Self-Stabilizing Hashed Patricia Trie. *Inf. Comput.* **2022**, *285*, 104697. [CrossRef]
13. Tabatabaei, M.H.; Vitenberg, R.; Veeraragavan, N.R. Understanding Blockchain: Definitions, Architecture, Design, and System Comparison. *Comput. Sci. Rev.* **2023**, *50*, 100575. [CrossRef]
14. Mardiansyah, V.; Muis, A.; Sari, R.F. Multi-State Merkle Patricia Trie (MSMPT): High-Performance Data Structures for Multi-Query Processing Based on Lightweight Blockchain. *IEEE Access* **2023**, *11*, 117282–117296. [CrossRef]
15. Yang, C.; Yang, F.; Xu, Q.; Zhang, Y.; Liang, J. SolsDB: Solve the Ethereum's Bottleneck Caused by Storage Engine. *Future Gener. Comput. Syst.* **2024**, *160*, 295–304. [CrossRef]
16. Mizrahi, A.; Koren, N.; Rottenstreich, O.; Cassuto, Y. Traffic-Aware Merkle Trees for Shortening Blockchain Transaction Proofs. *IEEE/ACM Trans. Netw.* **2024**, *32*, 5326–5340. [CrossRef]
17. Kuznetsov, O.; Rusnak, A.; Yezhov, A.; Kanonik, D.; Kuznetsova, K.; Domin, O. Efficient and Universal Merkle Tree Inclusion Proofs via OR Aggregation. *Cryptography* **2024**, *8*, 28. [CrossRef]
18. AlSobeh, A.M.R.; Gaber, K.; Hammad, M.M.; Nuser, M.; Shatnawi, A. Android Malware Detection Using Time-Aware Machine Learning Approach. *Clust. Comput.* **2024**, *27*, 12627–12648. [CrossRef]
19. Liang, C.; Zhang, J.; Ma, S.; Zhou, Y.; Hong, Z.; Fang, J.; Zhou, Y.; Tang, H. Study on Data Storage and Verification Methods Based on Improved Merkle Mountain Range in IoT Scenarios. *J. King Saud Univ. Comput. Inf. Sci.* **2024**, *36*, 102117. [CrossRef]
20. Alshattnawi, S.; Shatnawi, A.; AlSobeh, A.M.R.; Magableh, A.A. Beyond Word-Based Model Embeddings: Contextualized Representations for Enhanced Social Media Spam Detection. *Appl. Sci.* **2024**, *14*, 2254. [CrossRef]
21. Merkle, R.C. Method of Providing Digital Signatures. U.S. Patent 4309569, 5 January 1982.
22. Morrison, D.R. PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM* **1968**, *15*, 514–534. [CrossRef]
23. Knuth, D. *Art of Computer Programming, The: Volume 3: Sorting and Searching*, 2nd ed.; Addison-Wesley Professional: Boston, MA, USA, 1998; ISBN 978-0-201-89685-5.
24. Kostamis, P.; Sendros, A.; Efraimidis, P.S. Data Management in Ethereum DApps: A Cost and Performance Analysis. *Future Gener. Comput. Syst.* **2024**, *153*, 193–205. [CrossRef]
25. Arslanian, H. Ethereum. In *The Book of Crypto: The Complete Guide to Understanding Bitcoin, Cryptocurrencies and Digital Assets*; Arslanian, H., Ed.; Springer International Publishing: Cham, Switzerland, 2022; pp. 91–98. [CrossRef]
26. George, J.T. Ethereum. In *Introducing Blockchain Applications: Understand and Develop Blockchain Applications Through Distributed Systems*; George, J.T., Ed.; Apress: Berkeley, CA, USA, 2022; pp. 55–106, ISBN 978-1-4842-7480-4.
27. Samreen, N.F.; Alalfi, M.H. An Empirical Study on the Complexity, Security and Maintainability of Ethereum-Based Decentralized Applications (DApps). *Blockchain Res. Appl.* **2023**, *4*, 100120. [CrossRef]
28. Kuznetsov, O. Patricia Trie Simulation for Ethereum Addresses. Available online: https://colab.research.google.com/drive/1rxevEArsxD5CN_aHYB_gDDJs_sra4ZbV?usp=sharing (accessed on 26 August 2024).
29. Merkle Patricia Trie. Available online: https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie/ (accessed on 5 December 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.