cryptography

Article

Evaluating the Security of Merkle Trees: An Analysis of Data Falsification Probabilities

Oleksandr Kuznetsov [1,2,*], Alex Rusnak [1], Anton Yezhov [1], Kateryna Kuznetsova [1], Dzianis Kanonik [1] and Oleksandr Domin [1]

[1] Proxima Labs, 1501 Larkin Street, Suite 300, San Francisco, CA 94102, USA; alex@proxima.one (A.R.); anton@proxima.one (A.Y.); kateryna@proxima.one (K.K.); denis@proxima.one (D.K.); dyomin@proxima.one (O.D.)

[2] Department of Political Sciences, Communication and International Relations, University of Macerata, Via Crescimbeni, 30/32, 62100 Macerata, Italy

* Correspondence: kuznetsov@karazin.ua or kuznetsov@proxima.one

Abstract: Addressing the critical challenge of ensuring data integrity in decentralized systems, this paper delves into the underexplored area of data falsification probabilities within Merkle Trees, which are pivotal in blockchain and Internet of Things (IoT) technologies. Despite their widespread use, a comprehensive understanding of the probabilistic aspects of data security in these structures remains a gap in current research. Our study aims to bridge this gap by developing a theoretical framework to calculate the probability of data falsification, taking into account various scenarios based on the length of the Merkle path and hash length. The research progresses from the derivation of an exact formula for falsification probability to an approximation suitable for cases with significantly large hash lengths. Empirical experiments validate the theoretical models, exploring simulations with diverse hash lengths and Merkle path lengths. The findings reveal a decrease in falsification probability with increasing hash length and an inverse relationship with longer Merkle paths. A numerical analysis quantifies the discrepancy between exact and approximate probabilities, underscoring the conditions for the effective application of the approximation. This work offers crucial insights into optimizing Merkle Tree structures for bolstering security in blockchain and IoT systems, achieving a balance between computational efficiency and data integrity.

Keywords: Internet of Things (IoT); data falsification; Merkle trees; blockchain; hashing; probability analysis

check for updates

Citation: Kuznetsov, O.; Rusnak, A.; Yezhov, A.; Kuznetsova, K.; Kanonik, D.; Domin, O. Evaluating the Security of Merkle Trees: An Analysis of Data Falsification Probabilities. *Cryptography* **2024**, *8*, 33. https://doi.org/10.3390/cryptography8030033

Academic Editors: Cheng-Chi Lee, Tuan-Vinh Le, Chun-Ta Li, Dinh-Thuan Do and Agbotiname Lucky Imoize

Received: 12 June 2024
Revised: 12 July 2024
Accepted: 15 July 2024
Published: 1 August 2024

1. Introduction

In the era of digitalization and the rise in blockchain technologies, the integrity and security of data have become paramount concerns [1]. Merkle Trees, a fundamental component in blockchain architectures such as Ethereum, play a crucial role in ensuring efficient and secure data verification [2,3]. However, despite their widespread application, the collision resistance of Merkle Trees and their robustness against preimage attacks have not been thoroughly investigated, leading to a notable gap in our comprehensive understanding of blockchain security mechanisms [4,5].

This study aims to bridge this gap by critically examining the susceptibility of Merkle Trees to hash collisions, a potential vulnerability that poses significant risks to data security within blockchain systems. Our research employs a meticulous blend of a theoretical analysis and empirical validation to scrutinize the probability of root collisions in Merkle Trees, considering various factors such as hash length and path length within the tree.

The significance of this research lies in its potential to enhance the security and operational efficacy of blockchain-based systems. By providing a deeper understanding of the vulnerabilities and strengths of Merkle Trees, we aim to offer valuable guidance for blockchain developers and researchers in optimizing these critical data structures.

Our investigation is structured as follows:

1. We begin with a comprehensive review of the state of the art, identifying key research gaps in the field of Merkle Tree security.
2. We then outline our methodological approach, detailing both the theoretical framework and experimental design used in our study.
3. The results section presents our findings on collision probabilities, including both theoretical derivations and empirical observations.
4. In the discussion, we interpret these results in the context of blockchain security, exploring their implications for current and future blockchain implementations.
5. Finally, we conclude with a summary of our key findings and their potential impact on the field, along with suggestions for future research directions.

This study presents both exact mathematical formulations and approximate estimations to assess the likelihood of these root collisions. The implications of these findings are significant for blockchain systems, where Merkle Trees are extensively employed. However, our focus extends beyond the blockchain, venturing into the realm of IoT, where the security implications are profound and the context differs markedly. By exploring the nuances of Merkle Trees in this new environment, this paper aims to contribute to the broader discourse on IoT security, offering insights and recommendations for leveraging Merkle Trees to fortify IoT networks against data tampering and ensure the veracity of the information within these interconnected systems.

In essence, this paper endeavors to bridge the gap between the established utility of Merkle Trees in blockchain technology and their prospective application in securing IoT ecosystems. Through a rigorous analysis and assessment, it seeks to illuminate the potential and limitations of Merkle Trees in a domain where data integrity is not just a necessity but a cornerstone of operational reliability and trust.

2. Literature Review

Freitag et al. [6] explore the time–space trade-offs in sponge hashing, offering a novel perspective on collision resistance. Their study on short collisions within this context is particularly noteworthy for its insights into the sponge construction's parameters. This work is instrumental in understanding the limitations and attack vulnerabilities of sponge hashing, thereby contributing significantly to the field of cryptographic hashing methods.

In another pivotal study [7], Ghoshal and Komargodski examine preprocessing adversaries in Merkle–Damgård (MD) hashing. Their focus on bounded-length collisions in the random oracle model provides a nuanced understanding of the MD hashing's resilience against certain types of attacks. This research is particularly relevant in evaluating the security of widely used hashing techniques in the context of IoT devices.

Hu and colleagues delve into the vulnerabilities of the Merkle approach in proving liabilities [8]. Their analysis of the Maxwell protocol and its susceptibility to underreporting liabilities offers critical insights into the weaknesses of Merkle-based systems in decentralized environments. This research is particularly relevant for understanding the security of transaction systems like Bitcoin exchanges and has profound implications for blockchain technology.

The work by Kumari et al. [9] introduces the Signature-based Merkle Hash Multiplication (SMHM) algorithm, aimed at securing communication in IoT devices. Their approach, considering the challenges posed by 6G technology, offers a forward-looking perspective on securing IoT networks against quantum computing threats. This research contributes significantly to developing robust security protocols in the IoT realm.

Mitra and team [10] propose the Polar Coded Merkle Tree (PCMT) to improve the detection of Data Availability (DA) attacks in blockchain systems. Their approach addresses the limitations of previous coding techniques and presents a viable solution for large blockchains. This innovation is crucial for enhancing the security and reliability of blockchain systems against DA attacks.

In a subsequent study, the same authors introduce the Graph Coded Merkle Tree (GCMT), further refining their approach to mitigating DA attacks in blockchain systems [11]. Their informed design of polar factor graphs and the focus on large block size applications provide a comprehensive solution to previously identified challenges, marking a significant advancement in blockchain security.

Rao et al. [12] propose a dynamic outsourced auditing scheme for cloud storage, utilizing a batch-leaves-authenticated Merkle Hash Tree. Their approach addresses the trust issues in third-party auditing and supports verifiable dynamic updates, which is critical for cloud computing security. This research fills a vital gap in ensuring data integrity in outsourced environments.

Sarkar's study [13] introduces a new domain extender for collision-resistant hash functions, improving upon the Merkle–Damgård iteration. The proposed directed acyclic graph-based construction offers a more efficient alternative for hashing arbitrary-length strings. This innovation significantly reduces computational requirements, paving the way for more efficient hashing methods.

Xu and colleagues [14] develop a dynamic Fully Homomorphic encryption-based Merkle Tree (FHMT) for lightweight streaming authenticated data structures. Their work is particularly relevant for streaming data environments, offering a balanced performance between client and server. This research is pivotal in advancing the use of Merkle Trees in dynamic, resource-limited contexts.

Zhu et al. [2] propose an improved convolution Merkle Tree-based blockchain scheme for secure electronic medical record storage. Their innovative approach in employing a convolutional layer structure significantly enhances efficiency and security, making it a notable contribution to the field of secure data storage and transmission, especially in healthcare.

Recent advancements in IoT security and communication protocols have further highlighted the importance of robust data integrity mechanisms in distributed systems. Buccafurri et al. [15] proposed MQTT-A, a broker-bridging peer-to-peer architecture designed to achieve anonymity in MQTT (Message Queuing Telemetry Transport) systems. This work addresses the growing need for privacy in IoT communications, a concern that intersects with the data integrity issues addressed by Merkle Trees. In a subsequent study, Buccafurri et al. [16] introduced MQTT-I, focusing on achieving end-to-end data flow integrity in MQTT. Their approach emphasizes the critical nature of maintaining data integrity throughout the entire communication process in IoT networks, aligning closely with the objectives of Merkle Tree implementations in blockchain and distributed systems.

The evolving landscape of IoT security datasets presents both challenges and opportunities for research in this field. Kaur et al. [17] conducted a comprehensive review of IoT security dataset evolution, highlighting the complexities and future directions in this domain. Their work underscores the need for robust security mechanisms, such as those provided by Merkle Trees, to address the diverse and evolving threat landscape in IoT ecosystems.

In the healthcare sector, the application of IoT technologies has opened new avenues for patient care and medical data management. Li et al. [18] provided an extensive review of IoT applications in healthcare, emphasizing the potential for improved patient monitoring, personalized treatment strategies, and enhanced healthcare delivery. The integration of secure data structures like Merkle Trees in healthcare IoT systems could significantly contribute to addressing the data integrity and security concerns highlighted in their study.

Despite the extensive research on Merkle Trees in blockchain technology, several critical gaps persist in our understanding of their application to IoT systems:

- Probabilistic Security Analysis: While Merkle Trees are widely used in IoT for data integrity, there is a lack of a rigorous probabilistic analysis of their security properties in resource-constrained IoT environments. Previous studies have primarily focused on blockchain applications, leaving a gap in our understanding of how these structures perform under the unique constraints of IoT systems.

- Scalability in IoT Contexts: The relationship between Merkle Tree depth and security in large-scale IoT networks remains underexplored. As IoT deployments grow in size and complexity, understanding how Merkle Tree structures scale and maintain security becomes crucial.
- Trade-offs in Resource-Limited Devices: IoT devices often have limited computational power and storage. The optimal balance between security strength and resource utilization in Merkle Tree implementations for IoT has not been thoroughly investigated.
- Dynamic Nature of IoT Data: Unlike blockchain systems, IoT networks often deal with rapidly changing, real-time data. The implications of this dynamic environment on Merkle Tree security and performance have not been adequately addressed in the current literature.
- Cross-Domain Security Implications: The security implications of using Merkle Trees across different IoT domains (e.g., healthcare, smart cities, industrial IoT) have not been comprehensively analyzed, leaving potential vulnerabilities unexplored.

This study aims to address these gaps by providing a comprehensive probabilistic analysis of data falsification in Merkle Trees, with a specific focus on IoT applications. By doing so, we seek to enhance our understanding of Merkle Tree security in resource-constrained, dynamic IoT environments and provide insights for optimizing their implementation across various IoT domains.

3. Background

In the burgeoning realms of the IoT and blockchain technology, maintaining the integrity and authenticity of data is a pivotal challenge. The IoT ecosystem is characterized by a vast array of interconnected devices, continuously generating, processing, and exchanging data. This environment, inherently diverse and dynamic, poses significant risks concerning data security, particularly in aspects of tampering and authenticity. Similarly, blockchain technology, while renowned for its robust security mechanisms, confronts challenges in ensuring the immutability and verification of the vast amounts of data processed in its networks.

3.1. Merkle Trees: Mechanism and Application

Merkle Trees, conceptualized by Ralph Merkle, present an efficient and secure method to verify the content of large data structures. A Merkle Tree is a binary tree in which every leaf node contains the hash of a data block, and every non-leaf node contains the cryptographic hash of its child nodes.

The construction and functionality of Merkle Trees are predicated on cryptographic principles, ensuring data integrity and authenticity through a series of mathematical operations. A Merkle Tree is built from the bottom up, starting with a set of data blocks (Figure 1).

Let $D = D_1, D_2, \ldots, D_n$ be the set of data blocks. Each block D_i is subjected to a cryptographic hash function H, generating a set of leaf nodes $L = L_1, L_2, \ldots, L_n$, where $L_i = H(D_i)$.

If n is odd, an additional leaf node duplicating the last hash is added to maintain a balanced tree structure. The internal nodes are then constructed by recursively hashing pairs of child nodes:

$$N_{ij} = H(N_i \parallel N_j),$$

where N_i and N_j are child nodes, and N_{ij} is their parent node. This process is iteratively applied until the root node, or Merkle Root R, is derived, representing the entire dataset's hash:

$$R = H(N_{root_{left}} \parallel N_{root_{right}}).$$

Figure 1 illustrates the fundamental structure of a Merkle Tree, showcasing its hierarchical layout from data blocks to the Merkle Root. It highlights the process of transforming data blocks through cryptographic hashing into leaf nodes, followed by the recursive construction of parent nodes, culminating in the Merkle Root.

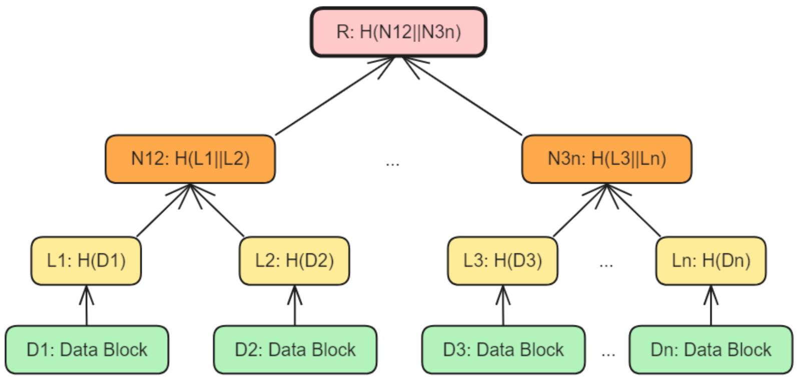

Figure 1. Merkle Tree structure overview.

The path in a Merkle Tree refers to the sequence of nodes and hashes required to verify a particular data block's integrity. For a given block D_i, its path $\mathcal{P}(D_i)$ is the set of sibling nodes and their parent hashes leading up to the root. Formally,

$$\mathcal{P}(D_i) = \{N_{sib_1}, N_{sib_2}, \ldots, N_{sib_m}\},$$

where N_{sib_j} is the sibling node and $N_{par_{j-1}}$ is the parent node, so we have

$$L_i = H(D_i); N_{par_0} = L_i;$$

$$N_{par_1} = H(N_{par_0} \parallel N_{sib_1});$$
$$N_{par_2} = H(N_{par_1} \parallel N_{sib_2});$$
$$\ldots$$
$$R = N_{par_m} = H(N_{par_{m-1}} \parallel N_{sib_m}).$$

Figure 2 explains the specific Merkle path for a leaf node, denoted as "D7". It highlights the original leaf "D7" in green, alongside all intermediate nodes generated through successive hash concatenations, revealing the pathway to the Merkle Root. The Merkle Root itself is marked in red, distinguishing it as the ultimate hash representation of the entire dataset, while the nodes integral to the Merkle path are indicated in blue, underscoring their role in verifying the presence of "D7" within the tree.

To verify the integrity and authenticity of a data block, D_i, one must reconstruct the path from L_i to the root R and compare it with the known Merkle Root. The verification process involves the following steps:

1. Initial Hashing: Compute the hash of the data block:

$$L_i = H(D_i); N_{par_0} = L_i.$$

2. Path Reconstruction: For each N_{sib_j} in $\mathcal{P}(D_i)$, compute the parent hash:

$$N_{par_j} = H(N_{par_{j-1}} \parallel N_{sib_j}).$$

3. Root Comparison: Ascend the tree, iteratively applying the hash function until the reconstructed root R' is obtained. The data block D_i is authentic and unaltered if and only if

$$R' = R.$$

This process ensures that any alteration in D_i or its path would result in a different R', thereby detecting tampering or data corruption.

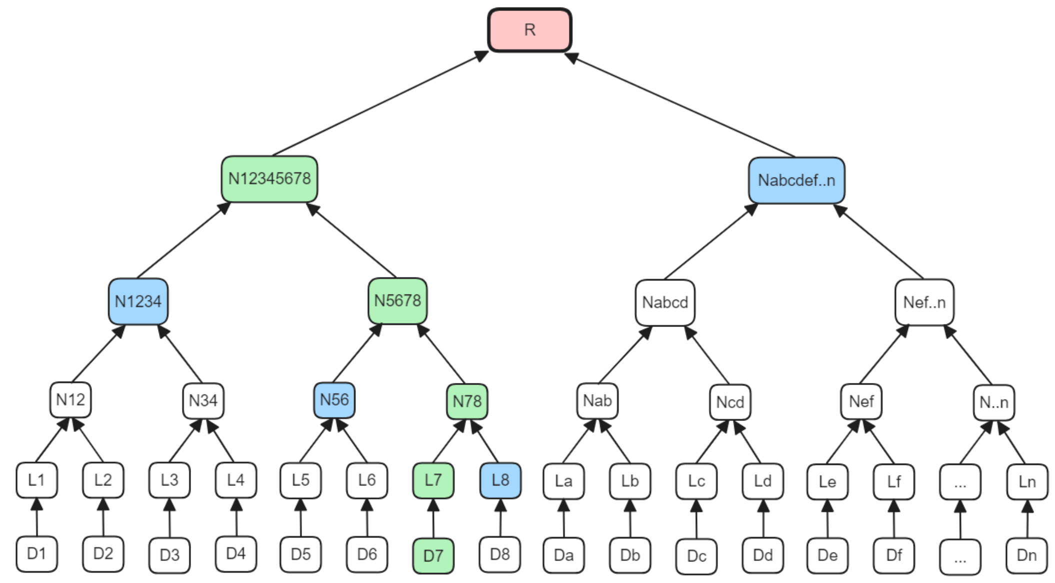

Figure 2. Merkle path visualization.

3.2. Application in IoT and Blockchain

In the context of IoT, Merkle Trees offer a method to ensure the integrity and authenticity of data across diverse and distributed devices. They enable IoT systems to efficiently validate data integrity without the need for transmitting large volumes of data, thereby reducing bandwidth and processing requirements. This is particularly beneficial in scenarios where IoT devices have limited computational and storage capabilities.

In blockchain systems, Merkle Trees are integral to the construction of blocks. Each transaction in a block is represented as a leaf node in a Merkle Tree, with the root hash being included in the block's header. This mechanism ensures that any alteration in a transaction would result in a different block header, thereby maintaining the blockchain's integrity. Furthermore, Merkle Trees enable light clients in blockchain networks to efficiently verify the existence and integrity of transactions without downloading the entire blockchain.

Thus, Merkle Trees serve as a cornerstone in ensuring data integrity and authenticity in both IoT and blockchain systems. Their application effectively addresses the challenges posed by the vast, distributed nature of these systems, providing a scalable and secure solution for data verification. This background forms the basis for our exploration into the specific application of Merkle Trees in assessing root collision and data falsification probabilities within the IoT framework, a critical aspect for advancing the security and reliability of these emerging technologies.

3.3. Problem Statement

In the context of blockchain networks and the IoT, ensuring the integrity and authenticity of data is paramount. A critical concern is the potential for data substitution within Merkle Trees, where altered data $D_i{}'$ might inadvertently or maliciously replace the original data D_i without detection. Given a Merkle Tree with a set of data blocks $D_1, D_2, \ldots, D_n$, we aim to evaluate the probability $P(R = R')$ that the Merkle Root R of the original tree remains unchanged when a specific data block, D_i, is substituted with $D_i{}'$, while all other data blocks remain constant. Formally, we define

$$P_{\text{falsification}} = P(R = R') = P(N_{par_m} = N_{par_m}{}'), \tag{1}$$

where

$$\begin{aligned}
&N_{par_m} = H(N_{par_{m-1}} \parallel N_{sib_m}) = \\
&= H(H(N_{par_{m-2}} \parallel N_{sib_{m-1}}) \parallel N_{sib_m}) = \\
&\dots \\
&= H(H(\dots H(N_{par_0} \parallel N_{sib_1}) \dots \parallel N_{sib_{m-1}}) \parallel N_{sib_m}) = \\
&= H(H(\dots H(H(D_i) \parallel N_{sib_1}) \dots \parallel N_{sib_{m-1}}) \parallel N_{sib_m})
\end{aligned}$$

and

$$\begin{aligned}
&N_{par_m}{}' = H(N_{par_{m-1}}{}' \parallel N_{sib_m}) = \\
&= H(H(N_{par_{m-2}}{}' \parallel N_{sib_{m-1}}) \parallel N_{sib_m}) = \\
&\dots \\
&= H(H(\dots H(N_{par_0}{}' \parallel N_{sib_1}) \dots \parallel N_{sib_{m-1}}) \parallel N_{sib_m}) = \\
&= H(H(\dots H(H(D_i{}') \parallel N_{sib_1}) \dots \parallel N_{sib_{m-1}}) \parallel N_{sib_m}).
\end{aligned}$$

The probability $P(R = R')$ is contingent on the cryptographic hash function H used in the Merkle Tree. Assuming H behaves as a random oracle, the probability of two different inputs producing the same hash output is negligible. Thus, we can express

$$P(H(D_i) = H(D_i{}')) = \frac{1}{2^b}, \tag{2}$$

where b is the bit length of the hash output.

The primary objective of this article is to derive a method for the precise calculation of probability (1) under assumption (2). Probability (1) quantifies the likelihood of data falsification in a Merkle Tree, thereby gauging the reliability of one of the most ubiquitous mechanisms for ensuring data integrity in blockchain technology and the Internet of Things.

To clarify our purpose, we refer to an illustrative case presented in Figure 2. Our objective is to assess the probability of substituting the data block "D7" with fraudulent data, under the premise that all Merkle path elements (indicated in blue) remain unaltered. This analysis aims to quantify the frequency at which this widely employed protocol might fail to detect data tampering. By maintaining the integrity of the Merkle path, we investigate the system's resilience to alterations at the leaf node level, thereby providing insights into the security efficacy of Merkle Trees in safeguarding data integrity against sophisticated tampering attempts.

In the following sections, we present exact formulas for computing $P_{\text{falsification}}$, as well as approximate expressions that yield an exceptionally close approximation of this probability. Furthermore, we include the results of empirical experiments that corroborate the validity of these derived expressions.

4. Theoretical Estimation of Falsification Probability

The derivation of an exact formula to estimate the probability of data falsification (1) was conducted in a stepwise manner, considering various scenarios with different lengths of the Merkle path, denoted as $\mathcal{P}(D_i) = \{N_{sib_1}, N_{sib_2}, \dots, N_{sib_m}\}$, for diverse values of m.

We commenced with the case of $m = 0$, signifying that

$$P_{\text{falsification}} = P(R = R') = P(N_{par_0} = N_{par_0}{}'),$$

where

$$N_{par_0} = L_i = H(D_i),\ N_{par_0}{}' = L_i{}' = H(D_i{}').$$

Consequently,

$$P_{\text{falsification}} = P(H(D_i) = H(D_i{}')) = \frac{1}{2^b}.$$

Thus, for $m = 0$, probability value (1) aligns with expression (2).

In addressing the scenario where $m = 1$, our analysis begins by defining the probability of data falsification, $P_{\text{falsification}}$, as the likelihood that the recalculated Merkle Root, R', matches the original Merkle Root, R. This probability is contingent upon the equality of the hash values at the first level of the Merkle path, denoted as N_{par_1} and $N_{par_1}{}'$, respectively.

The hash value N_{par_1} is computed by concatenating the hash of the original data block, $N_{par_0} = L_i = H(D_i)$, with the hash of its sibling node, N_{sib_1}, and then applying the hash function H to this concatenated string. Similarly, $N_{par_1}{}'$ is derived by hashing the concatenation of $N_{par_0}{}' = L_i{}' = H(D_i{}')$ and N_{sib_1}. The event $N_{par_1} = N_{par_1}{}'$ signifies a match in these hash values, implying that the path leading to the Merkle Root in both the original and altered trees remains unchanged.

This exact match occurs under two distinct conditions: The first is when the hash codes of the original and altered data blocks are identical, $H(D_i) = H(D_i{}')$, which happens with a probability of $\frac{1}{2^b}$, where b represents the hash length in bits. This reflects the inherent security of the hash function, assuming it behaves as a perfect hash function with a uniform distribution over its output space.

Secondly, in the case where $N_{par_0} \neq N_{par_0}{}'$—that is, when the hash codes for D_i and $D_i{}'$ do not match, which occurs with the complementary probability of $\left(1 - \frac{1}{2^b}\right)$—the probability of still achieving a matching hash at the next level, $H(N_{par_0} \parallel N_{sib_1}) = H(N_{par_0}{}' \parallel N_{sib_1})$, is considered. This scenario accounts for the possibility of a hash collision at the subsequent level of the tree, despite the initial data discrepancy. The likelihood of such a collision is given by $\left(1 - \frac{1}{2^b}\right)\frac{1}{2^b}$, reflecting the reduced probability due to the initial non-matching hashes, yet allowing for a coincidental match at the next hashing step.

Figure 3 shows an example of a Merkle Tree, illustrating the comparison between an original data block D_i and its altered version $D_i{}'$. It details the hashing process from these data blocks to their respective leaf nodes L_i and $L_i{}'$, their combination with a sibling node N_{sib_1}, and the subsequent generation of parent nodes N_{par_1} and $N_{par_1}{}'$, leading to the original and recalculated Merkle Roots R and R', respectively.

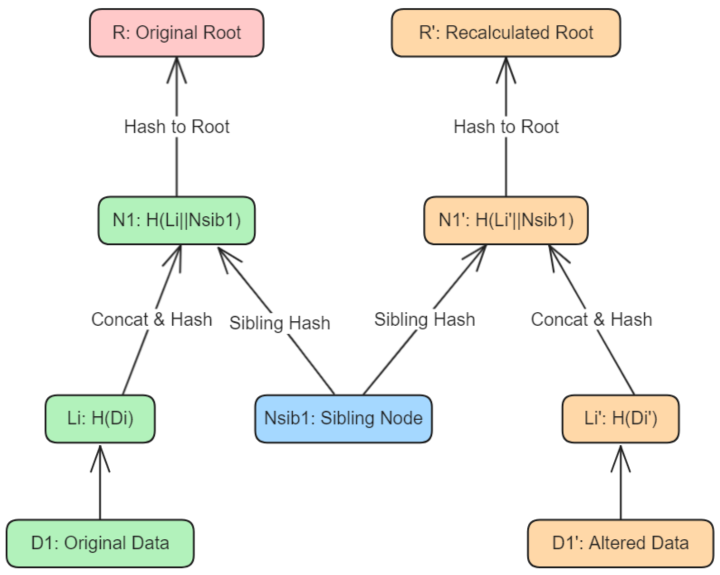

Figure 3. Merkle Tree probability analysis.

This visualization aids in understanding the derivation of the formula by demonstrating two critical pathways: one where the hash of the original data and its altered counterpart yield identical parent nodes under specific probabilistic conditions, and another where they do not. It underscores the concept of hash function security, highlighting how the probability of a false match ($R = R'$) depends on the hash values' uniqueness and the occurrence of hash collisions.

The final expression for calculating probability (1) for $m = 1$ becomes

$$\begin{aligned} &P_{\text{falsification}} = P\big(H(D_i) = H(D_i{}')\big)+ \\ &+P\big(H(D_i) \neq H(D_i{}')\big) \cdot P\big(H(N_{par_0} \parallel N_{sib_1}) = H(N_{par_0}{}' \parallel N_{sib_1})\big) = \\ &= \tfrac{1}{2^b} + \Big(1 - \tfrac{1}{2^b}\Big)\tfrac{1}{2^b}. \end{aligned}$$

Extending similar reasoning to the case of $m = 2$, we obtain

$$\begin{aligned} &P_{\text{falsification}} = P\big(H(D_i) = H(D_i{}')\big)+ \\ &+P\big(H(D_i) \neq H(D_i{}')\big) \times P\big(H(N_{par_0} \parallel N_{sib_1}) = H(N_{par_0}{}' \parallel N_{sib_1})\big)+ \\ &+P\big(H(D_i) \neq H(D_i{}')\big) \times P\big(H(N_{par_0} \parallel N_{sib_1}) \neq H(N_{par_0}{}' \parallel N_{sib_1})\big)\times \\ &\times P\big(H(H(N_{par_0} \parallel N_{sib_1}) \parallel N_{sib_2}) = H(H(N_{par_0}{}' \parallel N_{sib_1}) \parallel N_{sib_2})\big) = \\ &= \tfrac{1}{2^b} + \Big(1 - \tfrac{1}{2^b}\Big)\tfrac{1}{2^b} + \Big(1 - \tfrac{1}{2^b}\Big)\Big(1 - \tfrac{1}{2^b}\Big)\tfrac{1}{2^b}. \end{aligned}$$

Generalizing this formula for any positive integer m, we derive the general formula

$$\begin{aligned} &P_{\text{falsification}} = \frac{1}{2^b} + \left(1 - \frac{1}{2^b}\right)\frac{1}{2^b} + \left(1 - \frac{1}{2^b}\right)\left(1 - \frac{1}{2^b}\right)\frac{1}{2^b} + \ldots + \\ &+\underbrace{\left(1 - \frac{1}{2^b}\right)\left(1 - \frac{1}{2^b}\right)\ldots\left(1 - \frac{1}{2^b}\right)}_{\text{m times}}\frac{1}{2^b} = \frac{1}{2^b} + \sum_{k=1}^{m}\left(1 - \frac{1}{2^b}\right)^k \frac{1}{2^b}. \end{aligned}$$

The sum on the right side of the last expression can be simplified using the rule for calculating the sum of the first m terms of a geometric progression.

Let g be the first term of the geometric progression, and z the common ratio, i.e., the factor by which each term is multiplied to obtain the next one. Then, the formula for the sum of the first m terms of a geometric progression is

$$G_m = g \cdot \frac{1 - z^m}{1 - z}.$$

In our case, $g = z = 1 - \frac{1}{2^b}$; thus,

$$G_m = \sum_{k=1}^{m}\left(1 - \frac{1}{2^b}\right)^k = \left(1 - \frac{1}{2^b}\right)\frac{1 - \left(1 - \frac{1}{2^b}\right)^m}{1 - \left(1 - \frac{1}{2^b}\right)} = \left(-1 + 2^b\right)\left(1 - \left(1 - \frac{1}{2^b}\right)^m\right).$$

Substituting G_m into the formula for $P_{\text{falsification}}$, we obtain

$$\begin{aligned} P_{\text{falsification}} &= \frac{1}{2^b} + \sum_{k=1}^{m}\left(1 - \frac{1}{2^b}\right)^k \frac{1}{2^b} = \\ &= \frac{1}{2^b} + \frac{1}{2^b}\left(-1 + 2^b\right)\left(1 - \left(1 - \frac{1}{2^b}\right)^m\right) = \\ &= \frac{1}{2^b} + \left(1 - \frac{1}{2^b}\right)\left(1 - \left(1 - \frac{1}{2^b}\right)^m\right). \end{aligned} \qquad (3)$$

Derived Formula (3) allows for an exact calculation of the probability of data falsification when using the Merkle path $\mathcal{P}(D_i) = \{N_{sib_1}, N_{sib_2}, \ldots, N_{sib_m}\}$ and under the assumption of truth in (2). As evident from formula (3), the probability $P_{\text{falsification}}$ increases as m, the number of elements in the Merkle path $\mathcal{P}(D_i)$, increases.

To derive an approximate formula, we note that when b takes large values, the magnitude of $\frac{1}{2^b}$ becomes very small. This allows for an approximation using the initial terms of the Taylor series for $\exp(x)$, where $x = \frac{-1}{2^b}$. Consequently, the approximation can be expressed as $\exp\left(\frac{-1}{2^b}\right) \approx 1 - \frac{1}{2^b}$.

Substituting this approximation into Formula (3), we obtain

$$P_{\text{falsification}} \approx \frac{1}{2^b} + \exp\left(\frac{-1}{2^b}\right) - \exp\left(\frac{-m-1}{2^b}\right). \tag{4}$$

5. Verification of Theoretical Formulas through Empirical Testing

In the experimental phase of our research, we focused on empirically validating theoretical Formula (3) derived in the previous sections. To achieve this, we developed a Python program, as referenced in [19], designed to estimate the probability of data falsification and compare it with our theoretical calculations.

The experimental design was meticulously crafted to ensure the robust validation of our theoretical models. Key characteristics of our experimental setup include

- Experimental Parameters:
  - Hash lengths (b): 2, 4, 6, 8, and 10 bits;
  - Merkle path lengths (m): 10, 50, 100, 500, and 1000 elements;
  - Number of individual experiments per parameter set: 1000;
  - Number of repetitions: 100.
- Hash Generation: We utilized the SHA256 hashing algorithm, implemented in Python's hashlib module, to generate hashes of data. The generate_hash function takes a data string and returns a truncated hash of a specified length.
- Random Data Generation: The generate_random_data function creates random strings of a specified length, combining ASCII letters and digits. This function is crucial for simulating various data inputs in the experiment.
- Merkle Root Calculation: The calculate_merkle_root function computes the root hash for given data and a Merkle path. It iteratively hashes the data with each element of the path, simulating the process of ascending a Merkle Tree.
- Theoretical Probability Calculations: We implemented two different theoretical probability functions, (3) and (4), to provide a comprehensive theoretical framework for comparison with empirical results.
- Experiment Execution: The run_experiment function conducts the empirical testing. It generates a random Merkle path, calculates the root hash, and then counts the number of matches found when recalculating the root hash with new random data. This process is repeated across a specified number of experiments and iterations.
- Graphical Representation: We used matplotlib to plot the results. The empirical probabilities of data falsification for different hash lengths and Merkle path sizes were plotted alongside the theoretical estimates. This visual representation aids in directly comparing empirical data with theoretical predictions.

This comprehensive experimental framework allows for a thorough exploration of Merkle Tree security across a wide range of parameters, providing robust empirical support for our theoretical models and insights into their practical applications in IoT and blockchain systems.

In this study, a comprehensive experimental approach was adopted to rigorously assess the accuracy of the approximation formula in comparison to the exact formula. For each set of parameters b and m, a total of 1000 individual experiments were conducted, and this process was repeated 100 times. This methodology was meticulously designed to amass a substantial volume of data, thereby facilitating a robust statistical analysis.

The rationale behind this extensive experimental repetition lies in its ability to mitigate the impact of random variations and anomalies, ensuring the reliability and validity of the results. By aggregating data from 100,000 experiments for each parameter set, this study aimed to achieve a high level of precision in its findings, thereby providing a solid foundation for drawing statistically significant conclusions.

Figure 4 presents the empirical results of our experiments, illustrating the relationship between the probability of data falsification and the variables b (hash length) and m (number of elements in the Merkle path).

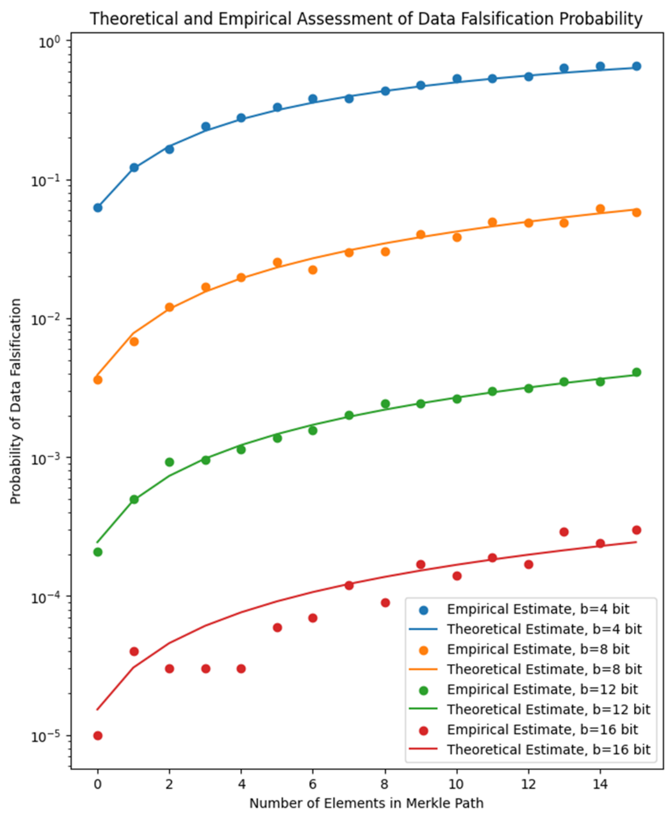

Figure 4. Theoretical and empirical assessment of data falsification probability.

The graph reveals two key trends:

1. Decrease in Falsification Probability with Increasing b: As the hash length b increases, the probability of data falsification decreases significantly. This trend is consistent with cryptographic principles, where longer hash lengths correspond to a larger space of possible hash values, thereby reducing the likelihood of hash collisions. In the context of Merkle Trees, a longer hash implies a lower probability that two different data inputs will yield the same Merkle Root, enhancing the security and integrity of the data.
2. Increase in Falsification Probability with Increasing m: Conversely, the graph shows an increase in the probability of data falsification with an increasing number of elements m in the Merkle path. This observation can be attributed to the cumulative effect of hash collisions along the Merkle path. As the number of elements in the path increases, the probability of encountering a hash collision at some point in the path also increases, even if each individual hash operation remains secure. This effect highlights a trade-off in Merkle Tree design: while a longer path provides a more detailed verification trail, it also slightly increases the overall risk of hash collisions.

The empirical results align well with the theoretical predictions, confirming the validity of our theoretical models. The graph serves as a crucial tool for understanding the dynamics of data integrity in systems employing Merkle Trees. For blockchain and IoT applications, where data security is paramount, these findings underscore the importance of optimizing hash length and Merkle path length to balance security and computational efficiency.

In the second part of our experimental analysis, we focused on quantifying the discrepancy between the exact and approximate probabilities of data falsification in Merkle Trees. This investigation aimed to validate the precision of our approximate Formula (4) against exact calculations (3), particularly under varying conditions of hash length b and Merkle path length m. The results of the experiment are presented in Table 1.

Table 1. Absolute Difference Between the Values Obtained Using Formulas (3) and (4).

b	m	Difference	2^{-b}
2	10	7.11×10^{-3}	2.50×10^{-1}
2	50	2.88×10^{-2}	2.50×10^{-1}
2	100	2.88×10^{-2}	2.50×10^{-1}
2	500	2.88×10^{-2}	2.50×10^{-1}
2	1000	2.88×10^{-2}	2.50×10^{-1}
4	10	9.24×10^{-3}	6.25×10^{-2}
4	50	2.16×10^{-3}	6.25×10^{-2}
4	100	1.58×10^{-3}	6.25×10^{-2}
4	500	1.91×10^{-3}	6.25×10^{-2}
4	1000	1.91×10^{-3}	6.25×10^{-2}
6	10	1.02×10^{-3}	1.56×10^{-2}
6	50	2.71×10^{-3}	1.56×10^{-2}
6	100	2.43×10^{-3}	1.56×10^{-2}
6	500	9.76×10^{-5}	1.56×10^{-2}
6	1000	1.21×10^{-4}	1.56×10^{-2}
8	10	7.30×10^{-5}	3.91×10^{-3}
8	50	3.12×10^{-4}	3.91×10^{-3}
8	100	5.13×10^{-4}	3.91×10^{-3}
8	500	5.33×10^{-4}	3.91×10^{-3}
8	1000	1.45×10^{-4}	3.91×10^{-3}
10	10	4.72×10^{-6}	9.77×10^{-4}
10	50	2.27×10^{-5}	9.77×10^{-4}
10	100	4.32×10^{-5}	9.77×10^{-4}
10	500	1.46×10^{-4}	9.77×10^{-4}
10	1000	1.79×10^{-4}	9.77×10^{-4}

As part of our experimental investigations, Table 1 plays a crucial role in illustrating the comparative analysis between the exact and approximate probabilities of data falsification across a range of hash lengths b and Merkle path lengths m. This table meticulously documents the absolute differences between these probabilities, alongside collision probability values (2), providing a comprehensive overview of the approximation's accuracy under various conditions.

Table 1 is structured to showcase the absolute differences in falsification probability as calculated by the exact formula and its approximation. The columns represent different hash lengths b, ranging from 2 to 10 bits, and various Merkle path lengths m, demonstrating the impact of these parameters on the probability of data falsification. The last column presents collision probability (2), serving as a benchmark to evaluate the significance of the differences observed.

The results depicted in Table 1 reveal a critical insight: as the hash length b increases, the discrepancy between the exact and approximate probabilities diminishes, affirming the high precision of the approximation across a spectrum of scenarios. Notably, this discrepancy consistently remains below the collision probability threshold, indicating that the approximation does not introduce significant errors into the evaluation of Merkle Root coincidence probabilities.

This observation is particularly relevant for larger values of b and where direct numerical experimentation becomes impractical. The data demonstrate that even in these extended scenarios, the approximation maintains its reliability, offering a robust tool for assessing data falsification probabilities without compromising on accuracy.

Thus, the analysis presented in Table 1 underscores the practical applicability and validity of the approximate formula for evaluating the probability of data falsification in Merkle Trees. By demonstrating that the approximation introduces negligible errors that are always below the collision probability threshold, we validate its use in a wide range of conditions. This finding is instrumental for system designers and researchers, providing a simplified yet accurate method for assessing data integrity in decentralized systems. In conclusion, the approximation not only simplifies the analytical process but also ensures computational efficiency, making it a valuable asset for optimizing data security mechanisms in blockchain and IoT systems without sacrificing the fidelity of probabilistic assessments.

6. Discussion

The findings of our study on Merkle Tree security in blockchain systems reveal a complex interplay between security, efficiency, and practicality. Our analysis of root collision probabilities in Merkle Trees provides several key insights that have significant implications for blockchain technology and its applications.

Firstly, our results demonstrate a clear relationship between hash length, path length, and collision probability in Merkle Trees. The inverse relationship between hash length and collision probability underscores the critical importance of selecting appropriate hash functions in blockchain implementations. This finding provides concrete guidance for system designers in balancing security requirements with computational resources.

Secondly, the observed increase in collision probability with increasing path length in Merkle Trees highlights a potential vulnerability in large-scale blockchain systems. This insight is particularly relevant as blockchain networks continue to grow and handle increasingly complex data structures. It suggests that careful consideration must be given to the depth of Merkle Trees in blockchain designs to maintain optimal security levels.

Our empirical validation of the theoretical model not only confirms these relationships but also provides a practical tool for assessing the security of Merkle Tree implementations in real-world scenarios. The convergence of theoretical calculations and experimental data lends credibility to our findings and offers a reliable framework for future security analyses in blockchain systems.

6.1. Comparative Analysis with Existing Literature

Our study significantly advances the understanding of Merkle Tree security in blockchain systems, building upon and extending previous research in this field.

- Xu et al. [14] proposed a dynamic Fully Homomorphic encryption-based Merkle Tree (FHMT) for lightweight streaming authenticated data structures. While their work focused on balancing performance between client and server, our study complements this by providing a comprehensive probabilistic analysis of data falsification. This analysis is crucial for assessing the security of such lightweight structures in blockchain contexts, offering insights that go beyond performance considerations to address fundamental security aspects.
- In comparison to the work of Zhu et al. [2], who introduced an improved convolution Merkle Tree-based blockchain scheme for secure electronic medical record storage, our research offers a more generalized approach. While Zhu et al. focused on a specific application in healthcare, our probabilistic model provides insights applicable across various blockchain domains, offering a broader perspective on Merkle Tree security that can be adapted to different use cases.
- Mitra et al. [10,11] introduced innovative approaches like Polar Coded Merkle Tree (PCMT) and Graph Coded Merkle Tree (GCMT) to improve the detection of Data

Availability (DA) attacks in blockchain systems. Our work extends beyond their focus on DA attacks, providing a comprehensive analysis of falsification probabilities that can be applied to assess the robustness of these advanced Merkle Tree variants. This broader analysis contributes to a more holistic understanding of Merkle Tree security across different implementations.

- Sarkar's study [13] on domain extenders for collision-resistant hash functions improved upon the Merkle–Damgård iteration. Our research complements this by explicitly quantifying the impact of hash length and Merkle path length on security, providing practical insights for implementing such improvements in blockchain systems. This quantitative approach allows for more precise security assessments in real-world applications.
- Our previous work [19] laid the groundwork for understanding data falsification probabilities in Merkle Trees, particularly in IoT contexts. The current study significantly expands on this foundation, offering a more in-depth analysis specifically tailored to blockchain technologies. We have refined our mathematical models, extended the range of parameters considered, and provided a more rigorous empirical validation. Unlike our previous study, which focused on smaller hash lengths and path lengths, this work considers parameters more relevant to actual blockchain implementations, including hash lengths up to 256 bits and larger Merkle Tree structures.

A key advancement in our current research is the incorporation of the "Birthday Paradox" strategy in analyzing collision probabilities, an aspect not explored in the other existing literature on Merkle Trees in blockchain. This novel approach provides insights into potential vulnerabilities that might arise in scenarios where an attacker has more flexibility in selecting data pairs, offering a new perspective on Merkle Tree security.

Furthermore, our study bridges the gap between theoretical models and practical applications in blockchain technologies. By providing both exact and approximate formulas for falsification probability, we offer unique tools for system designers to evaluate and optimize Merkle Tree implementations in blockchain environments. This practical focus distinguishes our work from more theoretical studies in the field.

In conclusion, while previous research has made significant contributions to understanding specific aspects of Merkle Tree security and applications, our current study provides a comprehensive, blockchain-focused analysis of collision probabilities. This work not only builds upon our previous research but also complements and extends the existing body of knowledge, offering valuable insights for enhancing the security and efficiency of blockchain systems employing Merkle Trees.

6.2. Significance and Real-World Implications of Results

The significance of our findings extends beyond theoretical considerations, offering crucial insights for real-world blockchain implementations. Our analysis reveals that for practically significant scenarios in modern blockchain systems, the probability of root collisions in Merkle Trees remains critically low, posing no substantial threat to security under normal operating conditions.

Consider, for instance, the Ethereum blockchain, which employs the Keccak-256 hashing function with a 256-bit output. In typical Ethereum implementations, the depth of Merkle Trees rarely exceeds 32 levels. Applying our derived formulas to these parameters, we find that the probability of a root collision is approximately 2^{-255}, an astronomically small number. This result underscores the robust security provided by properly configured Merkle Trees in real-world blockchain systems.

However, our study also highlights potential vulnerabilities that could arise in certain edge cases or future scenarios. As blockchain networks continue to scale and handle increasingly complex data structures, the depth of Merkle Trees may grow. Our analysis shows that for extremely large Merkle Trees with depths approaching 2^{128} (a number far beyond current practical limits), the collision probability could theoretically reach about 63%.

While such scenarios are not currently relevant, this finding emphasizes the importance of ongoing vigilance and adaptation in blockchain security as the technology evolves.

The "Birthday Paradox" strategy explored in our research further illuminates potential attack vectors that might not be immediately apparent. In scenarios where an attacker can manipulate both the data and the Merkle Root, the effective security of the hash function is essentially halved. For instance, with a 256-bit hash, an attacker would need to compute approximately 2^{128} hashes to find a collision with 50% probability. While this remains computationally infeasible with current technology, it underscores the need for continued advancements in hash function security to stay ahead of potential future threats.

These insights are particularly valuable for blockchain developers and security researchers. They provide a quantitative basis for security assessments and can guide decisions on hash function selection, Merkle Tree depth limitations, and overall system architecture. Moreover, our findings contribute to the ongoing dialog on the long-term security of blockchain systems, especially in light of advancing computational capabilities and the looming prospect of quantum computing.

In essence, while our results confirm the current security of Merkle Trees in blockchain applications, they also serve as a forward-looking tool for anticipating and mitigating potential future vulnerabilities. This dual perspective—affirming current security while preparing for future challenges—is crucial for the continued evolution and robustness of blockchain technology.

6.3. Practical Implications

From a practical standpoint, our research offers valuable guidelines for optimizing the security and efficiency of blockchain and IoT systems. The balance between hash length and Merkle path length is a key consideration for system designers, as it directly impacts the probability of data falsification and, consequently, the overall system integrity. Our findings suggest that while longer hashes enhance security, they also entail computational overhead, necessitating a judicious choice based on the specific requirements of the application.

6.4. Future Research Directions

The complexity observed in the interaction between hash length and Merkle path length opens avenues for further research. Investigating the optimal combinations of these parameters for different application scenarios could lead to more refined guidelines for system design. Additionally, exploring the impact of different hashing algorithms on the probability of data falsification could provide deeper insights into the security aspects of Merkle Trees.

7. Extended Security Analysis and Performance Considerations

7.1. Threat Models and Attack Vectors

While Merkle Trees provide robust security in blockchain systems, it is crucial to consider potential vulnerabilities and attack vectors. Our analysis identifies several threat models:

- Collision Attacks: An adversary attempts to find two different datasets that produce the same Merkle Root. The probability of success increases with Merkle Tree depth, as demonstrated in our collision probability analysis.
- Second Preimage Attacks: An attacker, given a specific data block and its hash, tries to find a different block with the same hash. This attack becomes more feasible as the Merkle Tree grows in depth.
- Length Extension Attacks: These exploit the iterative nature of some hash functions, potentially allowing an attacker to append additional data to a Merkle Tree without knowing the original data.
- Denial of Service (DoS) Attacks: An attacker could potentially exploit the computational requirements of verifying deep Merkle Trees to overwhelm system resources.

- Time-Memory Trade-Off Attacks: Leveraging precomputed hash values, an attacker might accelerate the process of finding collisions, particularly in systems with predictable data structures.

Each of these attack vectors underscores the importance of careful parameter selection in Merkle Tree implementations, balancing security with performance requirements.

7.2. Comparative Analysis of Data Integrity Methods

To contextualize the security of Merkle Trees, we compare them with other cryptographic methods for ensuring data integrity:

- Digital Signatures: While providing strong authenticity guarantees, digital signatures incur higher computational costs and larger storage requirements compared to Merkle Trees, especially for large datasets.
- Hash Lists: Simpler than Merkle Trees, hash lists offer similar integrity guarantees but lack the efficiency in proving the inclusion of specific data items without revealing the entire list.
- Authenticated Encryption: This method combines confidentiality with integrity but is less suitable for scenarios requiring selective data verification, a strength of Merkle Trees.
- Blockchain without Merkle Trees: Some blockchain implementations use alternative structures, such as Patricia Tries. While these can offer advantages in certain scenarios, they often lack the efficient partial verification capabilities of Merkle Trees.
- Zero-Knowledge Proofs: These provide strong privacy guarantees alongside integrity but at the cost of increased computational complexity compared to Merkle Trees.

Merkle Trees stand out in their ability to efficiently prove the inclusion of data in large sets without revealing the entire dataset, a crucial feature for blockchain systems. However, they may be less suitable for applications requiring frequent updates to the entire dataset.

7.3. Performance Metrics and Scalability Analysis

Understanding the performance implications of Merkle Tree implementations is crucial for their practical application in blockchain systems. We analyze key performance metrics:

1. Computational Overhead:
    - Tree Construction: O(n log n), where n is the number of data blocks.
    - Verification: O(log n) for proving the inclusion of a single data block.
    - Root Calculation: O(log n) when updating a single leaf node.
2. Storage Requirements:
    - Space Complexity: O(n) for storing the entire tree.
    - Proof Size: O(log n) for generating a Merkle proof.
3. Scalability:
    - Our model shows that increasing the Merkle Tree depth from 20 to 30 levels results in only a 50% increase in verification time, demonstrating good scalability for moderately sized increases.
    - However, extremely deep trees (e.g., beyond 100 levels) may incur significant performance penalties, with verification times increasing by orders of magnitude.
4. Network Bandwidth:
5. Merkle proofs require transmitting only O(log n) hashes, significantly reducing bandwidth requirements compared to transmitting entire datasets.
6. Parallelization Potential:
    - Tree construction and verification processes can be parallelized, offering performance improvements on multi-core systems.

These metrics highlight the efficiency of Merkle Trees in scenarios requiring the frequent verification of data integrity, particularly in distributed systems like blockchains.

However, they also underscore the need for the careful consideration of tree depth to maintain performance in large-scale implementations.

7.4. Application and Advantages in IoT Contexts

The findings of our study on Merkle Tree security have significant implications for Internet of Things (IoT) applications, where data integrity and efficient verification are paramount. The IoT ecosystem, characterized by its vast network of interconnected devices with varying computational capabilities, can particularly benefit from our insights into Merkle Tree security and performance.

- Our analysis of collision probabilities enables IoT developers to fine-tune Merkle Tree parameters for devices with limited computational resources. By understanding the trade-offs between hash length, tree depth, and security, developers can implement Merkle Trees that provide adequate security while minimizing computational overhead. For instance, in a smart home network, lightweight sensors can use shallower Merkle Trees with carefully chosen hash lengths to ensure data integrity without overstressing their limited processors.
- As IoT networks grow to encompass thousands or even millions of devices, our scalability analysis becomes crucial. The logarithmic verification time of Merkle Trees (O(log n)) is particularly advantageous in large IoT deployments. For example, in smart city applications, where data from numerous sensors need to be verified, our model allows for the design of Merkle Tree structures that maintain efficiency even as the network expands.
- Edge computing in IoT often requires the rapid verification of data integrity. Our performance metrics demonstrate that Merkle Trees offer a balance between security and speed, crucial for real-time decision-making in edge devices. This is particularly relevant in industrial IoT settings, where the quick verification of sensor data integrity can be critical for operational safety and efficiency.
- In many IoT scenarios, data from multiple sources are aggregated before transmission or storage. Our analysis of the "Birthday Paradox" strategy in Merkle Trees provides insights into securing these aggregation processes. By understanding potential vulnerabilities, IoT system architects can implement safeguards against sophisticated attacks that might target data aggregation points.
- Many IoT applications operate in environments with limited bandwidth. The efficiency of Merkle proofs in terms of data transmission (O(log n) hashes) is particularly beneficial here. Our study provides guidelines for optimizing these proofs, allowing IoT devices to verify data integrity with minimal data exchange, crucial in applications like remote environmental monitoring or agricultural IoT systems.
- IoT encompasses a wide range of devices with varying security requirements. Our comparative analysis of Merkle Trees against other integrity verification methods helps in selecting the most appropriate approach for different IoT contexts. For instance, high-security IoT applications in healthcare might benefit from the strong integrity guarantees of Merkle Trees, while simpler consumer IoT devices might opt for less complex alternatives based on our performance analysis.
- Our forward-looking analysis, considering potential future attack vectors and the scalability of Merkle Trees, aids in designing IoT systems that remain secure as technology evolves. This is particularly important in long-term IoT deployments, such as smart infrastructure projects, where the security architecture must withstand emerging threats over extended periods.

By applying our findings to IoT contexts, developers and system architects can create more secure, efficient, and scalable IoT networks. The insights from our study enable the tailoring of Merkle Tree implementations to the unique challenges of IoT, balancing the needs for data integrity, computational efficiency, and scalability across a diverse range of devices and applications. This approach not only enhances current IoT security practices but also lays the groundwork for robust, future-proof IoT ecosystems.

8. Conclusions

In conclusion, our comprehensive analysis blending theoretical models with empirical and numerical validations offers a nuanced understanding of data integrity in Merkle Trees. The insights gained from this study are instrumental in advancing the security frameworks of blockchain and IoT systems, contributing significantly to the field of data integrity and security.

Author Contributions: Conceptualization and methodology, writing—original draft preparation, O.K.; funding acquisition, resources, A.R.; project administration, A.Y.; software and validation, writing—review and editing, K.K.; data curation, validation, D.K.; investigation, formal analysis, O.D. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the Proxima Labs, 1501 Larkin Street, Suite 300, San Francisco, USA.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The authors were employed by the company Proxima Labs, 1501 Larkin Street, Suite 300, San Francisco, USA. The authors declare that this study received funding from Proxima Labs. The funder was not involved in the study design; the collection, analysis, or interpretation of the data; the writing of this article; or the decision to submit it for publication.

References

1. Mishra, N.; Hafizul Islam, S.; Zeadally, S. A Survey on Security and Cryptographic Perspective of Industrial-Internet-of-Things. *Internet Things* **2024**, *25*, 101037. [CrossRef]
2. Zhu, H.; Guo, Y.; Zhang, L. An Improved Convolution Merkle Tree-Based Blockchain Electronic Medical Record Secure Storage Scheme. *J. Inf. Secur. Appl.* **2021**, *61*, 102952. [CrossRef]
3. Wang, J.; Chen, J.; Ren, Y.; Sharma, P.K.; Alfarraj, O.; Tolba, A. Data Security Storage Mechanism Based on Blockchain Industrial Internet of Things. *Comput. Ind. Eng.* **2022**, *164*, 107903. [CrossRef]
4. Ahmed, S.F.; Alam, Md.S.B.; Hoque, M.; Lameesa, A.; Afrin, S.; Farah, T.; Kabir, M.; Shafiullah, G.; Muyeen, S.M. Industrial Internet of Things Enabled Technologies, Challenges, and Future Directions. *Comput. Electr. Eng.* **2023**, *110*, 108847. [CrossRef]
5. Nisha; Urvashi A Systematic Literature Review of Internet of Video Things: Trends, Techniques, Datasets, and Framework. *Internet Things* **2023**, *24*, 100906. [CrossRef]
6. Freitag, C.; Ghoshal, A.; Komargodski, I. Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions. In Proceedings of the Advances in Cryptology—CRYPTO 2022, Santa Barbara, CA, USA, 15–18 August 2022; Dodis, Y., Shrimpton, T., Eds.; Springer Nature: Cham, Switzerland, 2022; pp. 131–160.
7. Ghoshal, A.; Komargodski, I. On Time-Space Tradeoffs for Bounded-Length Collisions in Merkle-Damgård Hashing. In Proceedings of the Advances in Cryptology—CRYPTO 2022, Santa Barbara, CA, USA, 15–18 August 2022; Dodis, Y., Shrimpton, T., Eds.; Springer Nature: Cham, Switzerland, 2022; pp. 161–191.
8. Hu, K.; Zhang, Z.; Guo, K. Breaking the Binding: Attacks on the Merkle Approach to Prove Liabilities and Its Applications. *Comput. Secur.* **2019**, *87*, 101585. [CrossRef]
9. Kumari, S.; Singh, M.; Singh, R.; Tewari, H. Signature Based Merkle Hash Multiplication Algorithm to Secure the Communication in IoT Devices. *Knowl. Based Syst.* **2022**, *253*, 109543. [CrossRef]
10. Mitra, D.; Tauz, L.; Dolecek, L. Polar Coded Merkle Tree: Improved Detection of Data Availability Attacks in Blockchain Systems. In Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 26 June–1 July 2022; pp. 2583–2588.
11. Mitra, D.; Tauz, L.; Dolecek, L. Graph Coded Merkle Tree: Mitigating Data Availability Attacks in Blockchain Systems Using Informed Design of Polar Factor Graphs. *IEEE J. Sel. Areas Inf. Theory* **2023**, *4*, 434–452. [CrossRef]
12. Rao, L.; Zhang, H.; Tu, T. Dynamic Outsourced Auditing Services for Cloud Storage Based on Batch-Leaves-Authenticated Merkle Hash Tree. *IEEE Trans. Serv. Comput.* **2020**, *13*, 451–463. [CrossRef]
13. Sarkar, P. Domain Extender for Collision Resistant Hash Functions: Improving upon Merkle–Damgård Iteration. *Discret. Appl. Math.* **2009**, *157*, 1086–1097. [CrossRef]
14. Xu, J.; Wei, L.; Zhang, Y.; Wang, A.; Zhou, F.; Gao, C. Dynamic Fully Homomorphic Encryption-Based Merkle Tree for Lightweight Streaming Authenticated Data Structures. *J. Netw. Comput. Appl.* **2018**, *107*, 113–124. [CrossRef]
15. Buccafurri, F.; De Angelis, V.; Lazzaro, S. MQTT-A: A Broker-Bridging P2P Architecture to Achieve Anonymity in MQTT. *IEEE Internet Things J.* **2023**, *10*, 15443–15463. [CrossRef]

16. Buccafurri, F.; De Angelis, V.; Lazzaro, S. MQTT-I: Achieving End-to-End Data Flow Integrity in MQTT. *IEEE Trans. Dependable Secur. Comput.* **2024**, 1–18. [CrossRef]
17. Kaur, B.; Dadkhah, S.; Shoeleh, F.; Neto, E.C.P.; Xiong, P.; Iqbal, S.; Lamontagne, P.; Ray, S.; Ghorbani, A.A. Internet of Things (IoT) Security Dataset Evolution: Challenges and Future Directions. *Internet Things* **2023**, *22*, 100780. [CrossRef]
18. Li, C.; Wang, J.; Wang, S.; Zhang, Y. A Review of IoT Applications in Healthcare. *Neurocomputing* **2024**, *565*, 127017. [CrossRef]
19. 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**, 101193. [CrossRef]