Grenze International Journal of Engineering and Technology, June Issue

Merkle Tree-based Framework for Real-Time Data Integrity Visualization

Madhu Bhan[1], Siddharth Kumar Maharana[2] and Saravanan C[3]
[1-2]Amity University Bengaluru, Karnataka, India
Email: mbhan@blr.amity.edu, siddharthk.maharana@gmail.com
[3]R V College of Engineering, Bengaluru, India
Email: saravananc@rvce.edu.in

***Abstract*— In the context of distributed ledger technologies, the guarantee of data immutability is of the utmost importance. Even though the Merkle Tree design provides a mathematically careful guarantee of integrity, the practical verification processes are often still hidden from their creators and consumers. In this paper, an interactive verification structure, known as the Merkle Tree Visualizer, is proposed. The system also proposes an immediate graphical user interface that facilitates the recursive hashing of Ledger Entries using the hash-value algorithm, SHA-256. This framework is a live visual change of the so-called Avalanche Effect, unlike the traditional command line tools, where users must wait before the next line or output appears. The paper outlines how the system is implemented based on a reactive front-end architecture and how effectively the system is implemented in the demystification of cryptographic proofs. Findings show that dynamic visualization significantly reduces the cognitive barrier to perceiving data integrity and creates a solid foundation for future benchmarking of post-Quantum cryptography standards.**

***Index Terms*— Data Integrity, Cryptography, Merkle Tree, Blockchain, Tampering, Hashing, Ledger Verification.**

I. INTRODUCTION

The fast development of blockchain systems has made trustless verification the center of modern computer science. The core component in this paradigm is the Merkle Tree, which is a binary hash tree used to verify large datasets efficiently and does not require the download of the complete ledger. However, there is still a substantive difference between theoretical understanding of these structures and the practice and operational realization of these structures. The conventional cryptographic applications like OpenSSL are black boxes, which receive an input and give back a hexadecimal string without exposing the verification logic. This obstructs debugging of code by the programmer and clouds the security model by the stakeholders.

The implementation of cryptographic integrity checks faces three major challenges. The first is that abstraction of logic, where developers often use libraries without a proper understanding of the underlying propagation of nodes to roots and make inappropriate implementations. The second deals with invisible tampering. It is common in traditional databases to have data corruption that goes unnoticed. There is a need for the tools that visually track an invalid root hash to the respective compromised ledger entry. Third is about verification complexity.

Grenze ID: 01.GIJET.12.2.77_26

It is counter-intuitive that the concept of a Merkle Proof, i.e., where only sibling hashes are used to reconstruct the root hash does not require a visual representation making it harder to grasp and implement correctly.
For this purpose, a real-time Merkle Tree Visualizer design and development solution is proposed in this study. It is a web-based application that allows generating different entries within a ledger, creating the hash tree automatically, and marking the verification paths required in the Merkle proof. The rest of this paper is structured as follows. Section 2 is a review of related work. Section 3 outlines the system architecture and methodology that is proposed. The implementation details are presented in Section 4. Experimental results are discussed in Section 5 and future research directions are presented under Section 6.

II. Literature Review

In the context of distributed systems, especially blockchains, Merkle trees are an important cryptographic tool that helps verify data integrity and provides efficient verification capabilities. For blockchain systems, a recent paper by Kuznetsov et al. [1] investigates the collision probability and security issues of Merkle trees, which will ensure integrity of the blockchain's ledger if a proper hash function is used. Gracy et al. [2] suggested the use of pruned Merkle trees, demonstrating that the computation required for the trees could be significantly reduced and there is only a marginal amount of overhead for lightweight blockchain clients. Educational issues have also been studied, besides efficiency of structure. The effectiveness of teaching through interactive visualization is also supported by research in educating in cryptography, which shows that the use of interactive visualization in the teaching process is effective, which improves the students' understanding of advanced cryptography, such as hashing and Merkle trees [3]. Post-quantum blockchain security has become a focus with the advent of quantum computing. The performance evaluation of post-quantum secure blockchain systems [4] challenges the security of the traditional cryptographic schemes-based blockchain systems. A comparison of the quantum security of SHA-256 with classical hash functions further shows that the classical hash functions are not as quantum secure as SHA-256 [5]. To overcome these concerns, researchers have come up with quantum-resistant designs for Merkle trees along with post quantum cryptography and zero knowledge proofs [6]. Other improvements have been shown using parallel traversal of merkle trees [7]. The use of cryptographic primitives in blockchain is addressed in more general research [8] and the post-quantum signature schemes and their applications to blockchain in recent research [9]. Implementational challenges of PQC in the next generation systems are also explored [10]. In [11] the authors provide a method for efficiently proving a large set of data, using a single root hash, in the blockchain systems. It identifies that many records/documents can be committed as one and does not incur any storage overhead or transaction overhead for this. In [12] advanced storage techniques are explored to understand how data can be efficiently stored and managed in decentralized systems such as IPFS. It details how files are partitioned into smaller blocks and the Merkle DAG is used for integrity and verifiability, using root hashes. There are a large number of examples in the literature of cryptographic integrity based on Merkle structures, but little work has been done on the aspect of visual and interactive techniques for understanding the integrity and understanding of debugging.

III. System Design and Methodology

A. System Architecture

The Merkle Tree Visualizer is a three-panel single-page application (SPA) entirely served from the browser itself, without any need of a central back-end server. There are two significant benefits of this design decision. The first is that all cryptographic operations are performed within the user's trusted environment, which means it is never sent a round trip over a network to a server to be accessed by an attacker; the second is that all state updates are performed on the client side, using React's virtual DOM reconciliation, which means that no computation is done on the server, so no round trip to the server is needed to render. The whole system architecture is divided into three functional layers summarized on the integrated dashboard in Fig. 1.
The first layer is the Data Input Layer. All the Ledger Entry text fields are dynamically connected to the core state of the application via the use State hook of React in the left panel. Every keystroke triggers a state mutation, thus immediately triggering the tree re-computation pipeline. You can add another leaf to the ledger array with an “Add Block” button, and slots can be edited anytime. This design makes the append-only semantics of a real distributed ledger and allows for full mutability for educational exploration of tampering scenarios. Internally, each text value is converted to a UTF-8-byte stream prior to sending to the Web Crypto API ensuring that any change in characters is correctly represented as an independent 256-bit digest.

The second layer is the Tree Visualization Layer. The Centre panel grabs an SVG canvas, which is redrawn each time the state changes. The nodes are then arranged recursively, in depth-first manner, and the depth and offset from the root given as the pixel coordinates. The edges of the connectors are drawn using cubic Bézier curves that are a good approximation of the diagonals which appear in traditional binary tree diagrams. Colour semantics: Nodes with valid integrity are displayed in green (#00CC88), and nodes whose computed hash is not matching with the integrity of the snapshot will be displayed in red (#FF4D4D), nodes located in the active.

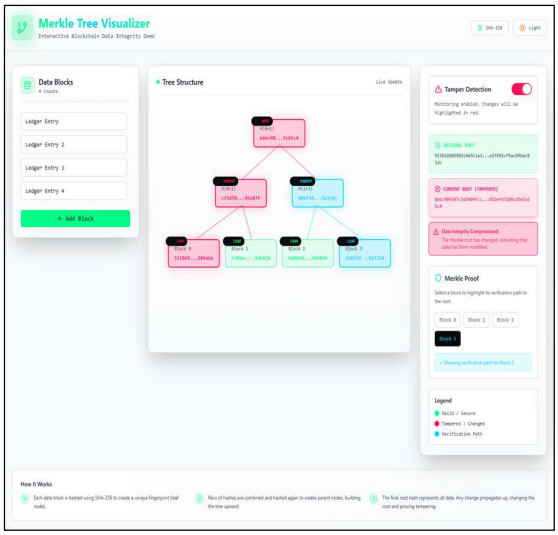

Figure 1. Full Integrated Dashboard of the Merkle Tree Visualizer Showing Tampered State

Merkle proof path are highlighted in cyan (#00B4D8). A ‘Live Update’ badge means that the input is updated as the state of the canvas changes, and a root hash truncation label (first eight and last eight hex characters respectively) is displayed in each node tile to help to visually compare the input's state.

The third layer is the Integrity Control and Proof Panel. The right side of the panel is broken up into three functional sub-modules. The Tamper Detection toggle allows to compare the current root hash with a snapshot, which is "frozen" when the toggle is set. If it is noticed, it will be indicated by a red alert banner “Data Integrity Compromised” and the root node tile will change its colour to indicate tamper. The Original Root and Current Root display blocks are showing the full hex digest of the mutation (SHA-256) so that humans can verify that the mutation is occurring. The Merkle Proof sub-module shows individual block selectors; clicking on any leaf block highlights the smallest sibling hash path to re-derive the root from that leaf only, in a very tangible way showing that the proof has an O(log n) complexity.

B. Methodology

Using a web interface, multiple data blocks could be entered into the system. These blocks are shown as "leaf nodes" on a tree structure. The application dynamically creates a hierarchy of the tree and keeps updating it in real-time. The visualization is updated on-the-fly if data is changed, so that users can check how the changes affect the entire tree, up to the root. The process of development follows a constructive research approach and is centered on the production of an artefact (software) solution to the problem formulated. The design of the app follows the Client-Side Rendering (CSR) approach which enables feedback without any latency. The Front-End Logic is developed on React.js to provide a better control of the tree nodes state. The system uses the Web Crypto API (SHA-256) to create cryptographically safe hash messages directly in the browser and capture the visualization of Web Crypto representing genuine security conduct.

In the hashing workflow the system is comprised of a dynamic set of inputs labeled “ledger entries”, N. The first step (Leaf Hashing) is to hash each ledger entry individually with the SHA-256 algorithm to get leaf hashes $H(L_i)="SHA-256"(〖"Ledger Entry" 〗_i). To create the parent nodes in the second step (Pairing), adjacent leaf hashes are concatenated and then re-hashed, so 〖$H〗_"parent" ="SHA-256"(H(L_i)‖H(L_(i+1))). It applies the same hashing/pairing to higher levels of the tree (recursively) until it reaches the top, which ultimately yields a single root hash that is a 64-character hexadecimal string showing the overall integrity of the data contained in the ledger

IV. IMPLEMENTATION

The implementation process was carried out in three phases such that it was modular and scaled. Phase-1

A. Phase 1:User Interface and Data Input

An interactive user interface was built to receive transaction data. The inputs were typed as such instead of generic text fields and were of type Ledger Entries. The user adds blocks through an “Add Block” Button as shown in Fig. 2. When Ledger Entry 1 is typed with the value of the string, which is Tx1 or Alice pays Bob the state management captures the value and the hashing engine is automatically activated.

B. Phase 2:Merkle Tree Generation

The system generates a Merkle Tree using a JavaScript utility function called generate-Merkle-Tree. The engine would automatically check whether the number of ledger entries is odd, and if this were the case, it would repeat the final hash to form a balanced binary tree as specified by the conventional blockchain protocols and shown in Fig. 3. The function returned a nested object data structure of hash values of all levels of the nodes (leaves, parents, root).

C. Phase 3:Visualization & Attack Simulation

The UI creates the tree with Scalable Vector Graphics (SVG) connector. Colour coding is used where in there is a green colour(#00ff00) on the valid nodes and blue colour is used on the path to be verified. To emulate the Man-in-the-middle-attack, a tamper-toggle component was created that enabled the user to edit an already committed ledger record and watch how the system reacted to this.
The tool successfully visualizes hierarchical data integrity verification in real time as shown in Fig. 4.

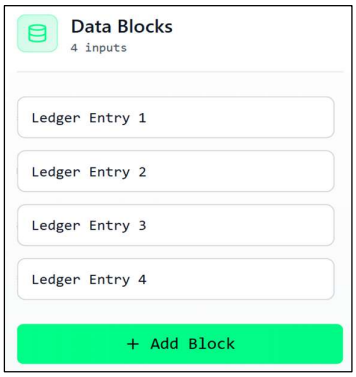

Figure 2. The Input Layer

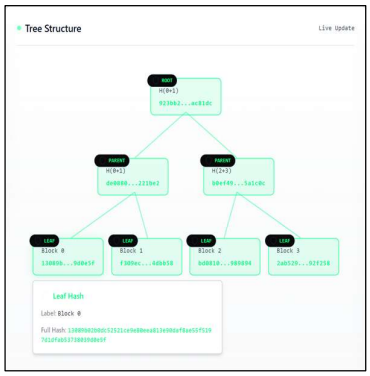

Figure 3. Merkle Tree construction and hash propagation

Figure 4. Integrity Verification

V. Results and Discussion

The developed system offers a very responsive system to analyse data integrity. It has the following features

A. The Avalanche Effect

For a leaf change such as when the input of the ledger entry was changed by one character (e.g., a 1 was changed to 2), the system showed the avalanche effect.

The hash of Ledger Entry 1 was changed immediately. As a result, the parent node hash was entirely modified, and the final Root Hash became an entirely new hexadecimal. Detection of data tampering showing change in Merkle root and affected nodes is shown in Fig. 5. This confirms that even microscopic tampering in the ledger renders the entire dataset invalid.

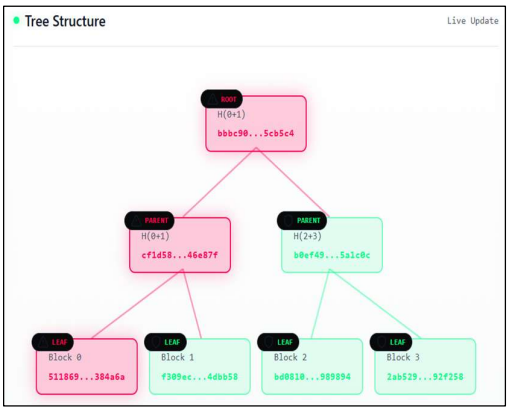

Figure 5. Data Tampering Visualization

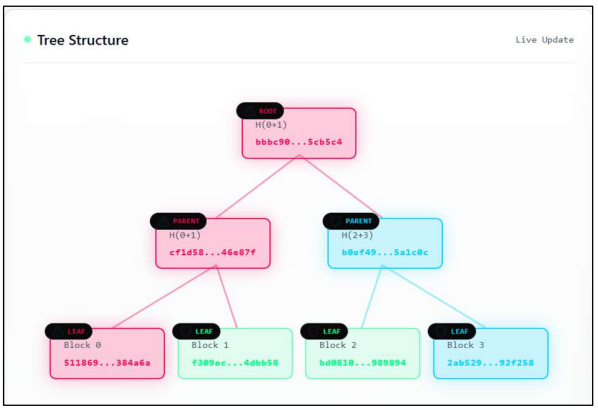

Figure 6. Visualization of Merkle Proof Path

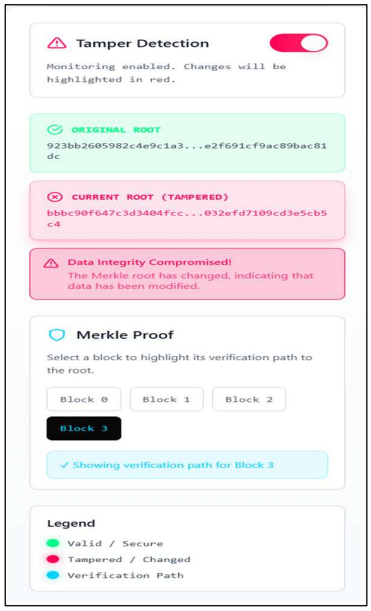

Figure 7. Real Time Integrity verification

B. Merkle Proof Path Visualization.

Making use of the Ledger Entry 1 as a verifiable entry, the system identified and highlighted the sibling nodes as shown in Fig. 6. The real time integrity dashboard as shown in Fig. 7, enables users to understand how individual data blocks can be validated efficiently without processing the entire dataset. This finding confirms that to authenticate one entry among four, only two more hashes are required to show clearly that it is efficient at O (log n).

C. Interpretability and Abstraction:

The hashing process is abstracted in standard libraries, and this, although efficient in production, might be counterproductive to educational purposes. This tool fills this gap by exposing the intermediate nodes (parent hashes) and thereby is useful in debugging the developers' own implementations by comparing them with the visualizer output.

D. Security Consequence of Visual Feedback:

The tamper-detection feature, which is a change in node colour, to red, gives a user-friendly warning. Such visualization dashboards can be scaled to Network Operations Centers (NOCs) in a real-world Cyberspace scenario to enable them to look at the state of data integrity at distributed nodes briefly.

E. Limitations

The existing system plots the SHA 256 algorithm. Although this is the industry standard this is soon to change to quantum computing, and it is advised that subsequent versions should use larger hash signatures (e.g., SPHINCS+) as an example of the storage overhead of post-quantum security.

VI. Conclusion and Future Work

This study was able to develop and establish an internet-based structure of visual profiling cryptographic data integrity. The system allows the user to obtain unequivocal data tampering evidence by transforming the abstract mathematical standard of Merkle Trees into an interactive graph of what can be called Ledger Entries. The modular one allows real-time interaction which makes it a useful tool in both the educational and preliminary system prototyping. The framework shows that visual profiling is not only an aesthetic addition but also a practical need to understand the complicated proliferation of trust in decentralized systems.

The improvement in the future will be aimed at two aspects. One on Post-Quantum Algorithms, where combining libraries of SPHINCS+ and Dilithium will allow comparing the time required to generate a tree and signature size side by side. Second on Large -Scale Simulation, where integration of a zoomable canvas interface will facilitate the display of trees with thousands of Ledger Entries, thus modeling block depths in the real world blockchain.

References

[1] Kuznetsov, O., Rusnak, A., Yezhov, A., Kuznetsova, K., Kanonik, D., & Domin, O.: Merkle Trees in Blockchain: A Study of Collision Probability and Security Implications. J. Cryptographic Engineering, Vol. 26 2024. doi: 10.1016/j.iot.2024.101193

[2] Gracy, M., Jeyavadhanam, R., & Raj, A. A.: Enhancing Transaction Verification Through Pruned Merkle Trees. J. Theoretical and Applied Information Technology, 101(20), pp. 6421 6433, 2023.

[3] Imad Al Saeed: Leveraging Visualization for Cryptography Education. Journal of Computing Sciences in Colleges, Vol. 40, Issue 4, pp. 63 – 71, 2024

[4] Y. Kim and J. Choi: An interactive learning tool for teaching public-key cryptography.". In: IEEE Access. Vol.7, pp. 12243-12252(2019).

[5] Prodipto Das, Sumit Biswas, Sandip Kanoo and Veeradittya Podder: Design and Comparative Analysis of Quantum Hashing Algorithms using Qiskit, 2025. doi:10.36227/techrxiv.24037440.v1

[6] Azhar, H. B., Butt, K. K., Awan, N. U., & Irshad, O.: Quantum-Resistant Merkle Trees: Enhancing Data Integrity with Post-Quantum Cryptography and Zero-Knowledge Proof, Vol. 8, Issue 2, 2025. doi: 10.56979/802/2025

[7] Wang, Z.: An Example of Parallel Merkle Tree Traversal. ACM Trans. on Embedded Computing Systems, Vol. 23, Issue 4,pp.1-26, 2024. doi: 10.1145/3659209

[8] Senarathna, J. I.: Blockchain and How It Relies on Cryptographic Methods. Preprints, 2025.

[9] Iavich, M.: Post-Quantum Digital Signature: Verkle-Based HORST. Cryptography 5(2), pp. 28 ,2025.

[10] Khan, M. A.: Implementation and Performance of Post-Quantum Cryptography for Next Generation CE Devices. Springer Int. J. Security and Networks, Vol. 5, 2025.

[11] Toyos-Marfurt, G., et al.: Notara: Efficient Blockchain Asset Notarization Service. In: Proceedings of the International Conference on Blockchain and Distributed Systems, pp. 1–6, 2024

[12] Qiyuan Liang, Yuening Yang, et al.: Towards Efficient Data Management For IPFS-dApps. arXiv:2404.16210 , 2024.