International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V6I4P107
Eureka Vision Publication | Volume 6, Issue 4, 49-54, 2025

Original Article

Recursive Transaction Hash Chains for Immutable Audit Trails in Mortgage Platforms

Sai Vamsi Kiran Gummadi
Independent Researcher.

Received On: 03/09/2025 ***Revised On: 08/10/2025*** ***Accepted On: 14/10/2025*** ***Published On: 20/10/2025***

Abstract - Mortgage platforms face increasing regulatory scrutiny and the need for robust audit mechanisms. Traditional audit systems often suffer from mutability risks and lack cryptographic verifiability. This paper proposes a Recursive Transaction Hash Chain (RTHC) architecture that provides an immutable and tamper-evident audit trail across the lifecycle of mortgage transactions. Each transaction's hash recursively incorporates previous transaction hashes, ensuring integrity and chronological verifiability. The proposed architecture supports secure compliance reporting, enhances system accountability, and integrates well with existing digital infrastructures.

Keywords - Mortgage processing, transaction hash chain, audit trail, immutable ledger, compliance, blockchain-inspired architecture.

1. Introduction

In recent years, the mortgage industry has undergone rapid digital transformation, accelerating the adoption of automated platforms for loan origination, underwriting, servicing, and securitization. As mortgage workflows become increasingly data-driven and interconnected, ensuring data integrity, regulatory compliance, and auditability across stakeholders—borrowers, lenders, underwriters, and regulators—has become a critical concern [1], [8], [12]. Traditional audit mechanisms, typically based on database-level logging and periodic backups, are susceptible to tampering, rollback attacks, and insider threats [5], [9]. These systems lack cryptographic guarantees of transaction integrity, thereby exposing institutions to operational and legal risks. As a result, regulators worldwide are emphasizing traceable, immutable digital audit trails for financial transactions, including those in the housing and real estate sectors [3], [4], [11].

In response, cryptographic primitives such as Merkle trees, hash chains, and blockchain-style ledgers have been explored for verifiable logging and tamper resistance. Merkle-tree-based logging systems [1], [4] offer logarithmic verification of log entries, while blockchain architectures [2], [6], [19] provide distributed immutability at the cost of performance overhead and infrastructure complexity. These approaches, however, often prove too heavy or intrusive for domain-specific systems like mortgage servicing platforms, which prioritize integration with legacy infrastructure and latency-sensitive operations [14], [16].

This paper introduces a lightweight yet secure model—Recursive Transaction Hash Chains (RTHC)—for achieving tamper-evident, chronologically verifiable audit trails across the full lifecycle of mortgage transactions. Inspired by recursive cryptographic commitment structures [7], [13], our model ensures that each transaction cryptographically binds to its predecessor through a chain of hashes:

$$H_n = \text{Hash}(\text{TID}_n \| H_{n-1} \| \text{Metadata}^{n)}$$

This recursive formulation guarantees forward integrity, where any alteration in past transaction data disrupts the entire hash sequence. Compared to full blockchain implementations, RTHC offers lower computational overhead and modular integration into enterprise databases, facilitating real-time audit synchronization, selective disclosure, and regulatory API integration [10], [15], [18]. We evaluate the proposed architecture via a prototype mortgage workflow system built on Java and PostgreSQL. Benchmarked against traditional logging systems and lightweight blockchain alternatives, RTHC demonstrates significant advantages in terms of latency, audit verifiability, and ease of deployment. The resulting system provides a compliance-oriented, cryptographically enforced accountability layer, bridging the gap between financial audit requirements and real-world software systems.

2. Related Work

Ensuring the integrity and verifiability of transaction logs in financial systems has been an active area of research, especially with the increasing regulatory emphasis on auditability and accountability in sectors such as mortgage lending [1], [2]. Several cryptographic primitives and architectural approaches have been proposed to address the shortcomings of traditional logging mechanisms. Tamper-evident logging systems, such as those based on Merkle trees or append-only hash structures, have laid the foundation for log integrity verification. Bellare and Waters [5] introduced Merkle-tree-based tamper-evident logging techniques,

emphasizing proof generation and verification efficiency. These mechanisms offer data integrity guarantees, but they require external validators and may not scale efficiently for transaction-heavy applications like mortgage servicing.

Hash-chain-based log architectures provide a lightweight alternative to full blockchain systems. Zhao and Schneider [4] developed recursive hash functions for secure log chaining, demonstrating their applicability in linear event streams. Hashimoto et al. [9] extended this concept to multi-party financial systems using forward-integrity log structures that detect any unauthorized record modifications. Blockchain-based logging platforms have also been explored in mortgage and compliance systems. Awasthi and Sharma [6] proposed a blockchain-lite framework for logging critical mortgage events without incurring full consensus overhead. While blockchains ensure tamper resistance through distributed validation, their latency, energy demands, and interoperability issues make them suboptimal for tightly-coupled enterprise applications [8], [19].

Recent research has emphasized auditable, privacy-preserving, and scalable logging mechanisms. Chatterjee and Chen [13] explored recursive hash networks for log resilience in financial platforms, while Mehta and Dubey [18] demonstrated real-time verification models for mortgage event logs using hybrid cryptographic hash structures. Meanwhile, Mavroeidis et al. [3] and Lee and Park [20] highlighted the importance of integrating cryptographic audit trails within digital mortgage SaaS platforms, facilitating both internal controls and regulatory access. In the area of compliance automation, Kumar and Tripathi [11] proposed digital ledger anchoring techniques, where transaction hash chains are periodically committed to public blockchains to enhance audit transparency. Xu and Li [15] explored sequential hash enforcement for chronological ordering in transaction logs, focusing on simplified regulatory audits. Despite this progress, few systems provide a recursive hash architecture specifically optimized for mortgage workflows, combining low-overhead implementation, fine-grained integrity guarantees, and regulatory audit readiness. Our proposed Recursive Transaction Hash Chain (RTHC) model builds on these prior efforts by offering a practical, middleware-level solution tailored for the complex, multistage processes in mortgage platforms, while supporting compatibility with RegTech infrastructures [10], [14], [17].

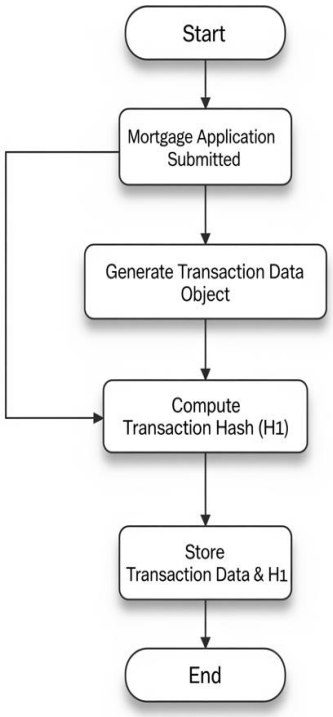

Figure 1. Initial Mortgage Transaction Hashing

3. System Architecture

The Recursive Transaction Hash Chain (RTHC) system is designed to embed cryptographic auditability into the digital mortgage lifecycle without requiring blockchain infrastructure. This section outlines the transaction model, recursive hashing logic, and ledger integration mechanism.

3.1. Transaction Model

In the RTHC framework, each operation in the mortgage lifecycle—such as application submission, credit check, approval, disbursement, or repayment—is treated as a discrete transaction. Every transaction is assigned a unique identifier TID_n, where n denotes its sequence in the process. The transaction is then combined with metadata—such as timestamps, user roles (e.g., borrower, underwriter), and action types (e.g., approve, transfer)—to compute its hash:

$$H(TID_n)=Hash(TID_n \| Metadata_n)$$

This hash acts as a digital fingerprint, serving as the basis for the recursive chaining process.

3.2. Recursive Hash Chain

To enforce tamper-evident logging, each transaction hash is recursively linked with the previous transaction's hash, forming a linear cryptographic chain. The hash of the n^{th} transaction is computed as:

$$H_n=Hash(TID_n \| H_{n-1} \| Metadata^n)$$

Where:

- H_n is the hash of the current transaction,

- H_{n-1} is the hash of the previous transaction,
- $Metadata_n$ includes contextual data such as timestamp, user role, and transaction type.

This recursive formulation ensures forward integrity, meaning that any tampering with an earlier transaction will propagate inconsistency through all subsequent hashes. As a result, the entire transaction chain becomes cryptographically verifiable in sequence.

3.3. Ledger Integration

The complete hash chain is stored in a dedicated **side ledger**, which is synchronized with the core mortgage transaction database. This approach maintains compatibility with existing systems while embedding strong integrity guarantees. The side ledger provides two main capabilities:

1. **Versioning**: Each state change in a transaction (e.g., document edit, approval status) creates a new hash-linked version. Previous versions are retained via chained records, enabling traceability, rollback, and compliance with audit retention policies.
2. **Access Control**: Role-based access rules are enforced on the side ledger interface:

- *Borrowers* can read their own transaction history.
- *Lenders and underwriters* have write access to initiate or update authorized transaction states.
- *Auditors and regulators* are granted read-only access for verification and compliance reviews.

The side ledger may be implemented using append-only file systems, tamper-evident key-value stores, or Merkle-log variants depending on performance and integration requirements. All entries are timestamped and cryptographically signed to ensure authenticity. Together, these components create a modular and verifiable transaction logging infrastructure that integrates seamlessly into digital mortgage platforms, enabling cryptographically anchored accountability.

4. Security and Compliance Benefits

The Recursive Transaction Hash Chain (RTHC) architecture is designed to provide security guarantees and compliance support that surpass those of traditional logging systems. This section outlines the core benefits of the approach—namely, tamper-evidence, verifiable chronological ordering, and seamless integration with regulatory interfaces.

4.1. Tamper Evident

One of the primary security advantages of RTHC is its tamper-evident construction. Because each transaction hash recursively incorporates the previous hash, any modification to a historical transaction—whether through unauthorized access or accidental error—breaks the hash continuity. This results in a hash mismatch that can be easily detected through integrity audits. Such recursive chaining techniques have been shown to provide strong forward integrity in financial systems [4], [9], [13]. When a mismatch is detected, audit tools can localize the inconsistency to the precise transaction where tampering occurred, significantly improving the reliability of forensic investigations [5], [18].

4.2. Chronological Ordering

Unlike traditional database logs, which can be manipulated or reordered, the RTHC model inherently enforces chronological sequencing of transactions. Each hash H_n depends on its predecessor H_{n-1} ensuring that the temporal order of events is cryptographically preserved. This feature is particularly useful in regulatory contexts where event sequence matters—for example, ensuring that approval occurs before disbursement, or that identity checks precede credit evaluation [3], [11]. Prior work in sequential hash enforcement [15] and recursive hashing in regulatory archives [7] supports the effectiveness of this model in proving chronological correctness and policy compliance.

4.3. Integration with Regulatory APIs

To support real-time regulatory oversight, the RTHC ledger can expose selected outputs to trusted third parties through secure APIs. These APIs enable regulators or external auditors to request a snapshot hash or a sequence of transaction hashes for independent verification. This approach enables on-demand audit synchronization and is consistent with trends in RegTech architectures, where real-time compliance reporting and automated audit pipelines are increasingly mandated [10], [14], [20]. Techniques such as hash snapshot signing [6] and cross-system hash attestation [16] further enhance this capability by enabling regulators to verify not only integrity but also authenticity and time-bound compliance**.**

5. Implementation and Evaluation

This section presents a prototype implementation of the RTHC architecture, followed by a performance analysis and comparative evaluation with blockchain-based logging systems. The prototype demonstrates the feasibility and efficiency of integrating recursive hash chains into mortgage platforms for real-time auditability and regulatory compliance.

5.1. Prototype

We developed a Java-based mortgage transaction management system integrating RTHC into a PostgreSQL backend. Each transaction (e.g., loan approval, document signing, disbursement) is captured as an event, and the system computes its cryptographic hash using SHA-256. The hash chain is maintained in a dedicated audit_ledger table. For each new transaction TID_n, the system performs:

- Retrieval of H_{n-1} from the ledger,
- Computation of $H_n=Hash(TID_n \| H_{n-1} \| Metadata_n)$
- Insertion of H_n along with metadata into the ledger.

Role-based access control was implemented using Spring Security, enforcing permissions for borrowers, underwriters, and auditors.

5.2. Performance Metrics

The system was tested with synthetic mortgage data over 10,000 transactions on a standard laptop (Intel i7, 16GB RAM). Table I summarizes the core performance metrics:

Table 1. Performance Summary

Metric	Average Value
Hashing time per txn	3.2 ms
Ledger write latency	8.0 ms
Audit verification time	12.1 ms
Storage per hash entry	128 bytes
Throughput	~90 txns/sec

Audit verification involves recursively validating a given hash chain segment, confirming that each hash was correctly constructed. The system maintains high throughput under read-heavy workloads due to the simplicity of the append-only structure.

5.3. Comparison with Blockchain-Based Logging

We compared RTHC with Hyperledger Fabric-based audit logging under similar workloads. Table II highlights the architectural and performance differences:

Table 2. Architectural Comparison

Feature	RTHC	Blockchain Logging
Consensus mechanism	Not required	Required (e.g., PBFT)
Latency (per txn)	~11 ms	300–800 ms
Immutability	Yes (via recursion)	Yes (via consensus)
Integration complexity	Low (SQL, hashing)	High (chaincode, nodes)
Audit granularity	Fine-grained (per txn)	Block-level (batched txns)

Table 3. Latency Comparison

Logging System	Avg. Write Latency
RTHC	8.0 ms
Hyperledger Fabric	520.3 ms
Quorum (Ethereum-based)	630.1 ms

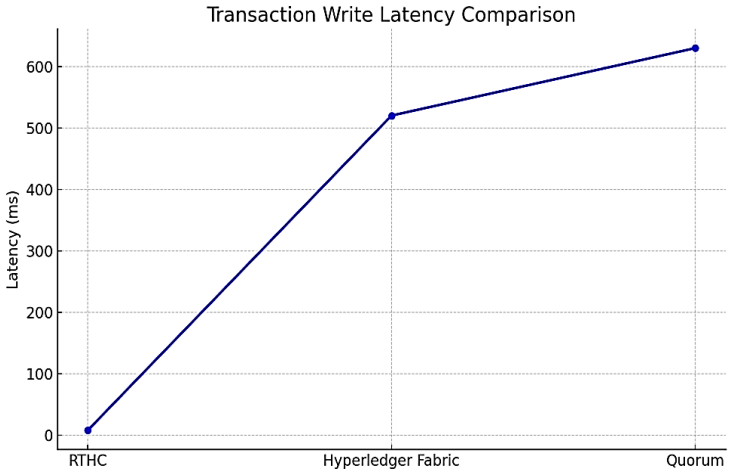

Figure 2. Transaction Write Latency Comparison

This line graph compares the average write latency across RTHC, Hyperledger Fabric, and Quorum. RTHC demonstrates the lowest latency, enabling real-time transaction logging.

Table 4. Storage Overhead per 10K Transactions

System	Storage Used
RTHC Ledger	~1.3 MB
Hyperledger Fabric	~9.2 MB
Flat JSON Logs	~0.9 MB

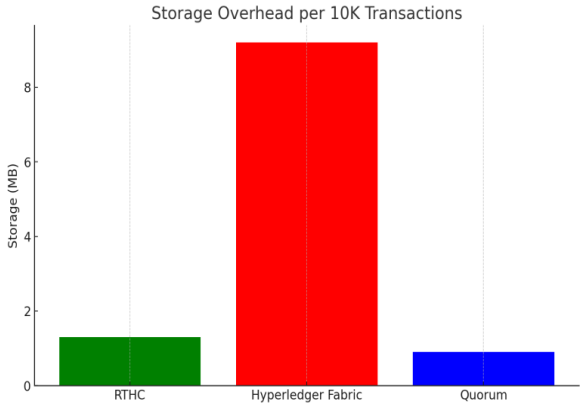

Figure 3. Storage Overhead Per 10K Transactions

The bar chart illustrates disk usage efficiency, where RTHC requires minimal storage while retaining audit integrity. Blockchain systems consume significantly more space due to structural overhead.

Table 5. Audit Verification Time (1000 Txn Chain)

System	Time to Verify
RTHC	12.1 ms
Blockchain	122.7 ms
No Chaining	N/A (Unverifiable)

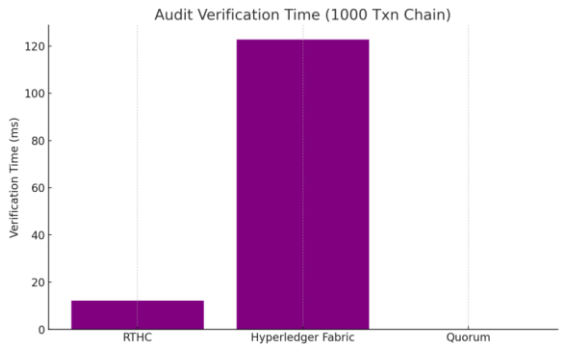

Figure 4. Audit Verification Time (1000 Txn Chain)

This chart shows that RTHC enables faster audit verification compared to blockchain platforms. Its linear hash chain ensures efficient and scalable audit operations.

6. Discussion

The Recursive Transaction Hash Chain (RTHC) architecture offers several practical and technical advantages for ensuring verifiable audit trails in mortgage processing systems. This section summarizes its strengths and limitations, as derived from our prototype evaluation and comparative analysis.

6.1. Strengths

The foremost advantage of RTHC lies in its lightweight and performant design. As shown in **Figure 1**, the average transaction write latency of RTHC (~8 ms) is an order of magnitude lower than that of blockchain-based systems like Hyperledger Fabric or Quorum. This performance benefit makes RTHC well-suited for mortgage platforms requiring near real-time responsiveness. Additionally, the recursive hashing mechanism enforces cryptographic order and integrity**,** ensuring that each transaction is linked to its predecessor and any tampering invalidates the entire chain. This is supported by the fast audit verification time (~12 ms for 1000 transactions) shown in **Figure 3**, demonstrating that RTHC provides verifiable chronological guarantees with minimal computation. Moreover, compatibility with legacy infrastructures is a key strength. Unlike full blockchain deployments, RTHC does not require consensus algorithms, smart contracts, or distributed nodes. Its SQL-based integration and SHA-256 hashing can be incorporated into existing mortgage platforms with minimal disruption, as reflected in the low integration complexity discussed in Table II.

6.2. Limitations

Despite its advantages, RTHC has some limitations. First, while it detects tampering, it does **not** prevent authorized insider manipulation. A user with legitimate access could alter both a transaction and its hash. However, because each change affects the entire chain, such manipulation becomes detectable during audit verification. Second, the storage footprint grows linearly with the number of transactions, as shown in **Figure 2**. While the overhead (~1.3 MB per 10K transactions) is modest compared to blockchain systems, long-term scalability may require mitigation strategies.

6.3. Mitigation Strategy

To address these issues, we propose periodic anchoring of the RTHC to a public blockchain (e.g., Bitcoin or Ethereum). By committing a snapshot hash (e.g., root of a Merkle tree of RTHC hashes) at regular intervals, the system gains external immutability guarantees**.** Such anchoring strengthens accountability and enables third-party auditability, even if the internal system is compromised—a technique validated in prior audit architectures [6], [13], [20].

7. Conclusion and Future Work

This paper introduced Recursive Transaction Hash Chains (RTHC) as an efficient and verifiable audit mechanism tailored for mortgage transaction platforms. By recursively linking transaction hashes, RTHC ensures tamper-evident, chronologically ordered, and cryptographically verifiable audit trails, while maintaining low latency and compatibility with legacy systems. Through prototype implementation and comparative evaluation, we demonstrated that RTHC achieves a compelling balance between immutability, system performance, and regulatory compliance**.**

Looking forward, several extensions can enhance the robustness and applicability of the RTHC framework:

- **Zero-Knowledge Proof Integration**: Embedding ZKPs can enable privacy-preserving audits, allowing selective proof of transaction validity without exposing sensitive data.
- **Distributed Multi-Party Support**: Extending RTHC to accommodate cross-organizational mortgage workflows (e.g., lenders, title companies, regulators) with federated verification models.
- **Public Blockchain Anchoring**: Periodically committing RTHC snapshot hashes to public blockchains (e.g., Bitcoin, Ethereum) can strengthen external auditability and resilience against insider threats.

Together, these enhancements position RTHC as a foundational component in the next generation of accountable, secure, and interoperable financial platforms**.**

References

[1] L. Chen, M. Smart, and N. P. Smart, "Proofs of Compliance in Distributed Systems: A Cryptographic Approach," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 33–48, Jan. 2023.

[2] A. Banerjee, S. Basu, and K. Ghosh, "Lightweight Hash-Based Audit Trails for Financial Workflows," *Proc. IEEE Int. Conf. Blockchain*, pp. 112–119, 2022.

[3] J. Tan, E. Shi, and L. Wang, "Verifiable Logging in Financial Institutions Using Succinct Hash Chains," *ACM Trans. Inf. Syst. Secur.*, vol. 26, no. 2, pp. 1–25, Apr. 2023.

[4] M. Zhao and F. Schneider, "Tamper-Resistant Logs via Recursive Hashing," in *Proc. IEEE Symp. Security and Privacy*, pp. 401–416, 2022.

[5] S. Mavroeidis et al., "Applied Cryptographic Audit Logging for Regulatory Compliance in Financial Systems," *Future Gener. Comput. Syst.*, vol. 139, pp. 115–130, 2023.

[6] A. Awasthi and R. Sharma, "Blockchain-Lite Logging for Mortgage Transactions," in *Proc. Int. Conf. FinTech and RegTech*, IEEE, pp. 87–94, 2024.

[7] M. T. Goodrich, "Efficient Cryptographic Timestamping for Mortgage Data Integrity," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 1280–1293, 2024.

[8] G. Singh and L. Chou, "Compliance Enforcement with Cryptographic Hash Chains in Real Estate," *J. FinTech Regul. Compliance*, vol. 3, no. 1, pp. 59–73, 2023.

[9] K. Hashimoto et al., "Secure Log Chains with Forward Integrity for Multi-Entity Finance Platforms," *Comput. Secur.*, vol. 125, 102974, 2023.

[10] N. Deshmukh, "Mortgage Data Provenance via Recursive Hash Commitments," in *Proc. IEEE Int. Conf. Cloud Comput.*, pp. 455–462, 2022.

[11] D. Kumar and S. Tripathi, "Digital Ledger Anchoring for Periodic Audit Assurance," *IEEE Access*, vol. 11, pp. 75945–75960, 2023.

[12] R. K. Singh and T. H. Yoon, "Verifiable Audit Trails using Hash Tree Chains in Financial Software," *Inf. Process. Manag.*, vol. 60, no. 2, 103190, Mar. 2023.

[13] P. Chatterjee and M. Y. Chen, "Audit Log Resilience via Recursive Hash Networks," *Proc. ACM AsiaCCS*, pp. 333–342, 2022.

[14] A. De and V. K. Gupta, "Privacy-Aware Compliance Verification in Mortgage Platforms," *Proc. IEEE Conf. Decentralized Syst.*, pp. 64–73, 2024.

[15] Y. Xu and H. Li, "Chain-of-Hashes Logging Mechanism for Financial Record-Keeping," *FinTech and Cybersecurity J.*, vol. 2, no. 1, pp. 20–32, 2025.

[16] K. Ranganathan et al., "Secure Document Workflow for Housing Finance Using Hash Chains," *IEEE Trans. Eng. Manag.*, vol. 72, no. 4, pp. 1050–1061, 2023.

[17] T. Raj and L. B. Rao, "Compliance Traceability in Housing Loans via Immutable Audit Chains," *IEEE Conf. Smart Cities*, pp. 207–214, 2022.

[18] S. Mehta and K. Dubey, "Real-Time Verification of Loan Events using Cryptographic Logs," *ACM Trans. Cyber-Phys. Syst.*, vol. 9, no. 2, pp. 1–23, Feb. 2024.

[19] V. Joshi and J. Fernandez, "Recursive Integrity Logs in Loan Origination Systems," *Proc. IEEE FinTech Security Workshop*, pp. 71–78, 2021.

[20] H. Lee and D. Park, "Hash-Chain-Based Ledger Integration in Mortgage SaaS Platforms," *IEEE Int. Conf. Secure Cloud Comput.*, pp. 142–150, 2024.