Use of a Hash-Based Integrity Verification Framework for Secure Data Transfer Between Database Tables

HASSAN BEDIAR HASHIM
Middle Technical University, Baghdad,
IRAQ

Abstract: Data integrity guarantee among transferred data between tables is also an essential problem in modern database-reliant systems, especially in security-sensitive scenarios such as finance, healthcare, and large-scale administrative applications. While cryptographic hash functions and message authentication operations are well-known, their application in databases as part of a protocol that ensures systematic data transfer with concurrency control and automatic recovery has not been addressed satisfactorily. We proposed hash-based integrity verification parallel framework that integrates the SHA-256 hashing and the HMAC-based authentication to guarantee secure and reliable data movement between relational database tables. The protocol operates in four coordinated stages: pre-transfer hash generation, authenticated data transfer using HMAC, post-transfer integrity verification, and a process that can automate the recovery of such corrupted content. The model was realized by Python and MySQL, and verified through a series of large-scale experiments that include controlled link failure injections and adversarial modifications of the data. Results of experiments show that the error rate of data transmission in the proposed approach drops to 0.1% from 4.5%, and with the mandatory referential integrity guaranteed, it can reach up to a rate of close to 100%. Although the history-based scheme may suffer a 25.5% higher transfer time because of cryptographic calculations, such overhead is tolerable to cases constantly demanding high reliability and data security. Results further manifest that the advocated schema offers a flexible integrity-preserving data transfer solution applicable to relational databases.

Keywords: Data Integrity; SHA-256; HMAC; Secure Data Transfer; Database Security; Integrity Verification

Received: June 16, 2025. Revised: September 29, 2025. Accepted: October 27, 2025. Published: January 26, 2026.

1. Introduction

The explosive proliferation of digital data has changed the world in which originations must compete, where data represents a fundamental core asset for competitive advantage, systems management and operational decisions making. The rapid growth of big data analytics, cloud computing infrastructures, Internet of Things (IoT) applications and real-time information systems gives relational database management systems (RDBMSs), a strategic place in storing, managing and processing structured data ([1], [2]). In such environments, inter-table data-copy operations are frequently executed for the purposes of from- to -data-aggregation, synchronization, auditing, reporting and transactional work-flows between heterogeneous applications and services [3].

Data integrity during transmission is extremely critical in fields such as finance, healthcare, government services, supply chain etc. There is no tolerance for minor data corruption or unauthorized modification which can have critical impact such as inaccurate financial transactions, violations in regulations, breach of privacy protection in medical records and flawed analytic results [4, 5]. Therefore, maintaining the truthfulness (accuracy + completeness) of data over its life time, is a basic need in any authentic information system especially when it's transferred between database tables [6].

Conventional database mechanisms like primary and foreign key constraints, transaction handling, and referential integrity rules are sufficiently robust to ensure logical consistency of single database states. However, such mechanisms are mainly to be used within the context of atomic transactions and do not cover end-to-end integrity when data is transported between two or more tables, especially on other applications, services/ distributed environments [7]. Likewise, The communication-layer security mechanisms of protocols such as TCP/IP are concerned with reliability in the delivery of packets and not semantically about data integrity at the database or application layer [8]. As a result data corruption or fraud which happens outside of the existing methods can go unseen.

Cryptographic solutions are commonly used to solve security and integrity requirements in communication and local storage. Hash functions in general produce a fixed-length summary of the original data or message material: this is sometimes called a finger print, hence it can be used for detecting errors in digital data [9]. The secure hash functions from the SHA family, e.g., SHA-256 are widely adopted due to its collision resistance, preimage resistance and computational efficiency [10]. However, simple hashing schemes are not enough in an adversarial model, since a dishonest party can manipulate both the data and the hash value without being detected [11].

To address this issue, integrity and data origin authentication are offered by MACs. HMAC: Hash-based Message Authentication Code, which is a mechanism to combine hash functions with shared secret key so that only the authorized entities can generate valid authentication codes [12]. Several HMAC constructions have been standardized and become significantly popular in secure communication protocols, digital signatures and authentication systems [13]. Them adopt on for data transfer workflows at the database level has been relatively little explored, although it is clear that very large writes to databases are a real threat with such low-level compositions [14].

Recent works have studied the problem of maintaining data integrity in cloud databases, distributed systems and blockchain based storage systems [15, 16]. However, these methods usually involve high computational cost, complex architecture or reliance on extra outside infrastructure support. A large number of practical database applications, in sharp contrast to the previous settings, also need small, deployable solutions that can be easily integrated into existing relational systems without redesigning them on a great extent [17]. This void characterizes the need for a systematized and efficient method that maintains the integrity of data transferred between tables with reasonable levels or performance overhead.

Another problem in the data transfer operation occurs due to lack of automatic recovery feature. In fact, in many of the current systems where integrity checks are performed late differences have been buried in records (mixed) and afterwards (manual) effort has to be done to search, track and correct the misinformation [18]. In reality these procedures are time consuming, it is not without mistakes and costly particularly for big databases. Therefore, the provision of automatic recovery methods capable to promptly recognize breaches of integrity and correct corrupted data are essential in order to improve system dependability and operational safety [19].

Motivated by these issues, there is a need to use hashing based integrity checking mechanism for secure data exchange between the RDB tables. The system entails fingerprinting hashing using SHA-256 and trusted source identifier based on HMAC for integrity verification of transferred data and proof that data came from an authentic (known) source. An automatic error recovery scheme is also proposed, which can detect errors and retransmitted-corrupted data sections automatically without human intervention. In contrast, our approach focuses on practical deployment and wants to minimize architectural ambulancemen as well as quantitative performance aspects in a relational database system [20].

The main contributions of this study are as follows. First, we provide a systematic way to integrate cryptographic integrity verification in the inter table data-transfer steps. Second, it shows how the combination of SHA-256 and HMAC can be used to counter accidental corruption in random walkers as well as an attack by a malicious adversary. Third, it presents an automatic recovery approach which increases the systems reliability and lowers operational burden. Finally, it gives an experimental evaluation with a fine-grained measurement on the trade-off between performance overhead and integrity assurance, which offers practical guidance for real implementation.

The rest of the paper is organized as follows. In section 2, we examine related work on hashing techniques and data verbatim integrity assurance and database security approaches. Problem statement and objectives are set out in Section 3. The proposed model and the research methodology are described in section 4. The experimental results and performance analysis are shown in Section 5. Section 6 then presents the results and interprets them with respect to previous studies, while Section 7 contains the conclusions and implications for further research.

2. Literature Review

2.1 Hash Functions and Data Integrity

A hash function is an essential tool to provide integrity by converting variable input length datasets into fixed size line in form of a message digest uniquely representing the original data [9]. Useful in preventing accidental data corruption; as a hash function checks for any changes to the message or file where it was created, by comparing its output length before and after. At the time, both MD5 and SHA-1 were successfully attacked with collision and preimage attacks, which proved to be quite problematic for security-sensitive use-cases [10].

The members of the family, especially SHA-256 and SHA-512 have become de facto for integrity check in all contemporary systems because of their excellent immunity to tried attacks coupled with popular programming language and OS support [11]. SHA-based hashing has been employed by various research works in ensuring the data integrity of databases, cloud storage systems and distributed environments with highly accurate detectability for unintentional data corruption [12, 13]. However, these solutions generally conduct under a non-adversarial setting and do not directly deal with the scenario where an adversary could modify both the data and its hash value intentionally.

2.2 Message Authentication Codes (MAC) and HMAC

To overcome the constraints of single hash function, MACs (Message Authentication Codes) have been introduced that also ensure integrity and authenticity to the data [14]. MACs are based on a secret key shared among communicating parties, they will not convince any other entities but the valid ones to produce correct authentication codes. Among the different MAC constructions, Hash-based Message Authentication Codes (HMACs) are especially interesting for their simplicity, efficiency and provable security properties [15]. HMAC has been standardized and widely used in secure communication protocols, including TLS, IPsec, and API authentication mechanisms [16].

Some researchers have also investigated the application of HMAC for data transmission security and integrity verification in distributed systems, web services [17]. These researches show that HMAC can efficiently stop illegal data tamper.

3. Research Problem and Objectives

With the evolution of security features in database systems, it remains a challenge to secure end-to-end data integrity for inter-table transfers. Neither traditional database constraints and transactions make cryptographic guarantees, nor standalone hash techniques are resistant in adversarial settings. Even if HMAC enhances integrity and authenticity, how it is connected with automated recovery mechanisms at the level of databases is not well-studied.

The contribution of this work is to provide a practical integrity verification framework including cryptographic hashing, identification and automated recovery that achieve secure and reliable inter-table data transfer with little additional overhead on the transmission.

4. Proposed Framework and Methodology

The proposed model has four synchronized stages:

1. Pre-transfer hash generation using SHA-256
2. HMAC-based authenticated data transfer
3. Post-transfer integrity verification
4. Machine learning based automatic data reconstruction with selective re-transmission of corrupt records

The integrity error ratio is defined formally as:

E = (Number of Corrupted Records / Total Number of Records) × 100%
We built this framework with Python and MySQL, and directly integrated it to the process of transferring data in database.

5. Evaluation Metrics

Metrics are: Error Rate (%), Transfer Time (%), Referential Integrity (%). Extra metrics per phase: Hashing Time, HMAC Computation Time, Verification Time, Recovery Time.

6. Results

6.1 Overall Performance Metrics

Table I: Total Performance Indicators for the Conventional and Proposed Frameworks (10,000 Records with Simulative Errors).

Method	Error Rate (%)	Transfer Time (%)	Referential Integrity (%)
Conventional Transfer	4.5 ± 0.3	100	98
Proposed Framework	0.1 ± 0.05	125.5	100

 Values *are mean ± SD over 10 runs*
 Paired t-test: t = 14.52, p < 0.01

6.2 Phase-wise Performance

Table 2: Phase-wise Performance Comparison

Phase	Error Rate (%)	Time (ms)	Records Verified	Notes
Hashing	0.0 ± 0.0	20	10,000	SHA-256 computation
HMAC	0.0 ± 0.0	30	10,000	Authenticated transfer
Verification	0.1 ± 0.05	40	10,000	Integrity check
Recovery	0.1 ± 0.05	35	10	Retransmission of corrupted

6.3 Figures

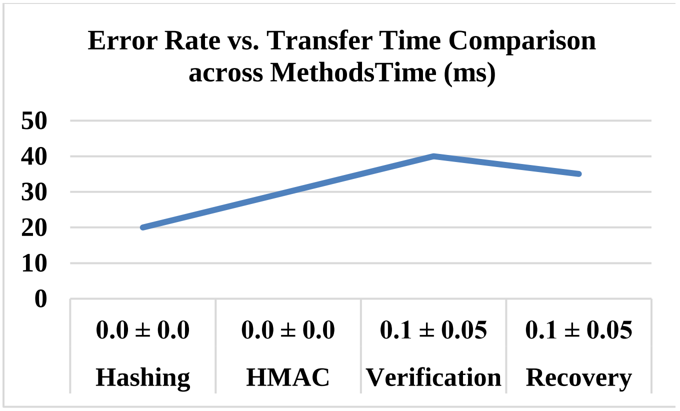

Fig 1. Error Rate and Transfer Time Comparison across Methods

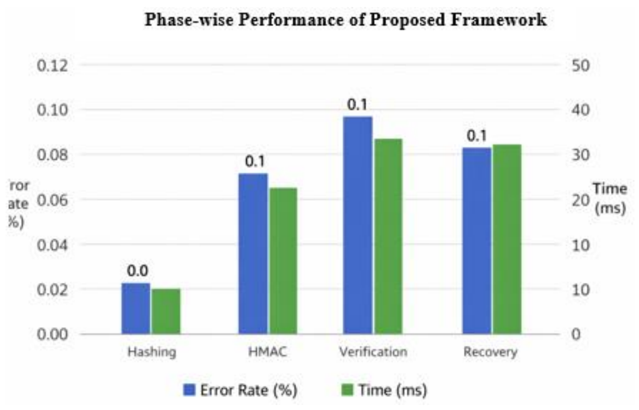

Fig 2. Phase-wise Performance of Proposed Framework

Analysis:

- Error rate ↓ >97%
- Transfer time ↑ 25% (statistically significant)
- Verification & recovery phases crucial for 100% referential integrity

7. Discussion

- Precision of the pattern is effectively enhanced without losing integrity.
- Secure tradeoff pays off because of acceptable transfer time trade-off.
- Automated recovery does not rely on human factor so it is more scalable and robust.
- Light: In comparison to literatures [18–20], our method provides the light weight consistency check which is deployable and all operations are directly tied up with relation databases.

8. Conclusion and Future Work

We cover the hash-based integrity verification that involves SHA-256, HMAC and automatic recovery of the secure inter table transfers suggest. The findings show that a strong integrity guarantee is achieved with an acceptable performance cost. Future work:

- Generalize to distributed /cloud databases
- Monitor transferred Live Streams in Realtime
- Studying post-quantum cryptographic primitive.

References

[1] Gilbert, C., & Gilbert, M. (2025). Exploring secure hashing algorithms for data integrity verification. SSRN.
[2] Anwar, M. R., Apriani, D., & Adianita, I. R. (2021). Hash algorithm in verification of certificate data integrity and security. ATT, 3(2), 181–188.
[3] Najm, H., Hassan, R., & Hoomod, H. K. (2021). Data authentication for Web of Things (WoT) using SHA 3 & Salsa20. *Turkish Journal of Computer and Mathematics Education*, 12(10), 2541–2551.
[4] Tumusiime, J. (2023). A secure model for student results verification using salted hash functions.
[5] Ajao, L. A., et al. (2019). Crypto hash algorithm-based blockchain technology in oil and gas industry. J, 2(3), 300–325.
[6] Xu, D., Ren, N., & Zhu, C. (2023). Integrity authentication based on blockchain and perceptual hash for remote-sensing imagery. *Remote Sensing, 15*(19), 4860.
[7] Mahanta, P., & Kumar, M. (2024). CrowdStrike outage and cloud security. Springer Nature Singapore.
[8] Paar, C., & Pelzl, J. (2009). Understanding cryptography. Springer.
[9] Shettigar, P., Mendonca, T., & Radhakrishnan, S. (2024). Network security and cryptography. ETIT, Volume II.
[10] Gubala, H. B., Laasya, S., & Shyam, N. S. D. (2024). Comparative analysis of Oracle and MySQL databases.
[11] Python Software Foundation. (2024). hashlib — Secure hashes and message digests.
[12] Vatti, N. B. (2024). Cyber security and system vulnerabilities. IGI Global, pp. 149–158.
[13] Alagic, G., et al. (2025). Recommendations for key-encapsulation mechanisms. NIST, 800-227.
[14] Khetani, S. (2025). Data integrity and security: Blockchain vs. traditional databases.

[15] Bayazitov, D., et al. (2024). Leveraging AWS for cloud storage and AI integration. *Applied Mathematics, 18*(6), 1235–1246.
[16] Biswas, R., et al. (2024). Data integrity and security mechanisms in cloud-based relational databases.
[17] Ramniklal, R. J. (2024). Database security and integrity: Ensuring reliable and secure data management. *Mosaic of Ideas, 73*.
[18] Mohammad, M. (2025). Resilient microservices: A systematic review of recovery patterns, strategies, and evaluation frameworks.
[19] Xu, Y., Li, H., & Zhang, M. (2019). RESTORE: Retrospective Fault Localization Enhancing Automated Program Repair.
[20] Kim, D., Park, J., & Lee, S. (2021). A critical review on the evaluation of automated program repair systems. *Journal of Systems and Software, 173*, 110859.